# 12th Workshop on Current Trends in Cryptology (CTCrypt 2023)


June 6-9, 2023, Volgograd, Russia

## Pre-proceedings

## CTCrypt 2023 is organized by

– Academy of Cryptography of the Russian Federation

– Steklov Mathematical Institute of Russian Academy of Science

– Technical Committee for Standardization «Cryptography and security mechanisms» (TC 026)

# Steering Committee

## Co-chairs

| | | |
|---|---|---|
| Aleksandr Shoitov | – | Academy of Cryptography of the Russian Federation, Russia |
| Vladimir Sachkov | – | Academy of Cryptography of the Russian Federation, Russia |
| Igor Kachalin | – | TC 026, Russia |

## Steering Committee Members

| | | |
|---|---|---|
| Andrey Zubkov | – | Steklov Mathematical Institute of RAS, Russia |
| Dmitry Matyukhin | – | TC 026, Russia |
| Andrey Pichkur | – | Federal Educational and Methodical Association in System of Higher Education on Information Security, Russia |

# Program Committee

## Co-chairs

| | | |
|---|---|---|
| Alexander Lapshin | – | Academy of Cryptography of the Russian Federation, Russia |
| Dmitry Matyukhin | – | TC 026, Russia |
| Andrey Zubkov | – | Steklov Mathematical Institute of RAS, Russia |

## Program Committee Members

| | | |
|---|---|---|
| Alexey Alexandrov | – | Vladimir State University named after Alexander and Nikolay Stoletovs, Russia |
| Sergey Checheta | – | Federal Educational and Methodical Association in System of Higher Education on Information Security, Russia |
| Alexander Cheryomushkin | – | Academy of Cryptography of the Russian Federation, Russia |
| Ivan Chizhov | – | Lomonosov Moscow State University, Russia |
| Vladimir Fomichev | – | "Security Code", LLC, Russia |
| Yury Kharin | – | Research Institute for Applied Problems of Mathematics and Informatics, Belarus |
| Grigory Marshalko | – | TC 026, Russia |
| Eduard Primenko | – | Lomonosov Moscow State University, Russia |
| Igor Semaev | – | The University of Bergen, Norway |
| Vasily Shishkin | – | "NPK Kryptonite", JSC, Russia |
| Stanislav Smyshlyaev | – | "Crypto-Pro", LLC, Russia |
| Alexey Tarasov | – | Federal Educational and Methodical Association in System of Higher Education on Information Security, Russia |
| Natalia Tokareva | – | Sobolev Institute of Mathematics SB RAS, Russia |
| Andrey Trishin | – | "Certification Research Center", LLC, Russia |
| Alexey Urivskiy | – | "InfoTeCS", JSC, Russia |
| Amr Youssef | – | Concordia University, Canada |
| Andrey Zyazin | – | Russian Technological University (MIREA), Russia |

## External Reviewers

Liliya Akhmetzyanova, Evgeny Alekseev, Alexandra Babueva, Andrey Bozhko, Dmitriy Burov, Sergey Grebnev, Sergey Kyazhin, Lidiya Nikiforova, Maxim Nikolaev, Vladimir Rudskoy, Dmitriy Trifonov.

# Invited Talks

# Adapted spectral-differential method for constructing differentially 4-uniform piecewise-linear substitutions, orthomorphisms, involutions over the field $\mathbb{F}_{2^n}$

Andrey Menyachikhin

TVP Laboratories, Moscow, Russia
and88@list.ru

**Abstract**

The substitution block ($s$-box) is one of the basic cryptographic components which plays an important role in fulfilling the Shannon's property of confusion in modern block ciphers. We introduce a new method for generating $s$-boxes with low differential uniformity and present our research findings in this report organized as follows.

Well-known approaches to constructing differentially 4-uniform $s$-boxes over the field $\mathbb{F}_{2^n}$ are discussed in the introduction.

The first part of the report is devoted to the problem of efficient computation of the linear and differential spectra of piecewise linear substitutions.

In the second part, we combine an algebraic and heuristic approaches to the construction of block cipher confusion components and present a new tool for constructing $s$-boxes with low differential uniformity. This tool is called adapted spectral-differential method.

The third part examines the linear equivalence problem for partially given piecewise-linear permutations.

In the fourth part of the report, we give lower and upper bounds on the differential characteristic of piecewise-linear permutations over the field $\mathbb{F}_{2^n}$.

Finally, we present a large number of pairwise CCZ-inequivalent differentially 4-uniform $s$-boxes over the field $\mathbb{F}_{2^8}$. We also present the first known example of a differentially 4-uniform orthomorphism over $\mathbb{F}_{2^8}$.

**Keywords:** substitution block, 4-uniform $s$-box.

# Contents

## Protocols



## Symmetric Cryptography

### Linear transformations

# Protocols

# Blind signature as a shield against backdoors in smart-cards

Liliya Akhmetzyanova, Evgeny Alekseev, Alexandra Babueva, Andrey Bozhko, and Stanislav Smyshlyaev

CryptoPro LLC, Moscow, Russia
{lah, alekseev, babueva, bozhko, svs}@cryptopro.ru

**Abstract**

The paper considers the problem of signature forgery (including signature key recovery) in case of presence backdoors in the hardware or software of functional key carriers (smart-cards). A new approach to solving the problem based on using blind signature schemes is proposed. It is shown that weak blindness and weak unforgeability of the blind signature schemes imply security against backdoors in smart-cards. As a concrete example, we consider blind version of the GOST signature scheme (the blind signature scheme proposed by Camenisch) and show that this scheme is resistant to backdoors under one single assumption that GOST is secure in the standard sense.

**Keywords:** Blind signature scheme, GOST R 34.10-2012, untrusted smart-cards, backdoors.

## 1 Introduction

Consider an information system consisting of two components: a smart-card (or token) used as a functional key storage and an application installed on a user device (desktop or handheld). The applied function of a system is to compute a signature of any document transmitting via the application with a key uploaded and stored on a smart-card. The components usually interact in the following way:

1. The user opens the application, chooses the document to be signed and pushes the button «Sign».
2. The application connects with the smart-card (usually by establishing a password protected secure channel, for detail see [8]) and sends to it the chosen document or the document hash value.
3. The smart-card computes the signature value of the document on its own under a stored private key and returns the computed value to the application.

4. The application verifies the received signature value and returns the signed document to the user.

Using smart-cards with unrecoverable private key cryptography «on board» is considered one of the most secure key management approach that allows to protect against adversaries which can get physical access to key storage devices. However, it has its own disadvantages. Unlike software applications which can be open-sourced and, therefore, fully verified, self-reliantly compiled and trustly installed by anyone, developing of smart-cards is much more technically difficult process that is usually implemented by specific companies specializing in this area. Indeed, the signing code is often hardwired directly into smart-card microchips to improve performance and, consequently, cannot be openly verified by outsiders: the users are given a ready-to-use «black-box» device. This makes it possible for unscrupulous developers to implement a malicious code.

In the current paper we address the security issues that arise when the smart card being used is seen as an untrusted component and is believed to contain backdoors. In the context of systems based on ElGamal-like or Schnorr-like signature schemes these issues are highly crucial, since this type of signatures use one-time random values which are generated with the smart-card and which compromise immediately leads to a user private key recovery. For instance, malicious smart-card can use low-entropy one-time values allowing an adversary (e.g. company implementing this backdoor) to carry out the brute force attack and recover the user key from one correct signature.

**Related work.** The paper [2] is devoted to these issues. Firstly, the paper introduces two types of adversary to be considered:

> **External adversary:** it models an honest-but-curious adversary acting on the application side; the adversary's goal is to make a new correct pair (message, signature) without interacting with a smart-card or, in other words, to make a forgery. Note that this threat includes the stronger one – key recovery. Considering of such adversaries covers the scenario where only honest user interacts with smart-card through verified and trusted application, but this application is less protected against memory leaking compared to the smart-card. To formalise the security against such adversaries, a security notion we called «robustness» is introduced.
>
> **Remark 1.** *Note that this type does not cover the capabilities of active adversaries that can directly interact (e.g. using its own malicious application) with the smart-card. In practice it means that the adversary that steals the smart-card cannot get access to its API. Considering of passive adversaries only is justified by the fact, that smart-cards are usually also protected with*

*a memorable password that should be entered by the human to get access to its API (see [3]).*

**Adversary with agent:** this adversary is supposed to consist of two parts. The first part is *a fully active adversary* on the smart-card side but it can interacts only with the trusted application, i.e. there is no other channel for data transmition from smart-card. The second part collects the pairs (message, signature) computed by application and malicious smart-card - it's agent. Similar to the first type of adversary, the goal is to make a forgery. To formalise the security against such adversaries, a security notion we called «backdoor resilience» is introduced.

In order to deal with these adversaries the paper [2] proposed a solution for the GOST signature scheme [9] based on the usage of the interactive Schnorr zero-knowledge proof. This protocol is executed with the main signing algorithm, its aim is to prove to the application that smart-card uses the «correct» one-time value (for details see the original paper). This solution has the following two significant drawbacks:

1. it allows to protect against *the semi-trusted* smart-card only: the crucial assumption for security is that low-level (short) arithmetic operations are implemented correctly in the smart-cards. Although it is realistic assumption, there are no convenient ways to validate this on practice.

2. it is not secure if the smart-card can terminate the signing process with the error on the application side. The paper [2] describes the concrete attack where the malicious smart-card successfully completes the signing protocol only if certain bits of resulting signature are equal to certain bits of the signing key. One approach to protect against this attack is to delete the private signing key immediately after such errors occur. However, in practice, errors can occur not only due to the adversary's actions, but also due to technical failures, so deleting the key after each error is not a practical solution.

**Our contribution.** In order to negate the disadvantages mentioned above we propose a new approach which main idea is to use the «blind versions» of the signature schemes. The blind signature schemes firstly introduced by Chaum [5] allow one party called User to obtain a signature for an arbitrary message after interacting with another party called Signer holding a signing key in such a way that the Signer does not receive any information about either the message or the signature value (blindness property) and the User can compute only one single signature per interaction with the Signer (unforgeability property).

In the context of considered signing system the smart-card executes the Signer side and the application executes the User side. Due to the blindness property,

the malicious smart-card learns no information about the signature during the protocol execution and, therefore, cannot «control» the signature values, e.g. by covertly transmiting bits of private key through the signature values. In particular, protection is achieved even if the smart-card performs any arbitrary algorithm, i.e. maliciously implements low-level arithmetic operations or generates low-entropy one-time random values.

In the current paper we give the formal definitions of security notions – robustness and backdoor resilience corresponding to external adversary and adversary with agent. After that we perform a formal analysis of the proposed solution regarding introduced security notions: we show that weak blindness (where an adversary cannot affect the key generation algorithm) and weak unforgeability (in non-concurrent setting against honest-but-curious adversaries) of the blind signature schemes imply both robustness and backdoor resilience. Moreover, for the GOST signature scheme we propose the concrete blind signature scheme for usage – the Camenisch's scheme [4] that provides perfect (strong) blindness and weak unforgeability that implied only by the unforgeability of GOST. It means that the Camenisch's blind signature scheme provides robustness and backdoor resilience under one single assumption that the GOST signature scheme provides standard security, i.e. is unforgeable under chosen message attack.

The rest of the paper is organised as follows. In Section 2 we remind the definitions of conventional and blind signature schemes, the accompanying security notions are given. In Section 3 the formal definitions of robustness and backdoor resilience are introduced. Section 4 is devoted to the formal analysis and Section 5 considers the Camenish's blind signature scheme in details.

## 2 Basic definitions

**(Conventional) signature schemes.** The conventional signature scheme $\mathsf{Sig}$ is determined by three algorithms:

- $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Sig.KGen}(\ )$: a key generation algorithm that outputs a secret key $\mathsf{sk}$ and a public key $\mathsf{pk}$;
- $\sigma \leftarrow \mathsf{Sig.Sign}(\mathsf{sk}, m)$: a signature generation algorithm that takes a secret key $\mathsf{sk}$, and a message $m$ and returns a signature $\sigma$.
- $b \leftarrow \mathsf{Sig.Vf}(\mathsf{pk}, m, \sigma)$: a (deterministic) verification algorithm that takes a public key $\mathsf{pk}$, a message $m$, and a signature $\sigma$, and returns 1 if $\sigma$ is valid on $m$ under $\mathsf{pk}$ and 0 otherwise.

*Correctness.* We say that $\mathsf{Sig}$ is correct if for every message $m$, with probability one over the sampling of parameters and the key pair $(\mathsf{sk}, \mathsf{pk})$ the equality $\mathsf{Sig.Vf}(\mathsf{pk}, m, \mathsf{Sig.Sign}(\mathsf{sk}, m)) = 1$ holds.

**Blind signature schemes.** The blind signature scheme $\mathsf{BS}$ is defined in the same way as the conventional signature scheme except for the signature generation algorithm which is replaced by the following protocol:

– $(b, \sigma) \leftarrow \langle \mathsf{BS.Signer}(\mathsf{sk}), \mathsf{BS.User}(\mathsf{pk}, m)\rangle$: an interactive signing protocol that is run between a Signer with a secret key $\mathsf{sk}$ and a User with a public key $\mathsf{pk}$ and a message $m$; the Signer outputs $b = 1$ if the interaction completes successfully and $b = 0$ otherwise, while the User outputs $\sigma$ that is either the resulting signature or an error message.

*Correctness.* We say that $\mathsf{BS}$ is correct if for every message $m$, with probability one over the sampling of parameters and the key pair $(\mathsf{sk}, \mathsf{pk})$, the signing protocol $\langle \mathsf{BS.Signer}(\mathsf{sk}), \mathsf{BS.User}(\mathsf{pk}, m)\rangle$ completes with $(1, \sigma)$, $\sigma \neq \perp$, such that $\mathsf{BS.Vf}(\mathsf{pk}, m, \sigma) = 1$.

In the current paper we are interested in the blind signature schemes that are built basing on some conventional signature schemes. We will say that the $\mathsf{BS}$ scheme is *a blind version* of the $\mathsf{Sig}$ scheme, if the $\mathsf{KGen}$ and $\mathsf{Vf}$ algorithms of these schemes coincides and for any $(\mathsf{sk}, \mathsf{pk})$, any message $m$ and any signature $\sigma$

$$\Pr\left[(1, \sigma) \leftarrow \langle \mathsf{BS.Signer}(\mathsf{sk}), \mathsf{BS.User}(\mathsf{pk}, m)\rangle\right] = \Pr\left[\sigma \leftarrow \mathsf{Sig.Sign}(\mathsf{sk}, m)\right],$$

where the corresponding probability spaces are determined by the randomness used in the signing protocol and signing algorithm.

*Three-move blind signature schemes.* For simplicity this paper focuses on three-move blind signature schemes. For such schemes the signing protocol can be described as follows:

$$\begin{aligned}
&(msg_{S,1}, state_S) \leftarrow \mathsf{BS.Signer}_1(\mathsf{sk})\\
&(msg_U, state_U) \leftarrow \mathsf{BS.User}_1((\mathsf{pk}, m), msg_{S,1})\\
&(msg_{S,2}, b) \leftarrow \mathsf{BS.Signer}_2(state_S, msg_{U,1})\\
&\sigma \leftarrow \mathsf{BS.User}_2(state_U, msg_{S,2})
\end{aligned}$$

where $msg_{role,i}, role \in \{U, S\}$, is the $i$-th message sent by the side with role $role$ during the protocol execution. The variable $state_{role}$ is aimed to keep the internal state for using on the next protocol stage. Here the User performs the $\mathsf{BS.User}_1$ and $\mathsf{BS.User}_2$ functions, and the Signer performs the $\mathsf{BS.Signer}_1$ and $\mathsf{BS.Signer}_2$ functions during the protocol execution.

**Security notions.** Hereinafter we describe the security notions using a game-based approach (for detail, see [10]). This approach uses the notion of «experiment» played between a challenger and an adversary. The adversary and challenger are modelled using consistent interactive probabilistic algorithms. The challenger simulates the functioning of the analysed cryptographic scheme for the adversary and

may provide him access to one or more oracles. The parameters of an adversary $\mathcal{A}$ are its computational resources (for a fixed model of computation and a method of encoding) and oracles query complexity. The query complexity usually includes the number of queries. Denote by $\mathsf{Adv}^{\mathrm{M}}_{S}(\mathcal{A})$ the measure of the success of the adversary $\mathcal{A}$ in realizing a certain threat, defined by the security notion M for the cryptographic scheme S.

The standard security notion for (probabilistic) signature schemes is strong unforgeability under chosen message attack (sUF-CMA). The formal definition is given below.

**Definition 1.** *For an adversary $\mathcal{A}$ and a signature scheme* Sig*:*

$$\mathsf{Adv}^{\text{sUF-CMA}}_{\mathsf{Sig}}(\mathcal{A}) = \Pr\left[\mathbf{Exp}^{\text{sUF-CMA}}_{\mathsf{Sig}}(\mathcal{A}) \to 1\right],$$

*where the* $\mathbf{Exp}^{\text{sUF-CMA}}_{\mathsf{Sig}}(\mathcal{A})$ *experiment is defined in the following way:*

$\mathbf{Exp}^{\text{sUF-CMA}}_{\mathsf{Sig}}(\mathcal{A})$

1: $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Sig.KGen}()$
2: $\mathcal{L} \leftarrow \varnothing$
3: $(m, \sigma) \leftarrow \mathcal{A}^{Sign}(\mathsf{pk})$
4: **if** $(m, \sigma) \in \mathcal{L}$ : **return** 0
5: **return** $\mathsf{Sig.Vf}(\mathsf{pk}, m, \sigma)$

*Oracle Sign*$(m)$

1: $\sigma \leftarrow \mathsf{Sig.Sign}(\mathsf{sk}, m)$
2: $\mathcal{L} \leftarrow \mathcal{L} \cup \{(m, \sigma)\}$
3: **return** $\sigma$

**Remark 2.** *The same security notion can be applied to the blind version* BS *of the signature scheme* Sig*. In this case the line 1 in the Sign oracle is replaced with the line* $(1, \sigma) \leftarrow \langle \mathsf{BS.Signer}(\mathsf{sk}), \mathsf{BS.User}(\mathsf{pk}, m)\rangle$*. It is easy to see that for such schemes* sUF-CMA*-security of the* Sig *scheme implies* sUF-CMA*-security of the* BS *scheme and vice versa.*

The standard notions for blind signature schemes are one-more unforgeability and blindness, their formal definitions can be found in [11]. In the current paper we consider only weak versions of these notions: weak unforgeability wUNF and weak blindness wBL.

*Weak unforgeability.* The weak unforgeability considers only an honest-but-curious adversary acting on the User side. This adversary can adaptively choose messages to be signed by making a query $m$ to the oracle and obtain in return a signature $\sigma$ and a specific value *view*. Thy latter consits of all incoming messages and values of all random parameters that are processing and sampling on the User side during the signing protocol execution. It is easy to see that for any blind signature scheme wUNF-security implies sUF-CMA-security. The formal definition of wUNF is given below.

**Definition 2.** *For an adversary $\mathcal{A}$ and a blind signature scheme* BS*:*

$$\mathsf{Adv}^{\text{wUNF}}_{\mathsf{BS}}(\mathcal{A}) = \Pr\left[\mathbf{Exp}^{\text{wUNF}}_{\mathsf{BS}}(\mathcal{A}) \to 1\right],$$

*where the* $\mathbf{Exp}^{\mathrm{wUNF}}_{\mathsf{BS}}(\mathcal{A})$ *experiment is defined in the following way:*

$\mathbf{Exp}^{\mathrm{wUNF}}_{\mathsf{BS}}(\mathcal{A})$

1: $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{BS.KGen}()$
2: $\mathcal{L} \leftarrow \varnothing$
3: $(m, \sigma) \leftarrow \mathcal{A}^{Sign}(\mathsf{pk})$
4: **if** $(m, \sigma) \in \mathcal{L}$ : **return** 0
5: **return** $\mathsf{BS.Vf}(\mathsf{pk}, m, \sigma)$

*Oracle* $Sign(m)$

1: $(1, (\sigma; view)) \leftarrow \langle \mathsf{BS.Signer}(\mathsf{sk}), \mathsf{BS.User}(\mathsf{pk}, m) \rangle$
2: $\mathcal{L} \leftarrow \mathcal{L} \cup \{(m, \sigma)\}$
3: **return** $\sigma, view$

*Weak blindness.* Informally, the blind signature scheme provides blindness if there is no way to link a (message, signature) pair to the certain execution of the signing protocol. In the context of strong notion the adversary can fully control the Signer side. Further we consider a weaker notion where the adversary cannot affect key generation algorithm. The formal definition is given below.

**Definition 3.** *For an adversary* $\mathcal{A}$ *and three-move blind scheme* $\mathsf{BS}$

$$\mathsf{Adv}^{\mathrm{wBL}}_{\mathsf{BS}}(\mathcal{A}) = \Pr\left[\mathbf{Exp}^{\mathrm{wBL},1}_{\mathsf{BS}}(\mathcal{A}) \to 1\right] - \Pr\left[\mathbf{Exp}^{\mathrm{wBL},0}_{\mathsf{BS}}(\mathcal{A}) \to 1\right],$$

*where the* $\mathbf{Exp}^{\mathrm{wBL},b}_{\mathsf{BS}}(\mathcal{A}),\ b \in \{0,1\}$, *experiments are defined in the following way:*

$\mathbf{Exp}^{\mathrm{wBL},b}_{\mathsf{BS}}(\mathcal{A})$

1: $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{BS.KGen}()$
2: $b_0 \leftarrow b$
3: $b_1 \leftarrow 1 - b$
4: $b' \leftarrow \mathcal{A}^{Init, User_1, User_2}(\mathsf{sk}, \mathsf{pk})$
5: **return** $b'$

*Oracle* $Init(m_0, m_1)$

1: $sess_0 \leftarrow \mathsf{init}$
2: $sess_1 \leftarrow \mathsf{init}$

*Oracle* $User_1(i, msg)$

1: **if** $i \notin \{0,1\} \vee sess_i \neq \mathsf{init}$ : **return** $\perp$
2: $sess_i \leftarrow \mathsf{open}$
3: $(msg_i, state_i) \leftarrow \mathsf{BS.User}_1((\mathsf{pk}, m_{b_i}), msg)$
4: **return** $msg_i$

*Oracle* $User_2(i, msg)$

1: **if** $sess_i \neq \mathsf{open}$ : **return** $\perp$
2: $sess_i \leftarrow \mathsf{closed}$
3: $\sigma_{b_i} \leftarrow \mathsf{BS.User}_2(state_i, msg)$
4: **if** $sess_0 = sess_1 = \mathsf{closed}$ :
5: **if** $\sigma_{b_0} = \perp \vee \sigma_{b_1} = \perp$ : **return** $(\perp, \perp)$
6: **return** $(\sigma_0, \sigma_1)$
7: **return** $\varepsilon$

## 3 Security notions

In the current section we give the formal game-based definitions of two security notions: backdoor resilience and robustness.

**Backdoor resilience/Security against adversary with agent.** Consider an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ consisting of two algorithms. An algorithm $\mathcal{A}_2$ denotes the part of the adversary $\mathcal{A}$ collecting signature values for adaptively chosen messages. An algorithm $\mathcal{A}_1$ denotes the agent acting on the backdoored smart-card side.

The formal definition of BDres (BackDoor resilience) for blind signature schemes is given below (see Definition 4). We parametrize this security model by value $k$ determining the number of challenger's attempts to produce correct signature for one message (details are described below).

**Definition 4.** *For any adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *and blind signature scheme* $\mathsf{BS}$*:*

$$\mathsf{Adv}^{\mathrm{BDres}_k}_{\mathsf{BS}}(\mathcal{A}) = \Pr\left[\mathbf{Exp}^{\mathrm{BDres}_k}_{\mathsf{BS}}(\mathcal{A}) \to 1\right],$$

*where the* $\mathbf{Exp}^{\mathrm{BDres}_k}_{\mathsf{BS}}(\mathcal{A})$, $k \in \mathbb{N}$, *experiment is defined in the following way:*

$\mathbf{Exp}^{\mathrm{BDres}_k}_{\mathsf{BS}}(\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2))$

1: $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{BS.KGen}()$
2: $\mathcal{L} \leftarrow \varnothing$
3: $\mathsf{lost} \leftarrow \mathsf{false}$
4: $st \leftarrow \mathcal{A}_1(\mathsf{sk}, \mathsf{pk})$
5: $(m, \sigma) \xleftarrow{\$} \mathcal{A}_2^{Sign}(\mathsf{pk})$
6: **if** $((m, \sigma) \in \mathcal{L}) \vee (\mathsf{lost} = \mathsf{true})$:
7: **return** 0
8: **return** $\mathsf{BS.Vf}(\mathsf{pk}, m, \sigma)$

*Oracle* $Sign(m)$

1: $i \leftarrow 0$
2: **do**
3: $(st, \sigma) \leftarrow \langle \mathcal{A}_1(st), \mathsf{BS.User}(\mathsf{pk}, m) \rangle$
4: $i \leftarrow i + 1$
5: **until** $(i \geqslant k) \vee (\sigma \neq \perp)$
6: **if** $\sigma = \perp$:
7: $\mathsf{lost} \leftarrow \mathsf{true}$
8: **return** $\perp$
9: $\mathcal{L} \leftarrow \mathcal{L} \cup \{(m, \sigma)\}$
10: **return** $\sigma$

At the experiment initialisation stage (line 1) the challenger modeling an honest application generates a key pair $(\mathsf{sk}, \mathsf{pk})$ according to the key generation algorithm and sends to $\mathcal{A}_1$ a pair $(\mathsf{sk}, \mathsf{pk})$ (line 4), while to $\mathcal{A}_2$ it sends a verification key $\mathsf{pk}$ only (line 5). This stage models the trusted process of generating keys, issuing corresponding certificate and uploading key material on the smart-card.

The $\mathcal{A}_2$ algorithm can make queries to the challenger signing oracle $Sign$ that returns signature values $\sigma$ for messages $m$ arbitrarily chosen by the adversary. Every signature value is computed during the execution of blind signing protocol between oracle that models the honest User side and the $\mathcal{A}_1$ algorithm modeling the malicious Signer side (line 3 in the oracle). Here the variable $st$ denotes the internal state of $\mathcal{A}_1$ that is kept from call to call.

The $\mathcal{A}_1$ algorithm is allowed to terminate the protocol execution with error $\perp$ on the User side (see line 5 in the oracle). For this reason for any requested message $m$ the oracle makes $k$ attempts to compute a correct signature, and in the case when all $k$ attempts fail, challenger returns 0 as a game result (meaning that

the adversary loses, see line 7 in the oracle). This simulates the scenario where the smart-card has failed and is no longer being used.

**Remark 3.** *Note that if the algorithm $\mathcal{A}_2$ can obtain errors from the signing oracle then there is always a trivial attack. Consider the agent $\mathcal{A}_1$ that successfully completes the signing protocol execution iff $i$-th bit of $\mathsf{sk}$ is equal to 1, where $i$ is a sequence number of query to oracle. Having such an agent on the smart-card side the $\mathcal{A}_2$ algorithm can recover all bits of signing key and trivially make a forgery.*

To break a backdoor resilience the algorithm $\mathcal{A}_2$ is needed to make a forgery $(m, \sigma)$ containing a signature $\sigma$ that was not previously returned by the oracle *Sign* in response to a query $m$.

**Robustness/Security against external adversary.** The formal definition of robustness is totally coincides with definition of weak unforgeability for blind signature scheme (see Definition 2). Note that according to this notion the adversary

1. can obtain signatures $\sigma$ and all values processing on the User side for arbitrarily chosen messages $m$, this simulates the scenario where the adversary gets an access to the memory of trusted application;
2. cannot open many parallel executions of signing protocol, this also fits practice since the smart-cards are usually low resource devices and can execute only one session.

# 4 Security analysis

## 4.1 Backdoor resilience/Security against adversary with agent

In this section we prove that weak blindness and standard unforgeability (sUF-CMA) imply backdoor resilience.

**Theorem 1.** *Fix $k \in \mathbb{N}$. For any adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ in the* $\mathrm{BDres}_k$ *model with summary time complexity at most $t$ making at most $q$ queries to the signing oracle, there exist an adversary $\mathcal{B}$ in the* sUF-CMA *model making at most $q$ queries to the signing oracle and an adversary $\mathcal{C}$ in the* wBL *model such that*

$$\mathsf{Adv}_{\mathsf{BS}}^{\mathrm{BDres}_k}(\mathcal{A}) \leqslant \mathsf{Adv}_{\mathsf{BS}}^{\text{sUF-CMA}}(\mathcal{B}) + q \cdot k \cdot \mathsf{Adv}_{\mathsf{BS}}^{\text{wBL}}(\mathcal{C}).$$

*Time complexities of $\mathcal{B}$ and $\mathcal{C}$ are at most $t$ and $tkq$ correspondingly.*

**Remark 4.** *If the blind signature scheme provides perfect blindness (i.e. $\mathsf{Adv}_{\mathsf{BS}}^{\text{wBL}}(\mathcal{C}) = 0$ for any $\mathcal{C}$ with any time complexity), then the bound is transformed as follows*

$$\mathsf{Adv}_{\mathsf{BS}}^{\mathrm{BDres}_k}(\mathcal{A}) \leqslant \mathsf{Adv}_{\mathsf{BS}}^{\text{sUF-CMA}}(\mathcal{B}).$$

*From the perspective of using conventional signature scheme* $\mathsf{Sig}$*, this inequality means that in order to provide backdoor resilience it is enough for this signature scheme to have its blind version* $\mathsf{BS}$ *(with* $\mathsf{Adv}^{\text{sUF-CMA}}_{\mathsf{BS}}(\mathcal{B}) = \mathsf{Adv}^{\text{sUF-CMA}}_{\mathsf{Sig}}(\mathcal{B})$*) and to be unforgable in the standard model. Note, that the bound does not depend on* $k$*, so this value can be chosen arbitrarily by the application developers.*

**Remark 5.** *For clarity, the proof is carried out for three-move blind signatures, but the proof does not base on any specific features of such scheme type and can be easily adapted for any-move blind signatures.*

*Proof.* The proof consits of two parts.

**Part 1.** Consider the consequence of several experiments where each next experiment slightly differs from the previous one.

**Game 0.** Let $\mathbf{Exp}^{0}_{\mathsf{BS}}(\mathcal{A}) = \mathbf{Exp}^{\text{BDres}_k}_{\mathsf{BS}}(\mathcal{A})$.

**Game 1.** Consider the following modified experiment $\mathbf{Exp}^{1}_{\mathsf{BS}}(\mathcal{A})$:

$\mathbf{Exp}^{1}_{\mathsf{BS}}(\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2))$

1 : $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{BS.KGen}()$
2 : $\mathcal{L} \leftarrow \varnothing$
3 : $\mathsf{lost} \leftarrow \mathsf{false}$
4 : $st \leftarrow \mathcal{A}_1(\mathsf{sk}, \mathsf{pk})$
5 : $(m, \sigma) \xleftarrow{\$} \mathcal{A}_2^{Sign}(\mathsf{pk})$
6 : **if** $((m, \sigma) \in \mathcal{L}) \vee (\mathsf{lost} = \mathsf{true})$:
7 : **return** 0
8 : **return** $\mathsf{BS.Vf}(\mathsf{pk}, m, \sigma)$

Oracle $Sign(m)$

1 : $i \leftarrow 0$
2 : **do**
3 : $(st, \sigma) \leftarrow \langle \mathcal{A}_1(st), \mathsf{BS.User}(\mathsf{pk}, m) \rangle$
4 : **if** $\sigma \neq \perp$:
5 : $(1, \sigma) \leftarrow \langle \mathsf{BS.Signer}(\mathsf{sk}), \mathsf{BS.User}(\mathsf{pk}, m) \rangle$
6 : $i \leftarrow i + 1$
7 : **until** $(i \geqslant k) \vee (\sigma \neq \perp)$
8 : **if** $\sigma = \perp$:
9 : $\mathsf{lost} \leftarrow \mathsf{true}$
10 : **return** $\perp$
11 : $\mathcal{L} \leftarrow \mathcal{L} \cup \{(m, \sigma)\}$
12 : **return** $\sigma$

$\mathbf{Exp}^{1}_{\mathsf{BS}}(\mathcal{A})$ differs from $\mathbf{Exp}^{0}_{\mathsf{BS}}(\mathcal{A})$ in additional lines 4 and 5 of the $Sign$ oracle code. If the oracle, interacting with the agent $\mathcal{A}_1$ as a user, completes the signing protocol with a correct signature, then the oracle recomputes a new signature honestly executing the signing protocol on its own (without interaction with the agent). The second part of the proof is devoted to estimation of winning probability difference for $\mathbf{Exp}^{1}_{\mathsf{BS}}(\mathcal{A})$ and $\mathbf{Exp}^{0}_{\mathsf{BS}}(\mathcal{A})$ .

**Game 2.** Consider the next modification: the experiment $\mathbf{Exp}^{2}_{\mathsf{BS}}(\mathcal{A})$. Here the oracle always responses to requests of $\mathcal{A}_2$ with a correct honestly generated signature even in the case when $\mathcal{A}_1$ provokes errors $k$ times in a row that sets the flag $\mathsf{lost}$ in $\mathbf{Exp}^{1}_{\mathsf{BS}}(\mathcal{A})$.

$\mathbf{Exp}^{2}_{\mathsf{BS}}(\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2))$

1 : $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{BS.KGen}()$
2 : $\mathcal{L} \leftarrow \varnothing$
3 : $st \leftarrow \mathcal{A}_1(\mathsf{sk}, \mathsf{pk})$
4 : $(m, \sigma) \xleftarrow{\$} \mathcal{A}_2^{Sign}(\mathsf{pk})$
5 : **if** $((m, \sigma) \in \mathcal{L})$:
6 : **return** 0
7 : **return** $\mathsf{BS.Vf}(\mathsf{pk}, m, \sigma)$

Oracle $Sign(m)$

1 : $i \leftarrow 0$
2 : **do**
3 : $(st, \sigma) \leftarrow \langle \mathcal{A}_1(st), \mathsf{BS.User}(\mathsf{pk}, m) \rangle$
4 : $i \leftarrow i + 1$
5 : **until** $(i \geqslant k) \vee (\sigma \neq \perp)$
6 : $(1, \sigma) \leftarrow \langle \mathsf{BS.Signer}(\mathsf{sk}), \mathsf{BS.User}(\mathsf{pk}, m) \rangle$
7 : $\mathcal{L} \leftarrow \mathcal{L} \cup \{(m, \sigma)\}$
8 : **return** $\sigma$

For this experiment:

$$\Pr\left[\mathbf{Exp}^{1}_{\mathsf{BS}}(\mathcal{A}) \to 1\right] = \Pr\left[\mathbf{Exp}^{1}_{\mathsf{BS}}(\mathcal{A}) \to 1 \wedge (\mathsf{lost} = \mathsf{false})\right] + \\ + \underbrace{\Pr\left[\mathbf{Exp}^{1}_{\mathsf{BS}}(\mathcal{A}) \to 1 \wedge (\mathsf{lost} = \mathsf{true})\right]}_{= 0 \text{ due to line 6 of } \mathbf{Exp}^{1}_{\mathsf{BS}}(\mathcal{A})} \leqslant \Pr\left[\mathbf{Exp}^{2}_{\mathsf{BS}}(\mathcal{A}) \to 1\right].$$

**Game 3.** Note that in the $\mathbf{Exp}^{2}_{\mathsf{BS}}(\mathcal{A})$ experiment the agent $\mathcal{A}_1$ can be thrown away since it cannot influence on the signature value anymore (see the $\mathbf{Exp}^{3}_{\mathsf{BS}}(\mathcal{A}_2)$ experiment below). Note that $\Pr\left[\mathbf{Exp}^{2}_{\mathsf{BS}}(\mathcal{A}_1, \mathcal{A}_2) \to 1\right] = \Pr\left[\mathbf{Exp}^{3}_{\mathsf{BS}}(\mathcal{A}_2) \to 1\right]$.

$\mathbf{Exp}^{3}_{\mathsf{BS}}(\mathcal{A}_2)$

1 : $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{BS.KGen}()$
2 : $\mathcal{L} \leftarrow \varnothing$
3 : $(m, \sigma) \leftarrow \mathcal{A}_2^{Sign}(\mathsf{pk})$
4 : **if** $(m, \sigma) \in \mathcal{L}$:
5 : **return** 0
6 : **return** $\mathsf{BS.Vf}(\mathsf{pk}, m, \sigma)$

Oracle $Sign(m)$

1 : $(1, \sigma) \leftarrow \langle \mathsf{BS.Signer}(\mathsf{sk}), \mathsf{BS.User}(\mathsf{pk}, m) \rangle$
2 : $\mathcal{L} \leftarrow \mathcal{L} \cup \{(m, \sigma)\}$
3 : **return** $\sigma$

Note that $\mathbf{Exp}^{3}_{\mathsf{BS}}$ is exactly the experiment $\mathbf{Exp}^{\text{sUF-CMA}}_{\mathsf{BS}}$, therefore $\Pr\left[\mathbf{Exp}^{2}_{\mathsf{BS}}(\mathcal{A}) \to 1\right] \leqslant \mathsf{Adv}^{\text{sUF-CMA}}_{\mathsf{BS}}(\mathcal{B})$ for $\mathcal{B} = \mathcal{A}_2$.

**Part 2.** To finalize the proof construct an adversary $\mathcal{C}$ breaking the blindness property. Introduce the following auxiliary experiment:

$\mathbf{Exp}^{4,b}_{\mathsf{BS}}(\mathcal{C})$

1 : $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{BS.KGen}()$
2 : $b' \xleftarrow{\$} \mathcal{C}^{Init, User_1, User_2}(\mathsf{sk}, \mathsf{pk})$
3 : **return** $b'$

Oracle $Init(m)$

1 : $sess \leftarrow \mathsf{init}$

Oracle $User_1(msg)$

1 : **if** $sess \neq \mathsf{init}$ : **return** $\perp$
2 : $sess \leftarrow \mathsf{open}$
3 : $(msg, state) \leftarrow \mathsf{BS.User}_1((\mathsf{pk}, m), msg)$
4 : **return** $msg$

Oracle $User_2(msg)$

1 : **if** $sess \neq \mathsf{open}$ : **return** $\perp$
2 : $\sigma \leftarrow \mathsf{BS.User}_2(state, msg)$
3 : **if** $(\sigma \neq \perp) \wedge (b = 0)$:
4 : $\quad (1, \sigma) \leftarrow \langle \mathsf{BS.Signer}(\mathsf{sk}), \mathsf{BS.User}(\mathsf{pk}, m) \rangle$
5 : **return** $\sigma$

Here an adversary can make only one query to each oracle (execute only one session). The adversary obtains a signature value generated by the oracles intacting with adversary, if $b = 1$, and a signature computed according to the protocol, otherwise. Note, that if the adversary provokes error in the session, then it always gets $\perp$ from the $User_2$ oracle regardless of bit $b$.

Using a standard technique called «hybrid argument» (see, e.g. [12]) it can be trivially shown, that there exists an adversary $\mathcal{C}'$ such that

$$\Pr\left[\mathbf{Exp}^0_{\mathsf{BS}}(\mathcal{A}) \to 1\right] - \Pr\left[\mathbf{Exp}^1_{\mathsf{BS}}(\mathcal{A}) \to 1\right] = \\ = q \cdot k \cdot \left(\Pr\left[\mathbf{Exp}^{4,1}_{\mathsf{BS}}(\mathcal{C}') \to 1\right] - \Pr\left[\mathbf{Exp}^{4,0}_{\mathsf{BS}}(\mathcal{C}') \to 1\right]\right).$$

Now let construct an adversary $\mathcal{C}$ using $\mathcal{C}'$ as a black box. The adversary $\mathcal{C}$ acts in the following way:

1. The adversary $\mathcal{C}$ obtains $(\mathsf{sk}, \mathsf{pk})$ and transmits this value to $\mathcal{C}'$.
2. When $\mathcal{C}'$ makes a query $m$ to the $Init$ oracle, $\mathcal{C}$ makes a query $(m, m)$ to its own $Init$ oracle.
3. After starting sessions, the adversary $\mathcal{C}$ firstly executes $sess_0$ according to the protocols:
   (a) It computes $(msg^0_{S,1}, state_S) \leftarrow \mathsf{BS.Signer}_1(\mathsf{sk})$ and makes a query $(0, msg^0_{S,1})$ to its own $User_1$ oracle.
   (b) Upon receiving $msg^0_{U,1}$, the adversary $\mathcal{C}$ computes $(msg^0_{S,2}, 1) \leftarrow \mathsf{BS.Signer}_2(state_S, msg_{U,1})$ and makes a query $(0, msg^0_{S,2})$ to its own $User_2$ oracle, receiving the $\varepsilon$ value.

Note that $\sigma_{b_0} \neq \perp$ due to correctness property of the blind signature scheme.

4. Then the adversary $\mathcal{C}$ intercepts all queries of $\mathcal{C}'$ and simply passes their in $sess_1$:

   (a) Intercepting from $\mathcal{C}'$ a query $msg_1$ to the $User_1$ oracle, $\mathcal{C}$ makes a query $(1, msg^1_{S,1})$, where $msg^1_{S,1} = msg_1$, to its own $User_1$ oracle and directly transmits the response $msg^1_{U,1}$ to $\mathcal{C}'$.

   (b) Intercepting from $\mathcal{C}'$ a query $msg_2$ to the $User_2$ oracle, $\mathcal{C}$ makes a query $(1, msg^1_{S,2})$, where $msg^1_{S,2} = msg_2$, to its own $User_2$ oracle. $\mathcal{C}$ receives $(\sigma_0, \sigma_1)$ and returns to $\mathcal{C}'$ the fist component $\sigma_0$. Note that $(\sigma_0, \sigma_1)$ can be $(\perp, \perp)$.

5. $\mathcal{C}$ returns the same bit as $\mathcal{C}'$ returns.

If the $\mathcal{C}$ interacts with the experimentator $\mathbf{Exp}^{\mathrm{wBL},1}_{\mathsf{BS}}$ ($\mathbf{Exp}^{\mathrm{wBL},0}_{\mathsf{BS}}$), then $\sigma_0 = \sigma_{b_1}$ ($\sigma_0 = \sigma_{b_0}$). Moreover, $\mathcal{C}$ returns $\perp$ at stage 4 iff $\mathcal{C}'$ provokes error in $sess_1$ that perfectly coincides with $\mathbf{Exp}^4_{\mathsf{BS}}$. Thus,

$$\Pr\left[\mathbf{Exp}^{4,1}_{\mathsf{BS}}(\mathcal{C}') \to 1\right] = \Pr\left[\mathbf{Exp}^{\mathrm{wBL},1}_{\mathsf{BS}}(\mathcal{C}) \to 1\right],$$

$$\Pr\left[\mathbf{Exp}^{4,0}_{\mathsf{BS}}(\mathcal{C}') \to 1\right] = \Pr\left[\mathbf{Exp}^{\mathrm{wBL},0}_{\mathsf{BS}}(\mathcal{C}) \to 1\right].$$

Summing up,

$$\begin{aligned}
&\Pr\left[\mathbf{Exp}^{0}_{\mathsf{BS}}(\mathcal{A}) \to 1\right] - \Pr\left[\mathbf{Exp}^{1}_{\mathsf{BS}}(\mathcal{A}) \to 1\right] = \\
&\qquad = q \cdot k \cdot \left(\Pr\left[\mathbf{Exp}^{4,1}_{\mathsf{BS}}(\mathcal{C}') \to 1\right] - \Pr\left[\mathbf{Exp}^{4,0}_{\mathsf{BS}}(\mathcal{C}') \to 1\right]\right) = \\
&= q \cdot k \cdot \left(\Pr\left[\mathbf{Exp}^{\mathrm{wBL},1}_{\mathsf{BS}}(\mathcal{C}) \to 1\right] - \Pr\left[\mathbf{Exp}^{\mathrm{wBL},0}_{\mathsf{BS}}(\mathcal{C}) \to 1\right]\right) = q \cdot k \cdot \mathsf{Adv}^{\mathrm{wBL}}_{\mathsf{BS}}(\mathcal{C}).
\end{aligned}$$

□

### 4.2 Robustness/Security against external adversary

The security of the considered signing system against external adversary directly follows from the weak unforgeability of the used blind signature scheme. Weak unforgeability, in its turn, is implied by the strong unforgeability against active adversary. However, there are blind signature schemes that do not provide strong unforgeability but potentially provide the weak one and thus are potentially suitable for providing robustness.

In the current paper we define the particular class of blind signature schemes based on ElGamal signature equation which provide the weak unforgeability. Namely, for such schemes we construct the security reduction to the unforgeability of the base ElGamal signature scheme. Note that all known ElGamal blind signature schemes do not provide strong unforgeability [1].

At first, let us introduce the required notations. We denote the group of points of elliptic curve over the prime field as $\mathbb{G}$, the order of the prime subgroup of $\mathbb{G}$ as $q$, an elliptic curve point of order $q$ as $P$ and zero point as $\mathcal{O}$. We denote by $H$ the hash function that maps binary strings to elements from $\mathbb{Z}_q$ and assume that all field operations are performed modulo $q$.

**ElGamal blind signature scheme.** The generalised ElGamal signature scheme was introduced in [7] and further extended in [6], we denote it by GenEG scheme. A key generation algorithm in this scheme involves picking random $d$ uniformly from $\mathbb{Z}_q^*$ (secret signing key) and defining $Q = dP$ (public verifying key). A signature for message $m$ is a pair $(r, s)$, where $r = (kP).x \bmod q$ for some $k$ picked uniformly at random from $\mathbb{Z}_q^*$ and $s$ is computed from the ElGamal signature equation $EG$:

$$EG(d, k, r, e, s) = 0,$$

where $e = H(m)$. All possible $EG$ equations are listed in [7]. To ensure functionality and security, certain $r, e, s$ values need to be excluded.

ElGamal blind signature scheme, denoted by GenEG-BS, was introduced in [1]. A key generation and verification algorithms in GenEG-BS scheme are the same as in the base GenEG scheme. An interactive signing protocol assumes that the Signer performs ElGamal signature generating algorithm for the $e$ value received from the User, the User algorithm is not determined and can be arbitrary. The parameters of the signing protocol are the base point $P$, public key $Q$ and the message $m$, we denote them by $par$.

In the current paper we impose the additional requirements on the algorithm performed by the User:

- all blinding factors (we denote them by $rnd$) used by the User are selected according to some distribution $\mathcal{D}$ that is independent on the values received from the Signer;
- the first component of the signature $r'$ is the $x$-coordinate of the $R'$ point, which is computed as a result of applying the function parameterized by the $par$ value (we denote it by $\mathcal{L}_1^{par}$) that takes as arguments the $R$ point received from the Signer and $rnd$ values. This function is linear by $R$ for all $rnd$ values generated according to the protocol;
- the second component of the signature $s'$ is computed as a result of applying the function parameterized by the $par$ value (we denote it by $\mathcal{L}_2^{par}$) that takes as arguments the $s$ value received from the Signer, $rnd$ values and point $R$. This function is linear by $s$ for all $rnd$ and $R$ values generated according to the protocol.

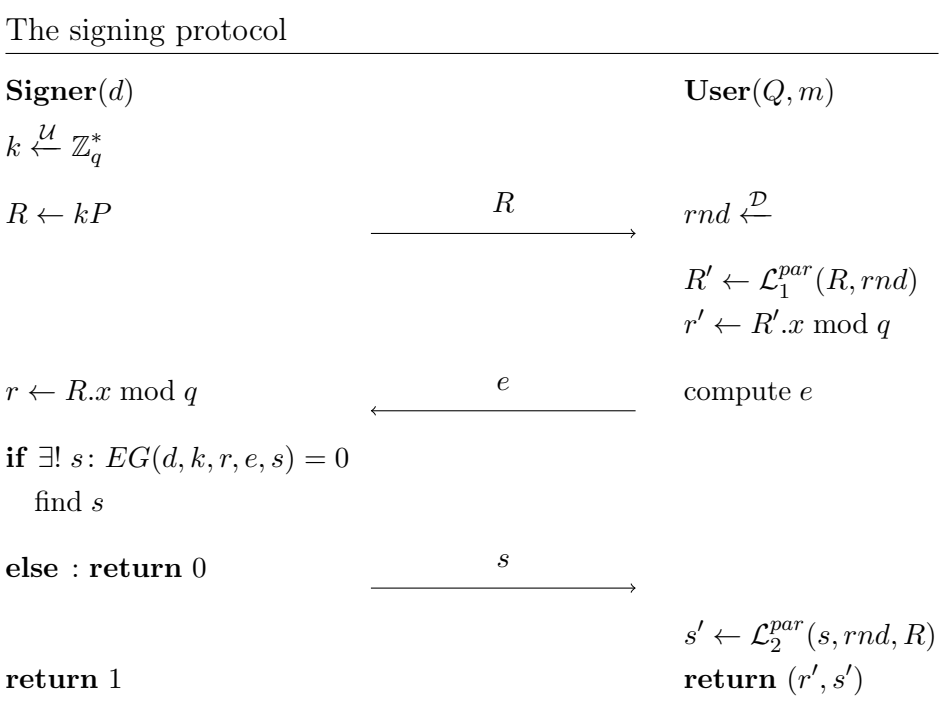



Figure 1: The signing protocol in $\mathsf{GenEG\text{-}BS}_{\mathcal{L}}$ scheme

We denote such scheme by $\mathsf{GenEG\text{-}BS}_{\mathcal{L}}$ scheme. The corresponding signing protocol is illustrated in Figure 1.

Let us show that the $\mathsf{GenEG\text{-}BS}_{\mathcal{L}}$ scheme is indeed the blind version of the GenEG scheme, i.e. provides the same distribution on the signature values. The distribution on GenEG signatures is defined by the uniform distribution on $k$ values. The distribution on $\mathsf{GenEG\text{-}BS}_{\mathcal{L}}$ signatures is defined by the distribution on $k'$ values, where $k'$ is such that $(k'P).x \bmod q = r'$. The $k'$ value is linear by $k$ since $R'$ value is linear by $R$ and $rnd$ values are chosen independently on $R$. Thus, the distribution on $k'$ values is also uniform.

Note that the User view in the $\mathsf{GenEG\text{-}BS}_{\mathcal{L}}$ scheme consists of the incoming messages $R, s$ and the blinding factors $rnd$ sampled by the User.

Now we are ready to construct the security reduction to the unforgeability of the conventional ElGamal signature scheme.

**Theorem 2.** *For any adversary $\mathcal{A}$ for* $\mathsf{GenEG\text{-}BS}_{\mathcal{L}}$ *scheme in the* wUNF *model with time complexity at most $t$ making at most $q$ queries to the signing oracle, there exist an adversary $\mathcal{B}$ for the conventional* GenEG *scheme in the* sUF-CMA *model with the same time complexity at most $t$ making at most $q$ queries to the signing oracle such that*

$$\mathsf{Adv}^{\mathrm{wUNF}}_{\mathsf{GenEG\text{-}BS}_{\mathcal{L}}}(\mathcal{A}) \leqslant \mathsf{Adv}^{\mathrm{sUF\text{-}CMA}}_{\mathsf{GenEG}}(\mathcal{B}).$$

*Proof.* Let construct the adversary $\mathcal{B}$ for the conventional GenEG scheme. The adversary $\mathcal{B}$ uses the adversary $\mathcal{A}$ as a black box. It intercepts the queries of the

adversary $\mathcal{A}$ to the signing oracle and process them by itself using its own signing oracle in the following way.

Receiving the query $m$, adversary $\mathcal{B}$ forwards $m$ to its own oracle and receives the signature $(r', s')$. Then it reconstructs $R'$ point from the verification algorithm and selects $rnd$ value according to the distribution $\mathcal{D}$. After that it calculates the $R$ value using $\mathcal{L}_1^{-1}$ function and $s$ value using $\mathcal{L}_2^{-1}$ function. It returns as an answer the signature $(r', s')$ and the $view = (R, s, rnd)$.

Note that $\mathcal{B}$ generates exactly the same distribution on signature values since $\mathsf{GenEG\text{-}BS}_{\mathcal{L}}$ scheme is the blind version of the $\mathsf{GenEG}$ scheme. The $rnd$ value is chosen as in the honest execution of blind signature protocol, $R$ and $s$ values are also computed as in the honest execution since $\mathcal{L}_1$ and $\mathcal{L}_2$ functions are unambiguously invertible.

When $\mathcal{A}$ returns a forgery, $\mathcal{B}$ translates it to its own challenger and stops. Obviously, if $\mathcal{A}$ wins, then $\mathcal{B}$ wins, whence follows the statement of the theorem. □

**Remark 6.** *The same result may be formulated for the Schnorr signature scheme and its blind version defined in [5]. The proof of the theorem is conducted in the same way.*

## 5 GOST-based blind signature scheme

We propose to use the concrete blind signature scheme in case of building the protection for GOST signature scheme [9]. This scheme was proposed in [4] in 1994 and is commonly refered to as the Camenisch scheme. We provide the definition of this scheme in terms of elliptic group notation.

The key generation algorithm is the same as in the general ElGamal signature scheme and assumes picking secret key $d$ uniformly from $\mathbb{Z}_q^*$ and defining public key $Q$ as $dP$. The signing protocol is defined in Figure 2. The verify procedure for the message $m$ and the signature $(r', s')$ assumes checking $r' \neq 0$ and verifying the equality $r' = R'.x \bmod q$, where $R' = (e')^{-1}(s'P - r'Q)$, $e'$ is equal to $H(m)$, if $H(m) \neq 0$, and to 1 otherwise. Note that the signing protocol in Figure 2 is defined for the case of using the elliptic curves of the prime order. Nevertheless, it can be slightly modified by adding the additional checks for use with non-prime order curves, e.g. with Edwards curves.

This scheme provides perfect blindness (see Theorem 2 [4]), but does not provide unforgeability in the strong sense. In [1] it was shown that it is vulnerable to the ROS-style attack, which is possible if the adversary, acting as a User, is given the opportunity to open $\ell \geqslant \lceil \log q \rceil$ parallel sessions of signing protocol. At the same time, this scheme is potentially secure against active adversary that is allowed to open only sequential sessions of the signing protocol. But even so, pro-

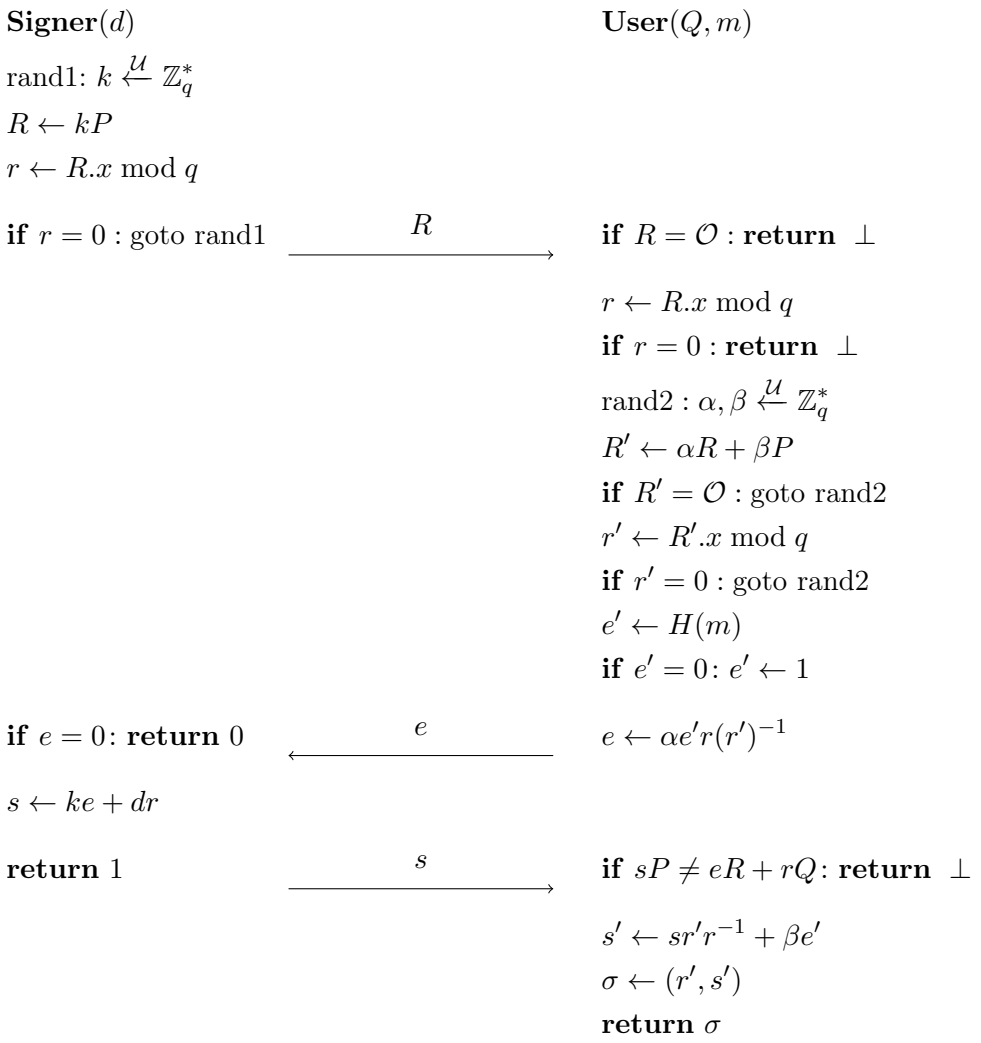



Figure 2: The signing protocol in Camenisch scheme

viding such strong unforgeability is not required for our application, our purpose is the weak unforgeability.

Camenisch scheme is the particular case of the $\mathsf{GenEG\text{-}BS}_{\mathcal{L}}$ scheme defined in Section 4.2. Indeed, the distribution $\mathcal{D}$ in this scheme is a uniform distribution on $\mathbb{Z}_q^* \times \mathbb{Z}_q^*$ that is independent on $R$, $\mathcal{L}_1^{(P,Q,m)}$ and $\mathcal{L}_2^{(P,Q,m)}$ are defined as follows:

$$\mathcal{L}_1^{(P,Q,m)}(R,(\alpha,\beta)) = \alpha R + \beta P; \quad \mathcal{L}_2^{(P,Q,m)}(s,(\alpha,\beta),R) = sr'r^{-1} + \beta e',$$

where $e' = H(m)$, $r = R.x \bmod q$, $r' = (\alpha R + \beta P).x \bmod q$. These functions are linear by $R$ and $s$ values respectively for all possible $rnd$ values. Moreover, zero $r$ and $e$ values are excluded by the corresponding checks on the Signer side as in the GOST signature scheme. Therefore, the result of Theorem 2 is applied to the Camenisch scheme, which means that it provides weak unforgeability under the assumption that GOST scheme provides unforgeability. The security in the sUF-CMA model, in its turn, directly follows from the weak unforgeability.

Thus, the Camenisch scheme is a blind version of the GOST scheme and can be applied in the systems realizing the GOST signature as the protection against

backdoors in smart-cards. It provides the security against external adversary and adversary with agent only by the security of the GOST signature scheme. Note that such solution in contrast to the solution from [2] does not need any additional assumptions about the smart-card such as correct implementation of low-level arithmetic operations and the absence of failures. Moreover, it requires less computations on the smart-card side.

# 6 Conclusion

The paper addressed the security issues that arise in signing systems when the smart-card being used for key storage and signing is believed to contain backdoors. A novel approach based on blind signature schemes to protect against backdoors has been proposed. It has been proven that weak versions of standard security properties (blindness and unforgeability) of blind signature scheme imply security against backdoors in smart-cards.

Moreover, the concrete solution in case of using the GOST signature scheme has been proposed. This solution is the well known Camenisch's blind signature scheme that provides perfect blindness. It was shown that the target security is held under the sole assumption that the GOST signature scheme provides standard security, i.e. is unforgeable under chosen message attack.

# References

[1] Akhmetzyanova L., Alekseev E., Babueva A., Smyshlyaev S., "On the (im)possibility of ElGamal blind signatures", Cryptology ePrint Archive, paper 2022/1128, 2022.

[2] Akhmetzyanova L., Alekseev E., Bozhko A., Smyshlyaev S., "Secure implementation of digital signature using semi-trusted computational core", *Mathematical Aspects of Cryptography*, **12**:4 (2021), 5–23.

[3] Wang Y., "Password Protected Smart Card and Memory Stick Authentication against Off-Line Dictionary Attacks.", *IFIP Advances in Information and Communication Technology*, **376**, ed. Gritzalis D., Furnell S., Theoharidou M., Springer, Berlin, Heidelberg, 2012.

[4] Camenisch, J. L., Piveteau, J. M., Stadler, M. A., "Blind signatures based on the discrete logarithm problem", *LNCS*, Advances in Cryptology — EUROCRYPT'94, **950**, ed. De Santis A, Springer, Berlin, Heidelberg, 1994.

[5] Chaum D., "Blind signatures for untraceable payments", *Advances in cryptology*, ed. Chaum D., Rivest R.L., Sherman A.T., Springer, Boston, MA, 1983, 199–203.

[6] Fersch M., *The provable security of Elgamal-type signature schemes*, Diss. Bochum, Ruhr-Universität Bochum, 2018.

[7] Harn, L., Xu, Y., "Design of generalised ElGamal type digital signature schemes based on discrete logarithm", *Electronics Letters*, **30**:24 (1994), 2025–2026.

[8] Akhmetzyanova L., Alekseev E., Oshkin I., Smyshlyaev S., "A review of the password authenticated key exchange protocols vulnerabilities and principles of the SESPAKE protocol construction", *Mathematical Aspects of Cryptography*, **7**, 2016, 7-28.

[9] *GOST R 34.10-2012. Information technology. Cryptographic data security. Signature and verification processes of electronic digital signature. National standard of the Russian Federation, STANDARTINFORM*, 2012, In Russian.

[10] Bellare M., Rogaway P., "The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs", *LNCS*, Advances in Cryptology - EUROCRYPT 2006, **4004**, ed. Vaudenay S., Springer, Berlin, Heidelberg, 2006.

[11] Fuchsbauer G., Plouviez A., Seurin Y., "Blind Schnorr signatures and signed ElGamal encryption in the algebraic group model", *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Springer, Cham, 2020, 63–95.

[12] Fischlin, M. Mittelbach, A., "An Overview of the Hybrid Argument", *Cryptology ePrint Archive, Paper 2021/088*, 2021.

# Two-party GOST in two parts: fruitless search and fruitful synthesis

Liliya Akhmetzyanova, Evgeny Alekseev, Alexandra Babueva, Lidiia Nikiforova, and Stanislav Smyshlyaev

CryptoPro LLC, Moscow, Russia
{lah, alekseev, babueva, nikiforova, svs}@cryptopro.ru

**Abstract**

In the current paper we investigate the possibility of designing secure two-party signature scheme with the same verification algorithm as in the Russian standardized scheme (GOST scheme). We solve this problem in two parts. The first part is a (fruitless) search for an appropriate scheme in the literature. It turned out that all existing schemes are insecure in the strong security models. The second part is a synthesis of new signature scheme and ends fruitfully. We synthesize a new two-party GOST signature scheme, additionally using the commitment scheme, guided by the features of the GOST signature scheme, as well as the known attacks on existing schemes. We prove that this scheme is secure in a bijective random oracle model in the case when one of the parties is malicious under the assumption that the classical GOST scheme is unforgeable in a bijective random oracle model and the commitment scheme is modelled as a random oracle.

**Keywords:** two-party signature, GOST signature.

## 1 Introduction

Electronic document management systems become a common daily occurrence in the modern world. Signature scheme is a fundamental component of these systems. The systems involve the client, who owns a private signing key, and the server, who manages the documents. The server sends the document to the client, who checks it and signs. It is highly desirable to implement the client side at the user mobile device to make the information system as user-friendly as possible. There is a problem of secure storage of the private key on a mobile device, since it is easy to gain physical access to the device, for example, as a result of theft. If the adversary gets access to the private key, it will be able to sign documents on behalf of the user. So, we need a way to protect the private key stored on the mobile device.

One method of protection is to use the so-called two-party signature scheme instead of the classical signature scheme. This method involves the private key sharing between the client and the server and generating the signature as a result of an interactive protocol run between them. We assume that no trusted party is involved in this process. Such protocol should not allow either party to create a signature without interacting with the other party. In particular, the server can not sign any document without the owner of the signing key. At the same time in case of theft of the user's device, the adversary gets access to only one part of the key and needs to interact with the server to create a signature. Note that the server can notify the user about each execution of the protocol via an outside channel, for example, by e-mail. The user whose mobile device has been stolen can report this to the server and forbid the possibility of creating a signature.

This method of protection should remain completely transparent to all external systems that can potentially use the generated signature. That is, it should not differ from the classic signature generated when the key is fully stored on the user's device. This means that the verification algorithm should be the same as in the classical scheme. We use the Russian signature scheme defined in [7, 8, 9, 10] (hereinafter — GOST scheme) as a classical signature scheme. Thus, to implement the described method of protecting the private key, we need a two-party signature scheme with the same verification algorithm as in the GOST scheme.

In literature there are a number of schemes [15, 13, 12, 1, 19] based on GOST signature equation, in which the signature is generated by several signers. It is not only two-party schemes, but also schemes for more participants: collective signature schemes (for $n$ parties) and threshold signature schemes where any subset of at least $t$ out of the $n$ parties can produce a valid signature. Note that such schemes are the extensions of two-party schemes, therefore they can also be used for private key protection.

However, it turned out that all existing schemes are not suitable for solving our problem. Some of them are proven secure in the weak security models while others are vulnerable to the attacks. Let's consider these schemes. The threshold signature scheme proposed in [1] uses a third trusted party to form the signature. The signature scheme proposed in [12] uses a new secret sharing algorithm for key generation and signing protocols and is proven secure only against passive adversary. The scheme proposed in [19] appeared to be insecure if the adversary is given the opportunity to open parallel sessions of the signing protocol. In this paper, we build a ROS-style attack [2] on this

scheme (see Appendix A.1). In [15, 13] there is no description of the distributed key generation protocol for the proposed scheme. It is not clear how to implement it if no third trusted party is involved. Moreover, the signing protocol of this scheme is vulnerable to a ROS attack.

We synthesize a new two-party GOST signature scheme, additionally using the commitment scheme, guided by the features of the GOST signature scheme, as well as the known attacks on existing schemes. Our scheme does not use any non-standard cryptographic mechanisms such as homomorphic encryption. Section 3 presents the design rationale of this scheme. A formal description of the scheme is provided in the Section 4. We prove that this scheme is secure in a bijective random oracle model in the case when one of the parties is malicious under the assumption that the classical GOST scheme is unforgeable in a bijective random oracle model and the commitment scheme is modelled as a random oracle. Our proof is based on the security proof of the GOST signature scheme presented in [4] and the security proof for the two-party Schnorr signature scheme [16]. A description of the security model and the main result are presented in the Section 5.

## 2 Basic notations and definitions

By $\{0,1\}^*$ we denote the set of all bit strings of finite length including the empty string. If $p$ is a prime number then the set $\mathbb{Z}_p$ is a finite field of size $p$. We assume the canonic representation of the elements in $\mathbb{Z}_p$ as integers in the interval $[0 \dots p-1]$. Each non-zero element $x$ in $\mathbb{Z}_p$ has an inverse $1/x$. We define $\mathbb{Z}_p^*$ as the set $\mathbb{Z}_p$ without zero element.

We denote the group of points of elliptic curve over the field $\mathbb{Z}_p$ as $\mathbb{G}$, the order of the prime subgroup of $\mathbb{G}$ as $q$ and elliptic curve point of order $q$ as $P$. We denote the x-coordinate of the point $R \in \mathbb{G}$ as $R.x$. We denote by H the hash function that maps binary strings of arbitrary length to the binary string of length $h$.

If the value $s$ is chosen from a set $S$ uniformly at random, then we denote $s \xleftarrow{\mathcal{U}} S$. If the variable $x$ gets the value $val$ then we denote $x \leftarrow val$. Similarly, if the variable $x$ gets the value of the variable $y$ then we denote $x \leftarrow y$. If the variable $x$ gets the result of an algorithm $A$ we denote $x \leftarrow A$.

The signature scheme $\mathsf{SS}$ is determined by three algorithms:

– $(d, Q) \leftarrow \mathsf{KGen}()$: a probabilistic key generation algorithm that returns the signature key pair $(d, Q)$, where $d$ is a private signing key, $Q$ is a public verifying key.

– $\sigma \leftarrow \mathsf{Sign}(d, m)$: a probabilistic signing algorithm that takes a signing key $d$ and a message $m$ as an input and outputs a signature $\sigma$ for the message $m$.

– $b \leftarrow \mathsf{Verify}(Q, m, \sigma)$: a (deterministic) verification algorithm that takes a public verifying key $Q$, a message $m$ and a signature $\sigma$ as an input and outputs 1 if $\sigma$ is valid and 0 otherwise.

The two-party signature scheme 2p-SS is determined by three algorithms:

– $((d_1, Q), (d_2, Q)) \leftarrow \mathsf{KGen}\langle \mathbf{P_1}(), \mathbf{P_2}() \rangle$: an interactive key generation protocol that is run between a party $\mathbf{P_1}$ and a party $\mathbf{P_2}$; for $i \in \{1, 2\}$ $\mathbf{P_i}$ outputs it's private key $d_i$ and a public verifying key $Q$.

– $(\sigma, \sigma) \leftarrow \mathsf{Sign}\langle \mathbf{P_1}(d_1, Q, m), \mathbf{P_2}(d_2, Q, m) \rangle$: an interactive signing protocol that is run between a party $\mathbf{P_1}$ and a party $\mathbf{P_2}$; for $i \in \{1, 2\}$ $\mathbf{P_i}$ takes it's private key $d_i$, a public verifying key $Q$ and a message $m$ as an input and outputs a signature $\sigma$ for the message $m$ if the interaction completes successfully and $\perp$ otherwise.

– $b \leftarrow \mathsf{Verify}(Q, m, \sigma)$: a (deterministic) verification algorithm that takes a public verifying key $Q$, a message $m$ and a signature $\sigma$ as an input and outputs 1 if $\sigma$ is valid and 0 otherwise.

The commitment scheme is determined by two algorithms:

– $(op, comm) \leftarrow \mathsf{Cmt}(m)$: a commitment algorithm that takes message $m \in \{0,1\}^*$ as an input and outputs a commitment $comm \in \{0,1\}^n$ and an opening value $op \in \{0,1\}^\kappa$.

– $b \leftarrow \mathsf{Open}(comm, m, op)$: a (deterministic) opening algorithm that takes a commitment $comm \in \{0,1\}^n$, a message $m \in \{0,1\}^*$ and an opening value $op \in \{0,1\}^\kappa$ and outputs 1 if $(op, comm)$ is valid on $m$ and 0 otherwise.

## 3 Design rationale

The GOST signature is a pair $(r, s)$:

$$s = ke + dr,\ r = R.x \bmod q = (kP).x \bmod q,$$

where $k$ is selected uniformly from $\mathbb{Z}_q^*$, $d \in \mathbb{Z}_q^*$ is a private key, $e$ is the hash of the message $m$. The secret parameters are the long-term signing key $d$ and the ephemeral value $k$.

Two-party signature scheme implies that both parties contribute to the generation of all secret parameters. Note that the signature equation is linear with respect to secret parameters $d$ and $k$. Thus, the straightforward way is to use additive secret sharing of these parameters: $k = k_1 + k_2$, $d = d_1 + d_2$. Then the signature $(r, s)$ is formed as:

$$s = (k_1 + k_2)e + (d_1 + d_2)r = (k_1 e + d_1 r) + (k_2 e + d_2 r),$$

$$r = (R_1 + R_2).x \mod q = (k_1 P + k_2 P).x \mod q.$$

Consider the naive version of the two-party GOST signature scheme between participants $\mathbf{P_1}$ and $\mathbf{P_2}$. The key generation protocol KGen and the signing protocol Sign are shown at Figure 1 and Figure 2 respectively.

**Key generation protocol.** At first, let's consider the KGen protocol. We claim that this key generation algorithm is not secure. Indeed, the party $\mathbf{P_2}$ can choose $Q_2$ depending on $Q_1$ in such a way that it will know the discrete logarithm of the final public key $Q$, i.e. the private key $d$. For example, the party $\mathbf{P_2}$ can set $Q_2 = P - Q_1$. Then, $Q = Q_1 + Q_2 = P$, $d = 1$.

One way to protect against this attack is to use a commitment scheme just like in the two-party Schnorr signature scheme [16]. Instead of sending $Q_1$, the party $\mathbf{P_1}$ can send the commitment to the value $Q_1$. Then party $\mathbf{P_2}$ does not know any information about $Q_1$ due to the «hiding» property of the commitment scheme and generates $Q_2$ independently of $Q_1$. The party $\mathbf{P_1}$ cannot change $Q_1$ after it receives $Q_2$ from the party $\mathbf{P_2}$ due to the «binding» property of the commitment scheme.

Another way of protection is to use the multiplicative method of the private key $d$ sharing as in the scheme proposed in [19]. The simple version of the algorithm is shown at Figure 3. The party $\mathbf{P_2}$ can also set the $Q_2$ value depending on $Q_1$. However in such case it only knows how $d$ depends on $d_1$, but does not know the $d$ value itself because of unpredictability of $d_1$. For example, the party $\mathbf{P_2}$ can set $Q_2 = Q_1$. Then, the party $\mathbf{P_1}$ calculates $Q = d_1 \cdot Q_1 = d_1^2 \cdot P$.

Note that in case of the multiplicative key sharing there is no obvious way how to further use the key shares to create a signature. Specifically, participants must calculate the value $d_1 \cdot d_2 \cdot r$ without revealing the secret to the other party. The authors of the paper [19] use the additively homomorphic encryption scheme based on factoring problem for such calculation.

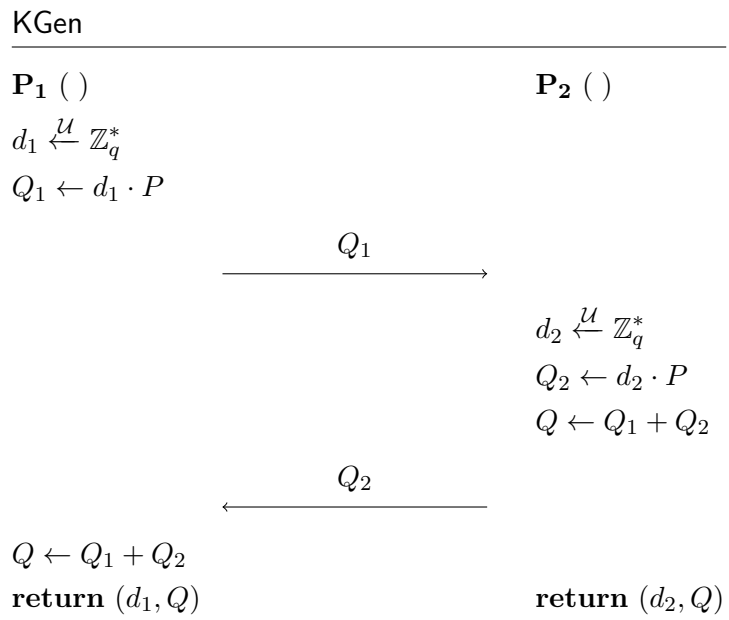



Figure 1: The key generation protocol of the naive version of the two-party GOST

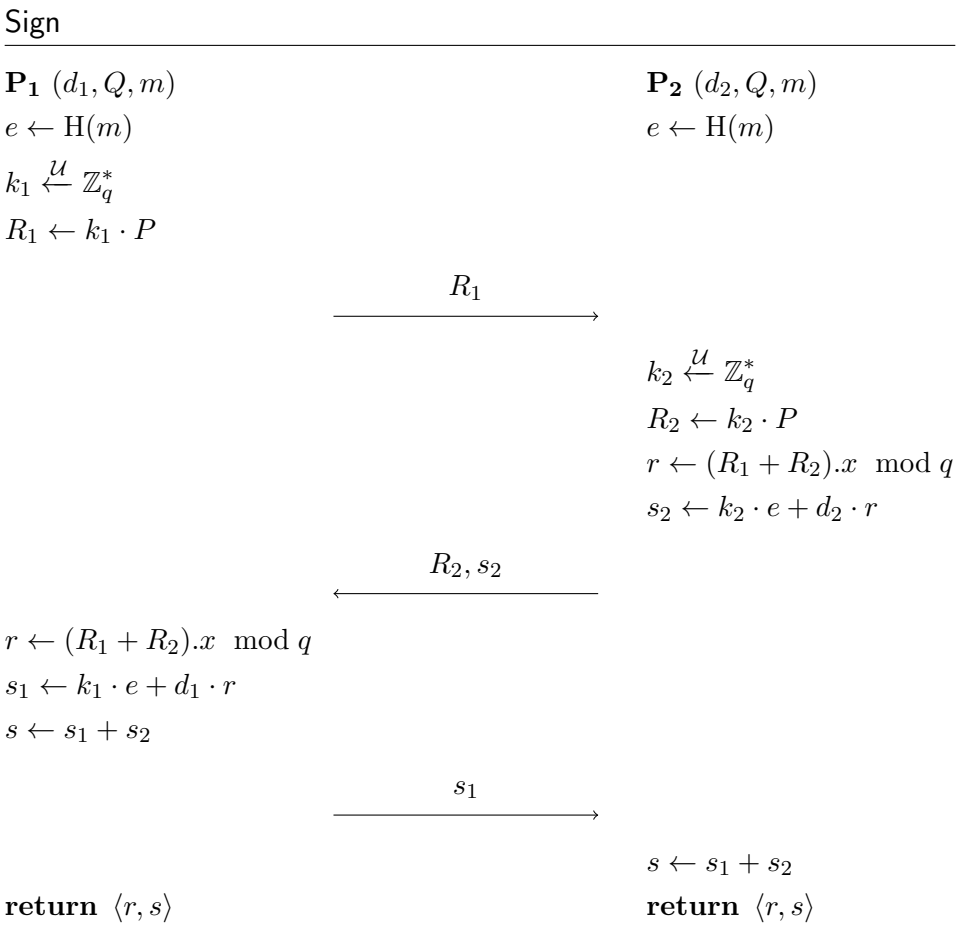



Figure 2: The signing protocol of the naive version of the two-party GOST

Since we strive not to use non-standard cryptographic mechanisms, we decide to use the additive secret sharing with the commitment scheme.

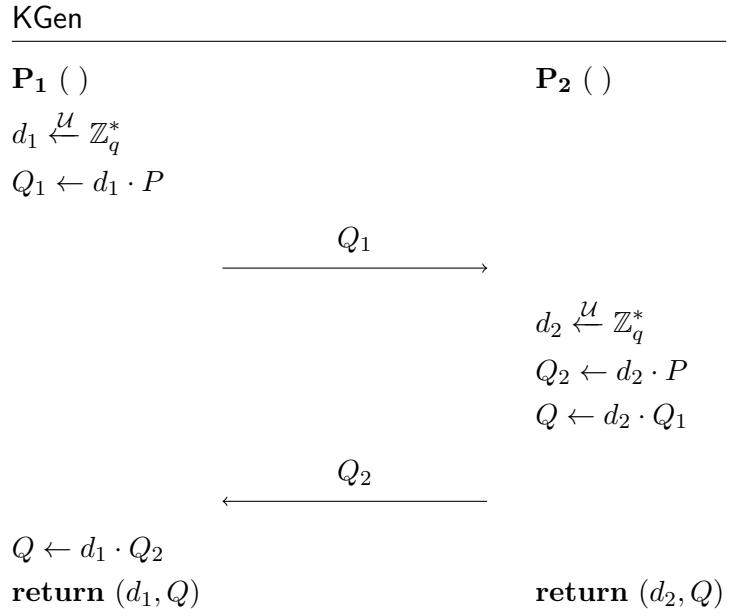



Figure 3: The KGen protocol of the scheme from [19]

**Signing protocol.** Let's consider the Sign protocol from Figure 2 which is the same as in the schemes proposed in [15, 13]. We claim that it is not secure, since it is vulnerable to the ROS-style attack.

The original ROS attack was proposed in [2], the authors show that it is applicable to some threshold signature schemes [6, 11]. The attack works if the one party is given the opportunity to open $l \geqslant \lceil \log q \rceil$ parallel sessions of signing protocol with the other party. Let's discuss the features of signing protocol from Figure 2 that make the attack applicable. The main observation is that one party can select its parameters when it knows the parameters selected by the other party. Indeed, the party $\mathbf{P_2}$ can open $l$ parallel sessions with $\mathbf{P_1}$, receive $l$ points $R_1^1, \ldots, R_1^l$ and construct the corresponding $R_2$ points in some specific way dependent on $R_1$ values. Note that in the scheme from [19] the party $\mathbf{P_2}$ also can vary $R_2$ after receiving $R_1$ from the party $\mathbf{P_1}$. We provide an explicit description of ROS-style attack on this scheme in Appendix A.1 and some modification of this attack for scheme defined at Figure 2 in Appendix A.2.

We use the commitment scheme to protect against this attack. Instead of sending $R_1$, the party $\mathbf{P_1}$ can send the commitment to the value $R_1$. Then party $\mathbf{P_2}$ does not know any information about $R_1$ due to the «hiding» property of the commitment scheme and generates $R_2$ independently of $R_1$.

The party $\mathbf{P_1}$ cannot change $R_1$ after it receives $R_2$ from the party $\mathbf{P_2}$ due to the «binding» property of the commitment scheme. Consequently, each party cannot vary any parameters after receiving the parameters selected by the other party.

Note that up to this point we have assumed that the message initially exists on both sides, i.e. given them as an input. In practice, one of the parties usually forwards the message to the other. It is important that each party captures the message before it learns the parameters of the other party to protect against the ROS-style attack. We provide the description of the attack on the modification of discussed signing protocol in which the party $\mathbf{P_1}$ selects message $m$ and sends it to the party $\mathbf{P_2}$ after receiving $R_2$ in Appendix A.3.

# 4 Two-party GOST

In this section we describe the two-party signature scheme 2p-GOST. It is based on the GOST signature scheme.

The key generation protocol KGen and the signing protocol Sign use a commitment scheme. The party $\mathbf{P_1}$ computes the Cmt function for commitment generation, the party $\mathbf{P_2}$ computes the Open function for commitment verification during the protocols execution.

Note that the HMAC [18] can be used as a commitment. Then for $m \in \{0,1\}^*$ the commitment scheme is defined at Figure 4.

| $\mathsf{Cmt}(m)$ | $\mathsf{Open}(comm, m, op)$ |
|---|---|
| 1 : $op \xleftarrow{\mathcal{U}} \{0,1\}^\kappa$ | 1 : $comm' \leftarrow \mathsf{HMAC}(op, m)$ |
| 2 : $comm \leftarrow \mathsf{HMAC}(op, m)$ | 2 : **if** $(comm' \neq comm)$ : **return** 0 |
| 3 : **return** $(op, comm)$ | 3 : **return** 1 |

Figure 4: HMAC as a commitment.

**Key generation protocol.** The party $\mathbf{P_1}$ and party $\mathbf{P_2}$ execute the KGen protocol. As a result of executing, a new key pair $(d, Q)$ for the GOST scheme is implicitly formed. But the signing key $d$ does not appear on either side. The output of the party $\mathbf{P_1}$ is a private key share $d_1$ and signature verification key $Q$. The output of the party $\mathbf{P_2}$ is a private key share $d_2$ and signature verification key $Q$. Note that $d = d_1 + d_2$.

A detailed description of the protocol is presented at Figure 5.

KGen

| $\mathbf{P_1}$ ( ) | | $\mathbf{P_2}$ ( ) |
|---|---|---|
| $d_1 \xleftarrow{\mathcal{U}} \mathbb{Z}_q^*$ | | |
| $Q_1 \leftarrow d_1 \cdot P$ | | |
| $(op_Q, comm_Q) \leftarrow \mathsf{Cmt}(Q_1)$ | | |
| | $\xrightarrow{comm_Q}$ | |
| | | $d_2 \xleftarrow{\mathcal{U}} \mathbb{Z}_q^*$ |
| | | $Q_2 \leftarrow d_2 \cdot P$ |
| | $\xleftarrow{Q_2}$ | |
| **if** $(Q_2 = -Q_1)$ : **return** $\perp$ | | |
| $Q \leftarrow Q_1 + Q_2$ | | |
| | $\xrightarrow{op_Q, Q_1}$ | |
| | | **if** $(\mathsf{Open}(comm_Q, Q_1, op_Q) = 0)$ : **return** $\perp$ |
| | | **if** $(Q_1 = -Q_2)$ : **return** $\perp$ |
| | | $Q \leftarrow Q_1 + Q_2$ |
| **return** $(d_1, Q)$ | | **return** $(d_2, Q)$ |

Figure 5: Key generation protocol of the 2p-GOST signature scheme.

**Verification algorithm.** This algorithm can be executed by anyone with the use of verification key $Q$ and is the same as in the GOST signature scheme. A detailed description of the algorithm is presented at Figure 6.

$\mathsf{Verify}(Q, m, \langle r, s \rangle)$

1 : **if** $(s = 0 \vee r = 0)$ : **return** 0
2 : $e \leftarrow \mathrm{H}(m)$
3 : **if** $e = 0 : e \leftarrow 1$
4 : $R \leftarrow e^{-1}sP - e^{-1}rQ$
5 : **if** $(R.x \bmod q \neq r)$ : **return** 0
6 : **return** 1

Figure 6: Verification algorithm of the 2p-GOST signature scheme.

**Signing protocol.** The party $\mathbf{P_1}$ and party $\mathbf{P_2}$ execute this protocol. Party $\mathbf{P_1}$ takes $d_1, Q$ generated as a result of KGen and the message $m$ as an input. Party $\mathbf{P_2}$ takes $d_2, Q$ generated as a result of KGen and the message $m$ as an

input. As a result of executing, each of the parties receives a signature $\sigma$ for the message $m$ corresponding to the signature verification key $Q$.

A detailed description of the protocol is presented at Figure 7.

Sign

| $\mathbf{P_1}$ $(d_1, Q, m)$ | | $\mathbf{P_2}$ $(d_2, Q, m)$ |
|---|---|---|
| $k_1 \xleftarrow{\mathcal{U}} \mathbb{Z}_q^*$ | | |
| $R_1 \leftarrow k_1 \cdot P$ | | |
| $(op_R, comm_R) \leftarrow \mathsf{Cmt}(R_1)$ | | |
| | $\xrightarrow{comm_R}$ | |
| | | $k_2 \xleftarrow{\mathcal{U}} \mathbb{Z}_q^*$ |
| | | $R_2 \leftarrow k_2 \cdot P$ |
| | $\xleftarrow{R_2}$ | |
| **if** $(R_2 = -R_1)$ : **return** $\perp$ | | |
| $r \leftarrow (R_1 + R_2).x \mod q$ | | |
| **if** $(r = 0)$ : **return** $\perp$ | | |
| $e \leftarrow \mathrm{H}(m)$ | | |
| **if** $(e = 0) : e \leftarrow 1$ | | |
| $s_1 \leftarrow k_1 \cdot e + d_1 \cdot r$ | | |
| | $\xrightarrow{op_R, R_1, s_1}$ | |
| | | **if** $(\mathsf{Open}(comm_R, R_1, op_R) = 0)$ : **return** $\perp$ |
| | | **if** $(R_1 = -R_2)$ : **return** $\perp$ |
| | | $e \leftarrow \mathrm{H}(m)$ |
| | | **if** $(e = 0) : e \leftarrow 1$ |
| | | $r \leftarrow (R_1 + R_2).x \mod q$ |
| | | $s_2 \leftarrow k_2 \cdot e + d_2 \cdot r$ |
| | | $s \leftarrow s_1 + s_2$ |
| | | $\sigma \leftarrow (r, s)$ |
| | | **if** $(\mathsf{Verify}(Q, m, \sigma) = 0)$ : **return** $\perp$ |
| | $\xleftarrow{s_2}$ | |
| $s \leftarrow s_1 + s_2$ | | |
| $\sigma \leftarrow (r, s)$ | | |
| **if** $(\mathsf{Verify}(Q, m, \sigma) = 0)$ : **return** $\perp$ | | |
| **return** $\sigma$ | | **return** $\sigma$ |

Figure 7: Signing protocol of the 2p-GOST signature scheme.

Note that GOST signature algorithm checks non-equality of $r$ and $s$ values to zero. The Sign protocol does not contain an explicit check of $s$ value being zero, since parties execute the Verify algorithm at the same round at which they calculate $s$, and the Verify algorithm contains this check. By the same reason, the party $\mathbf{P_2}$ does not check equality of $r$ to the zero.

# 5 Security notions and bounds

We introduce sOMUF-PCA notion (strong One More Unforgeability under Party Compromised Attack) to analyze the security of the 2p-GOST scheme. It is a natural commonly used model [16], [14] implying that the adversary acts as one of the parties.

We prove the security under some idealized assumptions:

– we model commitment scheme as a random oracle; the commitment for KGen protocol is modeled as oracle $qRO$; the commitment for Sign protocol is modeled as oracle $rRO$. The random oracle [3] is an ideal primitive which models a random function via oracle. It provides a random output for each new query, identical input queries are given the same answer;

– we model conversion function $r = f(R)$ in the GOST signature scheme using the bijective random oracle (see detailes below). The bijective random oracle [5] is an idealized public bijection that is accessible, in both directions, via oracles.

The security is reduced to the security of the GOST scheme regarding the sUF-KO (strong Unforgeability under Key Only attack) notion and the signum-relative collision resistant property of the used hash function family.

Before proceeding to the formulation of the result, let's define the considered target security model, the signum-relative collision resistant property, the sUF-KO notion and the bijective random oracle.

**sOMUF-PCA notion.** Let's describe the sOMUF-PCA notion informally. The adversary $\mathcal{A}$ compromises one of the parties and communicates with the other party in the 2p-SS signature scheme. At the beginning, it can execute the KGen protocol once by querying a $KGen$ oracle. This oracle models the actions of the honest (uncompromised) party. After executing the KGen protocol adversary can execute the Sign protocol. Meanwhile, the adversary can open the parallel sessions of this protocol. For these, the adversary can

make queries to the *NewSign* oracle for opening the session and then to the *Sign* oracle for execution the signing protocol. *Sign* oracle models the actions of the honest party. The adversary has the capability not to finish the sessions and provoke the failures on the honest party side. The adversary's goal is to make $l + 1$ correct (message, signature) pairs after $l$ successful interactions with the honest party. The probability of achieving the goal by the adversary $\mathcal{A}$ is denoted by $\mathsf{Adv}^{\mathsf{sOMUF\text{-}PCA}}_{\mathsf{2p\text{-}GOST}}(\mathcal{A})$.

Note that such way to formulate the threat via one-more forgery captures the intuition that it is impossible to create a forgery without interacting with an honest party. It was introduced for defining unforgeability of blind signature schemes [17]. The classical way to define unforgeability for standard signature scheme is to make only one forgery that is correct and non-trivial, i.e. was not obtained as a result of honest execution of the protocol. However, in case of two-party schemes some problems may occur while defining non-triviality. Indeed, as soon as the proposed model allows the adversary not to finish the sessions, the following situation is possible. The adversary acting as $\mathbf{P_2}$ computes the signature value and does not send it to the honest party at the last flow of the signing protocol. In this case the honest party could not determine whether the signature, returned as a forgery, is indeed fresh or was generated in the unfinished session. To address this problem we use one-more setting and consider the interaction successful if the honest party completes the computation of it's part of the signature and sends it to the adversary.

The formal description of the sOMUF-PCA notion is given in Appendix B.

**Signum-relative collision resistant property.** This property for a hash function family means that it is difficult to find two different messages $m_1$, $m_2$ such that the hash function values from these messages match up to the sign.

Throughout the paper we consider implicitly keyed hash functions H: $\{0,1\}^* \mapsto \{0,1\}^h$ with initialization vector assumed to be an implicit key. The experiments of the up-coming security definitions should be understood as implicitly first picking a random initialization vector $IV \in \mathcal{IV}$ and giving it to the adversary.

**Definition 1** (SCR property)**.** *For the family of hash functions* H

$$\mathsf{Adv}^{\mathsf{SCR}}_{\mathsf{H}}(\mathcal{A}) = \Pr\left[(m_1, m_2) \xleftarrow{\$} \mathcal{A} : \mathrm{H}(m_1) = \pm\mathrm{H}(m_2) \wedge m_1 \neq m_2\right]$$

**sUF-KO notion.** Consider the sUF-KO (strong Unforgeability under Key Only attack) notion for the signature scheme SS. The adversary $\mathcal{A}$ receives signa-

ture verification key $Q$. It's goal is to make a forgery.

**Definition 2.** *For a signature scheme* $\mathsf{SS}$

$$\mathsf{Adv}_{\mathsf{SS}}^{\mathsf{sUF\text{-}KO}}(\mathcal{A}) = \Pr\left[\mathbf{Exp}_{\mathsf{SS}}^{\mathsf{sUF\text{-}KO}}(\mathcal{A}) \to 1\right],$$

*where the experiment* $\mathbf{Exp}_{\mathsf{SS}}^{\mathsf{sUF\text{-}KO}}(\mathcal{A})$ *is defined in the following way:*

$\mathbf{Exp}_{\mathsf{SS}}^{\mathsf{sUF\text{-}KO}}(\mathcal{A})$

1: $(d, Q) \leftarrow \mathsf{SS.KGen}(\,)$
2: $(m, \sigma) \xleftarrow{\$} \mathcal{A}(Q)$
3: $res \leftarrow \mathsf{SS.Verify}(Q, m, \sigma)$
4: **return** $res$

**Bijective random oracle.** Bijective random oracle model (BRO model) was proposed in [5] to achieve provable security for signature schemes based on the ElGamal signature equation. In particular, $\mathsf{GOST}$ scheme is proven secure in the BRO model [4] (under some assumptions on the used primitives).

Bijective random oracle is used to model the mapping from group elements to the space $\mathbb{Z}_q$ used in $\mathsf{GOST}$ signature: $r = f(R) = R.x \bmod q$. We decompose the conversion function $f$ as follows:

$$f = \psi \circ \Pi \circ \phi,$$

where $\Pi$ is a bijection. The idea is to reflect in $\phi$ the structure of $f$ that involves only its domain and to reflect in $\psi$ the structure that involves only its range; the component that is responsible for disrupting any algebraic link between the domain and the range is modeled by $\Pi$. In security proofs we will replace $\Pi$ by a bijective random oracle.

For the $\mathsf{2p\text{-}GOST}$ and $\mathsf{GOST}$ signature schemes:

- $\phi : \mathbb{G} \to \{0,1\}^N, N = \lceil \log_2 p \rceil$, is deterministic encoding function that is implemented as the mapping the point with coordinates $(x, y)$ to the bit representation of the $x$-coordinate. This is semi-injection function, i.e. it is injective except for the mutually inverse elements $A, B$ for which the equality $\phi(A) = \phi(B)$ holds;
- $\psi : \{0, 1, \ldots, 2^N - 1\} \to \mathbb{Z}_q$ is a function that maps integer to elements of $\mathbb{Z}_q$ that is implemented as the reduction of an integer modulo q;
- $\Pi : \{0,1\}^N \to \{0, 1, \ldots, 2^N - 1\}$ is the link in the middle, bridging the range of $\phi$ with the domain of $\psi$.

Finally, we are ready to formulate the security bound for the 2p-GOST scheme.

**Theorem 1.** *Let $\mathcal{A}$ be an adversary with time complexity $T$ in the* sOMUF-PCA *model for the* 2p-GOST *scheme, making at most $q_R$ and $q_Q$ queries to the random oracles $rRO$ and $qRO$ respectively, at most $q_{BRO}$ and $q_{BRO^{-1}}$ queries to the bijective random oracles $BRO$ and $BRO^{-1}$ respectively and at most $q_{sign}$ queries to the oracle $NewSign$. Then, there exist an adversary $\mathcal{B}$ in the* sUF-KO *model for the* GOST *scheme and an adversary $\mathcal{C}$ that breaks the signum-relative collision resistant property of* H*, such that:*

$$\mathsf{Adv}^{\mathsf{sOMUF\text{-}PCA}}_{\mathsf{2p\text{-}GOST}}(\mathcal{A}) \leqslant \mathsf{Adv}^{\mathsf{sUF\text{-}KO}}_{\mathsf{GOST}}(\mathcal{B}) + \mathsf{Adv}^{\mathsf{SCR}}_{\mathsf{H}}(\mathcal{C}) + \\ + \frac{q_Q + q_{sign} \cdot (q_R + q_{sign})}{2^{\min\{\kappa, n\}}} + \frac{2(q_{BRO} + q_{BRO^{-1}} + 3q_{sign} + 1)^2}{q},$$

*where $\kappa, n$ are the parameters of the underlying commitment scheme.*

*An adversary $\mathcal{B}$ makes at most $(q_{BRO} + 2q_{sign} + 1)$ and $q_{BRO^{-1}}$ queries to the bijective random oracles $BRO$ and $BRO^{-1}$ respectively. The time complexities of $\mathcal{B}$ and $\mathcal{C}$ are at most $3T$.*

The proof of the theorem is provided in Appendix C.

The interpretation of the random oracle model and bijective random oracle model in our case is as follows. We do not cover the methods of cryptoanalysis that use the features of structure of the concrete commitment scheme to link its domain and range or exploit the connection between two algebraic structures: bit strings encoding the coordinates of elliptic curve points and the corresponding integers (see [3], [5]).

Let discuss the obtained security bound. Each term of the bound corresponds to the specific directions of cryptoanalysis that are meaningful for the proposed scheme. The first two terms reflect methods targeted at breaking the security of the underlying cryptographic mechanisms — the GOST signature scheme (in the no-message setting) and the hash function. Obviously, breaking each of these mechanisms allows to obtain a forgery for 2p-GOST.

The third term reflects methods of cryptoanalysis targeted at the commitment scheme (as a black box) and assuming the dishonest computation of commitment values by one of the parties. Indeed, this term is equal to the probability of guessing the input (output) of the commitment function, modelled as a random oracle, by its output (input) without querying it. Note that if the adversary is able to do so, the attacks described in Section 3 for the naive version of 2p-GOST become possible.

The last term in the bound reflects methods of cryptoanalysis assuming gathering the large number of (message, signature) pairs and exploiting some collisions or other connections of their values. A prime example of such attack is to find two signatures generated with the same $k = k_1 + k_2$ value and recovering the signing key. The success of such attack is of order $\frac{q_{sign}^2}{q}$.

Note that the obtained security bound demonstrates that our method of constructing two-party scheme based on the GOST scheme does not add any additional security assumptions except for the assumption that commitment is modelled as random oracle. Indeed, other two assumptions, bijective random oracle and signum-relative collision resistance of the used hash function family, are also the underlying assumptions for the security of the GOST scheme in the chosen-message setting (for details see [4]).

## 6 Conclusion

The first result of this paper is devoted to the analysis of existing signature schemes based on GOST signature equation, in which the signature is generated by several signers. We show that all these schemes are not suitable for providing signing key protection on the user mobile device. Some of these schemes use a trusted third party, others are proven secure in the weak security models. Moreover, we provide the attacks breaking unforgeability for some of these schemes.

The second result of this paper is devoted to the synthesis of the secure two-party signature scheme with the same verification algorithm as in the GOST signature scheme. We propose the 2p-GOST scheme which uses the commitment scheme. We prove that this scheme is secure in the case when one of the parties is malicious under the assumption that the classical GOST scheme is unforgeable and commitment scheme is modelled as a random oracle. This scheme can be used for providing signing key protection on the user mobile device.

## References

[1] Beresneva, Anastasia and Epishkina, Anna and Isupova, Olga and Kogos, Konstantin and Shimkiv, Mikhail, “Special digital signature schemes based on GOST R 34.10-2012”, *IEEE*, 2016, 135–140.

[2] Benhamouda, Fabrice and Lepoint, Tancrède and Loss, Julian and Orrù, Michele and Raykova, Mariana, “On the (in) security of ROS”, *Journal of Cryptology*, **35**:4 (2022), 25.

[3] Bellare, Mihir and Rogaway, Phillip, “Random oracles are practical: A paradigm for designing efficient protocols”, *Proceedings of the 1st ACM Conference on Computer and Communications Security*, =1993, 62–73.

[4] Fersch, Manuel, "The provable security of elgamal-type signature schemes", 2018.

[5] Fersch, Manuel and Kiltz, Eike and Poettering, Bertram, "On the provable security of (EC) DSA signatures", *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, 1651–1662.

[6] Gennaro, Rosario and Jarecki, Stanislaw and Krawczyk, Hugo and Rabin, Tal, "Secure distributed key generation for discrete-log based cryptosystems", *Journal of Cryptology*, **20** (2007), 51–83.

[7] *GOST R 34.10-2012. Information technology. Cryptographic data security. Signature and verification processes of electronic digital signature. National standard of the Russian Federation, STANDARTINFORM*, 2012, In Russian.

[8] *GOST 34.10-2018. Information technology. Cryptographic data security. Signature and verification processes of electronic digital signature. Interstate standard, Interstate Council for Standardization, Metrology and Certification (ISC)*, 2018, In Russian.

[9] *ISO/IEC 14888-3:2018, IT Security techniques – Digital signatures with appendix – Part 3: Discrete logarithm based mechanisms – Section 6: Certificate-based mechanisms – 6.9: ECRDSA*, 2018.

[10] Dolmatov V., Degtyarev A., *GOST R 34.10-2012: Digital Signature Algorithm*, RFC 7091, DOI 10.17487/RFC7091, 2013, https://www.rfc-editor.org/info/rfc7091.

[11] Komlo, Chelsea and Goldberg, Ian, "FROST: flexible round-optimized Schnorr threshold signatures", *Selected Areas in Cryptography: 27th International Conference, Halifax, NS, Canada (Virtual Event), October 21-23, 2020, Revised Selected Papers 27*, Springer, 2021, 34–65.

[12] Kim, Sungwook and Kim, Jihye and Cheon, Jung Hee and Ju, Seong-ho, "Threshold signature schemes for ElGamal variants", *Computer Standards & Interfaces*, **33**:4 (2011), 432–437.

[13] Kim, Tuan Nguyen and Ngoc, Duy Ho and Moldovyan, Nikolay A, "New Collective Signatures Based on the Elliptic Curve Discrete Logarithm Problem", *CMC-COMPUTERS MATERIALS & CONTINUA*, **73**:1 (2022), 595–610.

[14] Lindell, Yehuda, "Fast secure two-party ECDSA signing", *Advances in Cryptology–CRYPTO 2017: 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20–24, 2017, Proceedings, Part II 37*, Springer, 2017, 613–644.

[15] Moldovyan, Nikolai Andreevich, "Theoretical minimum and digital signature algorithms [there is a vulture]", 2010.

[16] Nicolosi, Antonio and Krohn, Maxwell N and Dodis, Yevgeniy and Mazieres, David, "Proactive Two-Party Signatures for User Authentication,", *NDSS*, 2003.

[17] Pointcheval, David and Stern, Jacques, "Provably secure blind signature schemes", *Advances in Cryptology—ASIACRYPT'96: International Conference on the Theory and Applications of Cryptology and Information Security Kyongju, Korea, November 3–7, 1996 Proceedings*, Springer, 1996, 252–265.

[18] Krawczyk H., Bellare M., Canetti R., *HMAC: Keyed-hashing for message authentication*, RFC2104, 1997, https://www.rfc-editor.org/info/rfc2104.

[19] Zhang, Yunru and Luo, Min and Choo, Kim-Kwang Raymond and Li, Li and He, Debiao, "Efficient and Secure Two-Party Distributed Signing Protocol for the GOST Signature Algorithm", *Security and Privacy in Social Networks and Big Data: 6th International Symposium, SocialSec 2020, Tianjin, China, September 26–27, 2020, Proceedings 6*, Springer, 2020, 3–19.

# A ROS-style attacks

## A.1 Scheme Zhang-Luo-Choo-Li-He

The scheme proposed in [19] uses the additively homomorphic encryption scheme $(Enc_{\mathsf{pk}}(\cdot), Dec_{\mathsf{sk}}(\cdot))$, where keys $(\mathsf{sk}, \mathsf{pk})$ are known to $\mathbf{P_1}$, and $c_d = Enc_{\mathsf{pk}}(d_1)$ is known to $\mathbf{P_2}$. In the attack, we use encryption scheme honestly, so details related to its correct using are omitted.

The signing protocol of this scheme is presented at Figure 8.

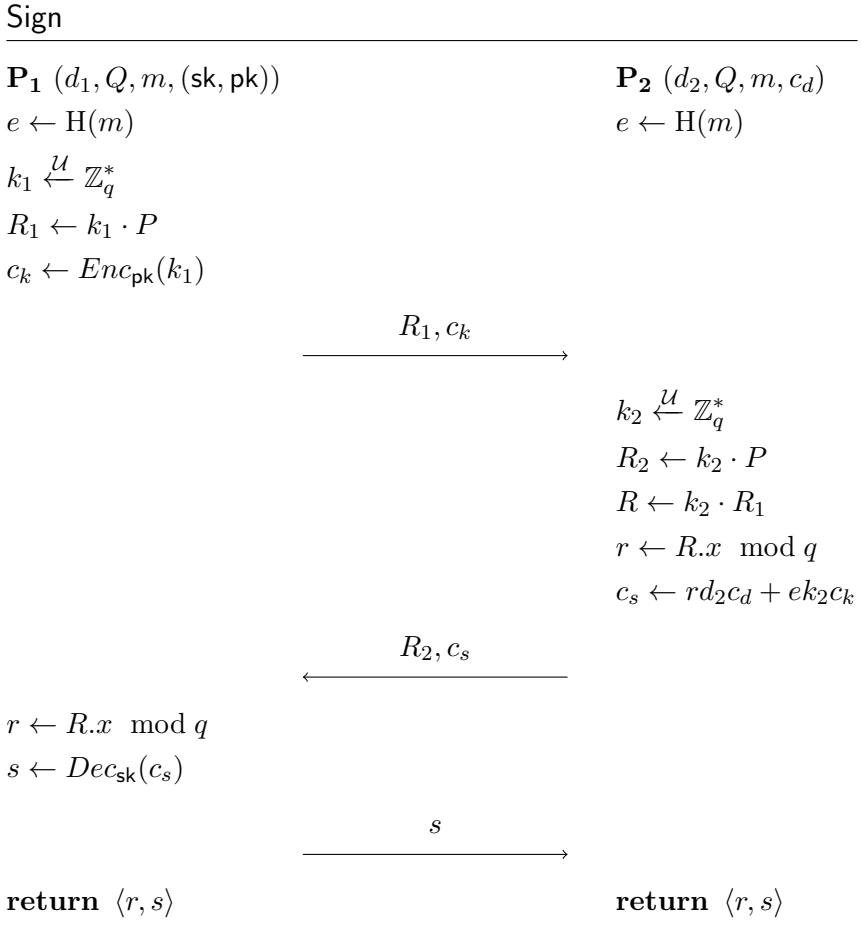



Figure 8: Signing protocol of the signature scheme [19].

The attack, presented below, allows an adversary acting as $\mathbf{P_2}$ to construct $(l+1)$ correct (message, signature) pairs after $l \geqslant \lceil \log q \rceil$ successful interactions with $\mathbf{P_1}$. The adversary acts as follows:

1. Selects message $m_l \in \{0,1\}^*$ for which a signature will be forged, let $e_l = \mathrm{H}(m_l)$.

2. Opens $l$ parallel sessions for some messages $m_0, \ldots, m_{l-1}$, querying $\mathbf{P_1}$,

let $e_i = \mathrm{H}(m_i), 0 \leqslant i \leqslant l-1$, and receives corresponding points $R_1^0, \ldots, R_1^{l-1}$.

3. Selects $k_2^{i,0}, k_2^{i,1} \in \mathbb{Z}_q^*, 0 \leqslant i \leqslant l-1$, then $R^{i,0} = k_2^{i,0} R_1^i$, $R^{i,1} = k_2^{i,1} R_1^i$, $0 \leqslant i \leqslant l-1$, $r_{i,0} = R^{i,0}.x \mod q$, $r_{i,1} = R^{i,1}.x \mod q$, $0 \leqslant i \leqslant l-1$, such that $k_2^{i,1^{-1}} r_{i,1} \neq k_2^{i,0^{-1}} r_{i,0}$, $0 \leqslant i \leqslant l-1$.

4. Defines $(\rho_0, \rho_1, \ldots, \rho_l)$ as the vector of coefficients placed before $x_i$ in the function $f : \mathbb{Z}_q^l \to \mathbb{Z}_q$; $f(x_0, \ldots, x_{l-1}) = \sum\limits_{i=0}^{l-1} 2^i \underbrace{\dfrac{x_i - k_2^{i,0^{-1}} r_{i,0}}{k_2^{i,1^{-1}} r_{i,1} - k_2^{i,0^{-1}} r_{i,0}}}_{b_i'} = \sum\limits_{i=0}^{l-1} \rho_i x_i + \rho_l$. Note that if $x_i = k_2^{i,0^{-1}} r_{i,0}$ then $b_i' = 0$, if $x_i = k_2^{i,1^{-1}} r_{i,1}$ then $b_i' = 1$ .

5. Defines $R^l = e_l^{-1} \left( \sum\limits_{i=0}^{l-1} \rho_i e_i R_1^i - \rho_l Q \right)$.

6. Defines $r_l = R^l.x \mod q$.

7. Defines $b_0, \ldots, b_{l-1}$ from the following equation: $r_l = \sum\limits_{i=0}^{l-1} 2^i b_i$.

8. Defines $k_2^i = k_2^{i,b_i}, r_i = r_{i,b_i}, 0 \leqslant i \leqslant l-1$; therefore, according to step 4, $r_l = \sum\limits_{i=0}^{l-1} 2^i b_i = \sum\limits_{i=0}^{l-1} \rho_i k_2^{i^{-1}} r_i + \rho_l$.

9. Defines $R_2^i = k_2^i P$, $0 \leqslant i \leqslant l-1$.

10. Calculates $c_s^0, \ldots, c_s^{l-1}$, according to the protocol.

11. Sends $R_2^0, \ldots, R_2^{l-1}$ and $c_s^0, \ldots, c_s^{l-1}$ values to $\mathbf{P_1}$ in the corresponding sessions;

12. Obtains responses $s^0, \ldots, s^{l-1}$ such that:

$$k_2^{i^{-1}} s^i = k_1^i e_i + d_1 d_2 r_i k_2^{i^{-1}}, \ 0 \leqslant i \leqslant l-1.$$

13. Defines $s^l = \sum\limits_{i=0}^{l-1} \rho_i k_2^{i^{-1}} s^i = \sum\limits_{i=0}^{l-1} \rho_i k_1^i e_i + \sum\limits_{i=0}^{l-1} \rho_i d_1 d_2 r_i k_2^{i^{-1}}$.

14. Outputs $\{m_i, (r_i, s^i)\}_{i=0}^l$.

Indeed, for $0 \leqslant i \leqslant l-1$ signature $(r_i, s^i)$ is valid for $m_i$ by attack construction. Consider the case $i = l$.

We show that the following signature verification equation holds:

$$e_l{}^{-1} s^l P = R^l + e_l{}^{-1} r_l Q.$$

The left side:

$$e_l{}^{-1} s^l P = e_l{}^{-1} \sum_{i=0}^{l-1} \rho_i e_i R_1^i + Q e_l{}^{-1} \sum_{i=0}^{l-1} \rho_i r_i k_2^{i\,-1}.$$

The right side:

$$R^l + e_l{}^{-1} r_l Q = e_l^{-1} \left( \sum_{i=0}^{l-1} \rho_i e_i R_1^i - \rho_l Q \right) + e_l{}^{-1} Q \left( \sum_{i=0}^{l-1} \rho_i k_2^{i\,-1} r_i + \rho_l \right) =$$
$$= e_l{}^{-1} \sum_{i=0}^{l-1} \rho_i e_i R_1^i + Q e_l{}^{-1} \sum_{i=0}^{l-1} \rho_i r_i k_2^{i\,-1}.$$

A condition $l \geqslant \lceil \log q \rceil$ is is necessary in order to be able to carry out the step 7.

### A.2 ROS attack on the straightforward scheme

This section contains some modification of the attack, described in Appendix A.1, for scheme defined at Figure 2.

We describe only the different steps:

3. Selects $k_2^{i,0}, k_2^{i,1} \in \mathbb{Z}_q^*, 0 \leqslant i \leqslant l-1$, then $R^{i,0} = k_2^{i,0} P + R_1^i$, $R^{i,1} = k_2^{i,1} P + R_1^i$, $0 \leqslant i \leqslant l-1$, $r_{i,0} = R^{i,0}.x \mod q$, $r_{i,1} = R^{i,1}.x \mod q$, $0 \leqslant i \leqslant l-1$, such that $r_{i,1} e_l (e_i)^{-1} \neq r_{i,0} e_l (e_i)^{-1}$, $0 \leqslant i \leqslant l-1$.

4. Defines $(\rho_0, \rho_1, \ldots, \rho_l)$ as the vector of coefficients placed before $x_i$ in the function $f : \mathbb{Z}_q^l \to \mathbb{Z}_q$; $f(x_0, \ldots, x_{l-1}) = \sum_{i=0}^{l-1} 2^i \underbrace{\frac{x_i - r_{i,0} e_l (e_i)^{-1}}{r_{i,1} e_l (e_i)^{-1} - r_{i,0} e_l (e_i)^{-1}}}_{b_i'} = \sum_{i=0}^{l-1} \rho_i x_i + \rho_l$. Note that if $x_i = r_{i,0} e_l (e_i)^{-1}$ then $b_i' = 0$, if $x_i = r_{i,1} e_l (e_i)^{-1}$ then $b_i' = 1$ .

5. Defines $R_1^l = \sum_{i=0}^{l-1} \rho_i R_1^i - e_l^{-1} \rho_l Q_1$.

6. Selects $k_2^l \in \mathbb{Z}_q^*$ and defines $R_2^l = k_2^l P$. Defines $R^l = R_1^l + R_2^l$ and $r_l = R^l.x \mod q$.

8. Defines $k_2^i = k_2^{i,b_i}, r_i = r_{i,b_i}, 0 \leqslant i \leqslant l-1$; $r_l = \sum\limits_{i=0}^{l-1} 2^i b_i = e_l \sum\limits_{i=0}^{l-1} \rho_i e_i^{-1} r_i + \rho_l$.

10. Calculates $s_2^0, \ldots, s_2^{l-1}$, according to the protocol.

11. Sends $R_2^0, \ldots, R_2^{l-1}$ and $s_2^0, \ldots, s_2^{l-1}$ values to $\mathbf{P_1}$ in the corresponding sessions.

12. Obtains responses $s_1^0, \ldots, s_1^{l-1}$ such that:
$$s_1^i = k_1^i e_i + d_1 r_i \; 0 \leqslant i \leqslant l-1.$$

13 Defines $s_1^l = e_l \sum\limits_{i=0}^{l-1} \rho_i {e_i}^{-1} s_1^i$, $s_2^l = k_2^l e_l + r_l d_2$, $s^i = s_1^i + s_2^i; 0 \leqslant i \leqslant l$.

Indeed, for $0 \leqslant i \leqslant l-1$ signature $(r_i, s^i)$ is valid for $m_i$ by attack construction. Consider the case $i = l$.

We show that the following signature verification equation holds:

$$R^l = {e_l}^{-1}(s^l P - r_l Q).$$

$$\begin{aligned}
{e_l}^{-1}(s^l P - r_l Q) &= {e_l}^{-1}(s_1^l P + s_2^l P - r_l Q_1 - r_l Q_2) = \\
&= \sum_{i=0}^{l-1} \rho_i {e_i}^{-1} s_1^i P + k_2^l P + e_l^{-1} r_l d_2 P - e_l^{-1} r_l Q_2 - e_l^{-1} r_l Q_1 = \\
&= \sum_{i=0}^{l-1} \rho_i {e_i}^{-1} s_1^i P - e_l^{-1} r_l Q_1 + R_2^l = \\
&= \sum_{i=0}^{l-1} \rho_i {e_i}^{-1} s_1^i P - Q_1 \left( \sum_{i=0}^{l-1} \rho_i e_i^{-1} r_i + e_l^{-1} \rho_l \right) + R_2^l = \\
&= \sum_{i=0}^{l-1} \rho_i \underbrace{e_i^{-1}(s_1^i P - r_i Q_1)}_{=R_1^i} - \rho_l e_l^{-1} Q_1 + R_2^l = \\
&= \underbrace{\sum_{i=0}^{l-1} \rho_i R_1^i - \rho_l e_l^{-1} Q_1}_{=R_1^l} + R_2^l = \\
&= R_1^l + R_2^l = R^l.
\end{aligned}$$

### A.3 ROS attack on the scheme with sent message

Let's describe a ROS attack on the following modification of the singing protocol at Figure 2: the message $m$ is argument only for $\mathbf{P_1}$, the party $\mathbf{P_2}$ receives $m$ from the party $\mathbf{P_1}$ in the third transmission.

This attack uses an opportunity to open several parallel sessions. The attack allows an adversary acting as $\mathbf{P_1}$ to construct $(l+1)$ correct (message, signature) pairs after $l \geqslant \lceil \log q \rceil$ successful interactions with $\mathbf{P_2}$.

The adversary acts as follows:

1. Selects message $m_l \in \{0,1\}^*$ for which a signature will be forged, let $e_l = \mathrm{H}(m_l)$.
2. Opens $l$ parallel sessions, selects $R_1^i = k_1^i P$, $0 \leqslant i \leqslant l-1$, and sends corresponding $comm_R^0, \ldots, comm_R^{l-1}$ to the second user. Receives $R_2^0, \ldots, R_2^{l-1}$.
3. Defines $r_i = (R_1^i + R_2^i).x \mod q, 0 \leqslant i \leqslant l-1$.
4. Selects $m_i^0, m_i^1, 0 \leqslant i \leqslant l-1$, such that $r'_{i,0} \neq r'_{i,1}$, where:
$$e_i^0 = \mathrm{H}(m_i^0),\ e_i^1 = \mathrm{H}(m_i^1),$$
$$r'_{i,0} = e_l(e_i^0)^{-1} r_i,\ r'_{i,1} = e_l(e_i^1)^{-1} r_i.$$
5. Defines $(\rho_0, \rho_1, \ldots, \rho_l)$ as the vector of coefficients placed before $x_i$ in the function $f : \mathbb{Z}_q^l \to \mathbb{Z}_q$; $f(x_0, \ldots, x_{l-1}) = \sum\limits_{i=0}^{l-1} \underbrace{2^i \dfrac{x_i - r'_{i,0}}{r'_{i,1} - r'_{i,0}}}_{b'_i} = \sum\limits_{i=0}^{l-1} \rho_i x_i + \rho_l$.
Note that if $x_i = r'_{i,0}$ then $b'_i = 0$, if $x_i = r'_{i,1}$ then $b'_i = 1$ .
6. Defines $R_2^l = \sum\limits_{i=0}^{l-1} \rho_i R_2^i - e_l^{-1} \rho_l Q_2$.
7. Selects $k_1^l$ from $\mathbb{Z}_q^*$ and defines $R_1^l = k_1^l P$.
8. Defines $r_l = (R_1^l + R_2^l).x \mod q$.
9. Defines $b_0, \ldots, b_{l-1}$ from the following equation: $r_l = \sum\limits_{i=0}^{l-1} 2^i b_i$.
10. Defines $r'_i = r'_{i,b_i}, e_i = e_i^{b_i}, m_i = m_i^{b_i}, 0 \leqslant i \leqslant l-1$; therefore $r_l = \sum\limits_{i=0}^{l-1} \rho_i r'_i + \rho_l = e_l \sum\limits_{i=0}^{l-1} \rho_i e_i^{-1} r_i + \rho_l$.

11. Calculates $s_1^i$, according to the protocol: $s_1^i = k_1^i \cdot e_i + r_i \cdot d_1$.

12. Sends $op_R^i, R_1^i, s_1^i,\ 0 \leqslant i \leqslant l-1$, values to $\mathbf{P_2}$ in the corresponding opened sessions.

13. Obtains responses $s_2^0, \ldots, s_2^{l-1}$such that:

$$s_2^i P = e_i R_2^i + r_i Q_2,\ 0 \leqslant i \leqslant l-1.$$

14. Defines $s_2^l = e_l \sum\limits_{i=0}^{l-1} \rho_i e_i^{-1} s_2^i$. Calculates $s_1^l = k_1^l e_l + r_l \cdot d_1$.

15. Defines $s^i = s_1^i + s_2^i, 0 \leqslant i \leqslant l$.

16. Outputs $\{m_i, (r_i, s^i)\}_{i=0}^l$.

Indeed, for $0 \leqslant i \leqslant l-1$ signature $(r_i, s^i)$ is valid for $m_i$ by attack construction. Consider the case $i = l$.

We show that the following signature verification equation holds:

$$R^l = {e_l}^{-1}(s^l P - r_l Q).$$

$$R^{l'} = {e_l}^{-1}\left(s^l P - r_l Q\right) = {e_l}^{-1}\left((s_1^l + s_2^l)P - r_l(Q_1 + Q_2)\right) =$$

$$= {e_l}^{-1}\left(e_l \sum_{i=0}^{l-1} \rho_i e_i^{-1} s_2^i P - \left(e_l \sum_{i=0}^{l-1} \rho_i e_i^{-1} r_i + \rho_l\right) \cdot Q_2 + \underbrace{(s_1^l P - r_l Q_1)}_{=e_l R_1^l}\right) =$$

$$= \sum_{i=0}^{l-1} \rho_i e_i^{-1} s_2^i P - \sum_{i=0}^{l-1} \rho_i e_i^{-1} r_i Q_2 - {e_l}^{-1}\rho_l Q_2 + R_1^l =$$

$$= \sum_{i=0}^{l-1} \rho_i \underbrace{e_i^{-1}(s_2^i P - r_i Q_2)}_{=R_2^i} - {e_l}^{-1}\rho_l Q_2 + R_1^l = \underbrace{\sum_{i=0}^{l-1} \rho_i R_2^i - {e_l}^{-1}\rho_l Q_2}_{=R_2^l} + R_1^l = R_2^l + R_1^l.$$

and $R^{l'}.x = r_l \mod q$ from the step 8.

A condition $l \geqslant \lceil \log q \rceil$ is is necessary in order to be able to carry out the step 9.

# B Security notions

In this section we formally define the security model used for two-party signature schemes.

**Definition 3.** *For a two-party signature scheme* 2p-SS

$$\mathsf{Adv}^{\mathsf{sOMUF\text{-}PCA}}_{\mathsf{2p\text{-}SS}}(\mathcal{A}) = \Pr\left[\mathbf{Exp}^{\mathsf{sOMUF\text{-}PCA}}_{\mathsf{2p\text{-}SS}}(\mathcal{A}) \to 1\right],$$

*where the experiment* $\mathbf{Exp}^{\mathsf{sOMUF\text{-}PCA}}_{\mathsf{2p\text{-}SS}}(\mathcal{A})$ *is defined in the following way:*

$\mathbf{Exp}^{\mathsf{sOMUF\text{-}PCA}}_{\mathsf{2p\text{-}SS}}(\mathcal{A})$

$\Pi \xleftarrow{\mathcal{U}} Perm\left(\{0,1\}^N \to \{0,\ldots,2^N-1\}\right)$
$\Pi_R \xleftarrow{\mathcal{U}} Func(\{0,1\}^\kappa \times \mathbb{G} \to \{0,1\}^n)$
$\Pi_Q \xleftarrow{\mathcal{U}} Func(\{0,1\}^\kappa \times \mathbb{G} \to \{0,1\}^n)$
$l \leftarrow 0,\ SESS \leftarrow \emptyset$
$sid \leftarrow -1$
$round_{kg} \leftarrow 0,\ ctx_{kg} \leftarrow \emptyset$
$p \leftarrow \mathcal{A}()$
**if** $(p \neq 1 \wedge p \neq 2)$ : **return** $\perp$
$(Q, d_p) \leftarrow (\varepsilon, \varepsilon)$
$\{(m_i, \langle r_i, s_i\rangle)\}_{i=1}^{l+1} \xleftarrow{\$} \mathcal{A}^{KGen,NewSign,Sign,BRO,BRO^{-1},rRO,qRO}(p)$
**return** $((\forall i \neq j \in \{1,\ldots,l+1\}:$
$(m_i, \langle r_i, s_i\rangle) \neq (m_j, \langle r_j, s_j\rangle)) \wedge$
$\wedge\ (\forall i \in \{1,\ldots,l+1\} : \mathsf{Verify}(Q, m_i, \langle r_i, s_i\rangle)))$

*Oracle* $BRO(\alpha)$

**return** $\Pi(\alpha)$

*Oracle* $BRO^{-1}(\beta)$

**return** $\Pi^{-1}(\beta)$

$rRO(op_R, R)$

**return** $\Pi_R(op_R, R)$

$qRO(op_Q, Q)$

**return** $\Pi_Q(op_Q, Q)$

$KGen(msg)$

**if** $(Q \neq \varepsilon)$ : **return** $\perp$
**return** $\mathsf{ExecKGen}^p(msg)$

$NewSign(m)$

**if** $(Q = \varepsilon \vee d_p = \varepsilon)$ : **return** $\perp$
$round \leftarrow 0,\ ctx \leftarrow \{round\},\ flag \leftarrow 0$
$state \leftarrow (m, ctx, flag)$
$sid \leftarrow sid + 1$
$SESS \leftarrow SESS \cup \{(sid, state)\}$
**return** $sid$

$Sign(sid, msg)$

**if** $((sid, \cdot) \notin SESS)$ : **return** $\perp$
$state \leftarrow SESS[sid]$
$(state', msg') \leftarrow \mathsf{ExecSign}^p(state, msg)$
**if** $(msg' = \perp)$ : **return** $\perp$
$SESS[sid] \leftarrow state'$
$(m, ctx, flag) \leftarrow state'$
**if** $(flag)$ : $l \leftarrow l + 1$
**return** $msg'$

*where* $\mathsf{ExecKGen}^p$ *and* $\mathsf{ExecSign}^p$ *are functions that define the execution of the* KGen *and* Sign *protocols of the* 2p-SS *scheme by an uncompromised party, i.e.* $P_{3-p}$.

Let's define the functions $\mathsf{ExecKGen}^p$ and $\mathsf{ExecSign}^p$, where $p = 1, 2$, for the 2p-GOST scheme.

$\mathsf{ExecKGen}^1(msg)$

1: **if** $(round_{kg} = 0)$ :
2: $\quad d_1 \xleftarrow{\mathcal{U}} \mathbb{Z}_q^*$
3: $\quad Q_1 \leftarrow d_1 P$
4: $\quad op_Q \xleftarrow{\mathcal{U}} \{0,1\}^\kappa$
5: $\quad comm_Q \leftarrow qRO(op_Q, Q_1)$
6: $\quad msg' \leftarrow \{comm_Q\}$
7: **else if** $(round_{kg} = 1)$ :
8: $\quad Q_2 \leftarrow msg$
9: $\quad$ **if** $(Q_2 = -Q_1)$ : **return** $\perp$
10: $\quad Q \leftarrow Q_1 + Q_2$
11: $\quad msg' \leftarrow \{op_Q, Q_1\}$
12: **else** :
13: $\quad msg' \leftarrow \varepsilon$
14: $round_{kg} \leftarrow round_{kg} + 1$
15: // Update the $ctx_{kg}$ value
16: **return** $msg'$

$\mathsf{ExecSign}^1(state, msg)$

1: $round \leftarrow state.ctx.round$
2: **if** $(round = 0)$ :
3: $\quad e \leftarrow \mathrm{H}(state.m)$
4: $\quad$ **if** $(e = 0) : e \leftarrow 1$
5: $\quad k_1 \xleftarrow{\mathcal{U}} \mathbb{Z}_q$
6: $\quad R_1 \leftarrow k_1 P$
7: $\quad op_R \xleftarrow{\mathcal{U}} \{0,1\}^\kappa$
8: $\quad comm_R \leftarrow rRO(op_R, R_1)$
9: $\quad msg' \leftarrow \{comm_R\}$
10: **else if** $(round = 1)$ :
11: $\quad R_2 \leftarrow msg$
12: $\quad$ **if** $(R_2 = -R_1)$ :
13: $\quad\quad$ **return** $(state, \perp)$
14: $\quad R \leftarrow R_1 + R_2$
15: $\quad r \leftarrow \psi(\Pi(\phi(R)))$
16: $\quad$ **if** $(r = 0)$ : **return** $(state, \perp)$
17: $\quad s_1 \leftarrow k_1 \cdot e + d_1 \cdot r$
18: $\quad state.flag \leftarrow 1$
19: $\quad msg' \leftarrow \{op_R, R_1, s_1\}$
20: **else if** $(round = 2)$ :
21: $\quad s_2 \leftarrow msg$
22: $\quad s \leftarrow s_1 + s_2$
23: $\quad msg' \leftarrow \varepsilon$
24: $\quad$ **if** $(\mathsf{Verify}(Q, state.m, \langle r, s\rangle) = 0)$ :
25: $\quad\quad$ **return** $(state, \perp)$
26: **else**
27: $\quad msg' \leftarrow \varepsilon$
28: // Update the $state.ctx$ value
29: **return** $(state, msg')$

$\mathsf{ExecKGen}^2(msg)$

1: **if** $(round_{kg} = 0)$ :
2: $\quad comm_Q \leftarrow msg$
3: $\quad d_2 \xleftarrow{\mathcal{U}} \mathbb{Z}_q^*$
4: $\quad Q_2 \leftarrow d_2 P$
5: $\quad msg' \leftarrow \{Q_2\}$
6: **else if** $(round_{kg} = 1)$ :
7: $\quad op_Q, Q_1 \leftarrow msg$
8: $\quad$ **if** $(comm_Q \neq qRO(op_Q, Q_1))$ :
9: $\quad\quad$ **return** $\perp$
10: $\quad$ **if** $(Q_1 = -Q_2)$ : **return** $\perp$
11: $\quad Q \leftarrow Q_1 + Q_2$
12: $\quad msg' \leftarrow \varepsilon$
13: **else** :
14: $\quad msg' \leftarrow \varepsilon$
15: $round_{kg} \leftarrow round_{kg} + 1$
16: // Update the $ctx_{kg}$ value
17: **return** $msg'$

$\mathsf{ExecSign}^2(state, msg)$

1: $round \leftarrow state.ctx.round$
2: **if** $(round = 0)$ :
3: $\quad e \leftarrow \mathrm{H}(state.m)$
4: $\quad$ **if** $(e = 0) : e \leftarrow 1$
5: $\quad comm_R \leftarrow msg$
6: $\quad k_2 \xleftarrow{\mathcal{U}} \mathbb{Z}_q$
7: $\quad R_2 \leftarrow k_2 P$
8: $\quad msg' \leftarrow \{R_2\}$
9: **else if** $(round = 1)$ :
10: $\quad (op_R, R_1, s_1) \leftarrow msg$
11: $\quad$ **if** $(comm_R \neq rRO(op_R, R_1))$ :
12: $\quad\quad$ **return** $(state, \perp)$
13: $\quad$ **if** $(R_1 = -R_2)$ :
14: $\quad\quad$ **return** $(state, \perp)$
15: $\quad R \leftarrow R_1 + R_2$
16: $\quad r \leftarrow \psi(\Pi(\phi(R)))$
17: $\quad s_2 \leftarrow k_2 \cdot e + d_2 \cdot r$
18: $\quad s \leftarrow s_1 + s_2$
19: $\quad state.flag \leftarrow 1$
20: $\quad msg' \leftarrow \{s_2\}$
21: $\quad$ **if** $(\mathsf{Verify}(Q, state.m, \langle r, s\rangle) = 0)$ :
22: $\quad\quad$ **return** $(state, \perp)$
23: **else** :
24: $\quad msg' \leftarrow \varepsilon$
25: // Update the $state.ctx$ value
26: **return** $(state, msg')$

# C Security proof of the scheme

*Proof.* Let's $\mathbf{Exp}^0(\mathcal{A})$ denote the original security experiment as defined in the sOMUF-PCA security model definition (see Definition 3). We fix $\mathcal{A}$ – the adversary that makes forgery for the 2p-GOST scheme in the sOMUF-PCA model. The adversary has the access to the random oracles $rRO$, $qRO$, the bijective random oracles $BRO$ and $BRO^{-1}$, the key generation oracle $KGen$, the $NewSign$ oracle, initiating a new signing session, and the signing oracle $Sign$. We assume that adversary can make at most $q_R$ and $q_Q$ queries to the oracles $rRO$ and $qRO$ respectively, at most $q_{BRO}$ and $q_{BRO^{-1}}$ queries to the oracles $BRO$ and $BRO^{-1}$ respectively and at most $q_{sign}$ queries to the oracle $NewSign$. Our goal is to upper-bound $\Pr\left[\mathbf{Exp}^{\mathsf{sOMUF\text{-}PCA}}_{\mathsf{2p\text{-}GOST}}(\mathcal{A}) \to 1\right] = \Pr\left[\mathbf{Exp}^0(\mathcal{A}) \to 1\right]$.

**Construction of adversary $\mathcal{C}$.** $\mathbf{Exp}^1(\mathcal{A})$ is the modification of the $\mathbf{Exp}^0(\mathcal{A})$ obtained by implementing $\Pi$, $\Pi_R$, $\Pi_Q$ using «lazy sampling» (see Figure 9). Here and after we denote the difference between experiments by color in pseudocode. We write **abort** in the experiment pseudocode as a shortcut

for **return** 0 and in the oracle pseudocode to denote that experiment should stop and return 0.

The idea is to «open» new pairs $(\alpha, \Pi(x))$ and triples $(op_R, R, \Pi_R(op_R, R))$ or $(op_Q, Q, \Pi_Q(op_Q, Q))$ as soon as the adversary asks for it. From now onward we denote by $\Pi$ the subset of $(\{0,1\}^N, \{0,\ldots,2^N-1\})$, which is defined by the union of two sets $\Pi^S$ and $\Pi^O$. We store the pairs obtained from queries to the $BRO$ and $BRO^{-1}$ oracles in $\Pi^O$ set and the pairs obtained from queries to the $Sign$ oracle in $\Pi^S$ set. If $(\alpha, \beta) \in \Pi$, we denote $\beta$ as $\Pi(\alpha)$ and $\alpha$ as $\Pi^{-1}(\beta)$. We write $(\alpha, \cdot) \in \Pi$ shorthand for the condition that there exists $\beta$ such that $(\alpha, \beta) \in \Pi$. We write $(\cdot, \alpha) \in \Pi$ shorthand for the condition that there exists $\beta$ such that $(\alpha, \beta) \in \Pi$. Analogically, we denote by $\Pi_R$ and $\Pi_Q$ the subsets of $(\{0,1\}^\kappa, \mathbb{G}, \{0,1\}^n)$, that store the triples obtained from queries to the $rRO$ and $qRO$ oracle respectively. The shorthands for the conditions that there exist the triples belonging to the corresponding sets are defined in the same way as for $\Pi$ set.

These modifications do not affect the distribution on $qRO$ and $rRO$ outputs. There are the following differences between $\mathbf{Exp}^0(\mathcal{A})$ and $\mathbf{Exp}^1(\mathcal{A})$ in implementing permutation $\Pi$:

1. at the $BRO$ oracle: **abort** if $(\cdot, \beta) \in \Pi$ (line 3);
2. at the $BRO^{-1}$ oracle: **abort** if $(\alpha, \cdot) \in \Pi$ (line 3);
3. at the $Sign$ oracle in the function $\mathsf{ExecSign}^1$ or the function $\mathsf{ExecSign}^2$: **abort** if $(\cdot, \beta) \in \Pi$ (line 19 or 24).

To estimate the difference between $\mathbf{Exp}^0(\mathcal{A})$ and $\mathbf{Exp}^1(\mathcal{A})$, we should estimate the probability that $\mathbf{Exp}^1(\mathcal{A})$ aborts in these ways.

Let's consider the $BRO$ oracle. Note that is executed not only when adversary makes direct query to it but also during the Verify procedure (in signing oracle and finalization of the experiment). Since the number of forgeries does not exceed $(q_{sign}+1)$, the number of $BRO$ executions does not exceed $(q_{BRO}+2q_{sign}+1)$. The value $\beta$ is uniformly distributed on a set $\{0,\ldots,2^N-1\}$ of cardinality $2^N$. In the worst case the adversary $\mathcal{A}$ has already made all queries to the $BRO$, $BRO^{-1}$, $Sign$ oracles and thus $\Pi$ contains at least $(q_{BRO}+q_{BRO^{-1}}+3q_{sign}+1)$ elements. The abort condition is met if the value $\beta$ hits one of elements in $\Pi$. We can estimate this probability as $\dfrac{q_{BRO}+q_{BRO^{-1}}+3q_{sign}+1}{2^N}$. As oracle $BRO$ is executed at most $(q_{BRO}+2q_{sign}+1)$ times, the overall probability can be bounded by $(q_{BRO}+2q_{sign}+1)\cdot\dfrac{q_{BRO}+q_{BRO^{-1}}+3q_{sign}+1}{2^N}$.

Similarly, consider the $BRO^{-1}$ and $Sign$ oracles. We get the following:

$$\Pr\left[\textbf{abort in line 3 at the } BRO^{-1} \text{ oracle}\right] \leqslant$$
$$\leqslant q_{BRO^{-1}} \cdot \frac{q_{BRO} + q_{BRO^{-1}} + 3q_{sign} + 1}{2^N};$$
$$\Pr\left[\textbf{abort in line 19 in } \mathsf{ExecSign}^1 \text{ at the } Sign \text{ oracle}\right] =$$
$$= \Pr\left[\textbf{abort in line 24 in } \mathsf{ExecSign}^2 \text{ at the } Sign \text{ oracle}\right] \leqslant$$
$$\leqslant q_{sign} \cdot \frac{q_{BRO} + q_{BRO^{-1}} + 3q_{sign} + 1}{2^N}.$$

Thus,

$$\Pr\left[\mathbf{Exp}^0(\mathcal{A}) \to 1\right] - \Pr\left[\mathbf{Exp}^1(\mathcal{A}) \to 1\right] \leqslant \frac{(q_{BRO} + q_{BRO^{-1}} + 3q_{sign} + 1)^2}{2^N}.$$

**$\mathbf{Exp}^1(\mathcal{A})$**

1 : $(\Pi^O, \Pi^S) \leftarrow (\emptyset, \emptyset), \Pi \leftarrow \Pi^O \cup \Pi^S$
2 : $\Pi_R \leftarrow \emptyset$
3 : $\Pi_Q \leftarrow \emptyset$
4 : $l \leftarrow 0,\ SESS \leftarrow \emptyset$
5 : $sid \leftarrow -1$
6 : $round_{kg} \leftarrow 0,\ ctx_{kg} \leftarrow \emptyset$
7 : $p \leftarrow \mathcal{A}()$
8 : **if** $(p \neq 1 \wedge p \neq 2)$ : **return** $\perp$
9 : $(Q, d_p) \leftarrow (\varepsilon, \varepsilon)$
10 : $\{(m_i, \langle r_i, s_i\rangle)\}_{i=1}^{l+1} \xleftarrow{\$} \mathcal{A}^{KGen, NewSign, Sign, BRO, BRO^{-1}, rRO, qRO}(p)$
11 : **return** $((\forall i \neq j \in \{1, \ldots, l+1\}$ :
12 : $(m_i, \langle r_i, s_i\rangle) \neq (m_j, \langle r_j, s_j\rangle)) \wedge$
13 : $\wedge\ (\forall i \in \{1, \ldots, l+1\} : \mathsf{Verify}(Q, m_i, \langle r_i, s_i\rangle)))$

Oracle $BRO(\alpha)$

1 : **if** $(\alpha, \cdot) \in \Pi$ : **return** $\Pi(\alpha)$
2 : $\beta \xleftarrow{\mathcal{U}} \{0, \ldots, 2^N - 1\}$
3 : **if** $((\cdot, \beta) \in \Pi)$ : **abort**
4 : $\Pi^O \leftarrow \Pi^O \cup \{(\alpha, \beta)\}$
5 : $\Pi \leftarrow \Pi^O \cup \Pi^S$
6 : **return** $\beta$

Oracle $BRO^{-1}(\beta)$

1 : **if** $(\cdot, \beta) \in \Pi$ : **return** $\Pi^{-1}(\beta)$
2 : $\alpha \xleftarrow{\mathcal{U}} \{0,1\}^N$
3 : **if** $((\alpha, \cdot) \in \Pi)$ : **abort**
4 : $\Pi^O \leftarrow \Pi^O \cup \{(\alpha, \beta)\}$
5 : $\Pi \leftarrow \Pi^O \cup \Pi^S$
6 : **return** $\beta$

$rRO(op_R, R)$

1 : **if** $((op_R, R, \cdot) \in \Pi_R)$ : **return** $\Pi_R(op_R, R)$
2 : $comm_R \xleftarrow{\mathcal{U}} \{0,1\}^n$
3 : $\Pi_R \leftarrow \Pi_R \cup \{(op_R, R, comm_R)\}$
4 : **return** $comm_R$

$qRO(op_Q, Q)$

1 : **if** $((op_Q, Q, \cdot) \in \Pi_Q)$ : **return** $\Pi_Q(op_Q, Q)$
2 : $comm_Q \xleftarrow{\mathcal{U}} \{0,1\}^n$
3 : $\Pi_Q \leftarrow \Pi_Q \cup \{(op_Q, Q, comm_Q)\}$
4 : **return** $comm_Q$

$\mathsf{ExecSign}^1(state, msg)$ $(\mathbf{Exp}^1)$

1 : $round \leftarrow state.ctx.round$
2 : **if** $(round = 0)$ :
3 : $e \leftarrow \mathrm{H}(state.m)$
4 : **if** $(e = 0) : e \leftarrow 1$
5 : $k_1 \xleftarrow{\mathcal{U}} \mathbb{Z}_q$
6 : $R_1 \leftarrow k_1 P$
7 : $op_R \xleftarrow{\mathcal{U}} \{0,1\}^\kappa$
8 : $comm_R \leftarrow rRO(op_R, R_1)$
9 : $msg' \leftarrow \{comm_R\}$
10 : **else if** $(round = 1)$ :
11 : $R_2 \leftarrow msg$
12 : **if** $(R_2 = -R_1)$ :
13 : **return** $(state, \perp)$
14 : $R \leftarrow R_1 + R_2$
15 : **if** $((\phi(R), \cdot) \in \Pi)$ :
16 : $r \leftarrow \psi(\Pi(\phi(R)))$
17 : **else**
18 : $\beta \xleftarrow{\mathcal{U}} \{0, \ldots, 2^N - 1\}$
19 : **if** $((\cdot, \beta) \in \Pi)$ : **abort**
20 : $\Pi^S \leftarrow \Pi^S \cup \{(\phi(R), \beta)\}$
21 : $\Pi \leftarrow \Pi^S \cup \Pi^O$
22 : $r \leftarrow \psi(\beta)$
23 : **if** $(r = 0)$ : **return** $(state, \perp)$
24 : $s_1 \leftarrow k_1 \cdot e + d_1 \cdot r$
25 : $state.flag \leftarrow 1$
26 : $msg' \leftarrow \{op_R, R_1, s_1\}$
27 : **else if** $(round = 2)$ :
28 : $s_2 \leftarrow msg$
29 : $s \leftarrow s_1 + s_2$
30 : $msg' \leftarrow \varepsilon$
31 : **if** $(\mathsf{Verify}(Q, state.m, \langle r, s\rangle) = 0)$ :
32 : **return** $(state, \perp)$
33 : **else**
34 : $msg' \leftarrow \varepsilon$
35 : // Update the *state.ctx* value
36 : **return** $(state, msg')$

$\mathsf{ExecSign}^2(state, msg)$ $(\mathbf{Exp}^1)$

1 : $round \leftarrow state.ctx.round$
2 : **if** $(round = 0)$ :
3 : $e \leftarrow \mathrm{H}(state.m)$
4 : **if** $(e = 0) : e \leftarrow 1$
5 : $comm_R \leftarrow msg$
6 : $k_2 \xleftarrow{\mathcal{U}} \mathbb{Z}_q$
7 : $R_2 \leftarrow k_2 P$
8 : $msg' \leftarrow \{R_2\}$
9 : **else if** $(round = 1)$ :
10 : $(op_R, R_1, s_1) \leftarrow msg$
11 : **if** $(comm_R \neq rRO(op_R, R_1))$ :
12 : **return** $(state, \perp)$
13 : **if** $(R_1 = -R_2)$ :
14 : **return** $(state, \perp)$
15 : $R \leftarrow R_1 + R_2$
16 : **if** $((\phi(R), \cdot) \in \Pi)$ :
17 : $r \leftarrow \psi(\Pi(\phi(R)))$
18 : **else**
19 : $\beta \xleftarrow{\mathcal{U}} \{0, \ldots, 2^N - 1\}$
20 : **if** $((\cdot, \beta) \in \Pi)$ : **abort**
21 : $\Pi^S \leftarrow \Pi^S \cup \{(\phi(R), \beta)\}$
22 : $\Pi \leftarrow \Pi^S \cup \Pi^O$
23 : $r \leftarrow \psi(\beta)$
24 : $s_2 \leftarrow k_2 \cdot e + d_2 \cdot r$
25 : $s \leftarrow s_1 + s_2$
26 : $state.flag \leftarrow 1$
27 : $msg' \leftarrow \{s_2\}$
28 : **if** $(\mathsf{Verify}(Q, state.m, \langle r, s\rangle) = 0)$ :
29 : **return** $(state, \perp)$
30 : **else** :
31 : $msg' \leftarrow \varepsilon$
32 : // Update the *state.ctx* value
33 : **return** $(state, msg')$

Figure 9: $\mathbf{Exp}^1(\mathcal{A})$ for the 2p-GOST scheme in the sOMUF-PCA model.

$\mathbf{Exp}^2$ is the modification of the $\mathbf{Exp}^1$ in which forgeries obtained by finding a signum-relative collision are not counted (see Figure 10).

To estimate the difference between the $\mathbf{Exp}^1$ and $\mathbf{Exp}^2$, we should estimate the probability that the $\mathbf{Exp}^2$ aborts in line 12.

Let construct an adversary $\mathcal{C}$ that breaks the signum-relative collision resistant property of $\mathsf{H}$. The adversary $\mathcal{C}$ implements the $\mathbf{Exp}^2$ for $\mathcal{A}$. Note that he is able to do this as soon as we replace $\Pi$, $\Pi_R$, $\Pi_Q$ implementations with lazy sampling. $\mathcal{A}$ delivers $(l+1)$ forgeries to $\mathcal{C}$, and $\mathcal{C}$ finds the signum-

relative collision iff the condition in lines 11-12 is met.

Thus, we obtain the following bound:

$$\Pr\left[\mathbf{Exp}^1(\mathcal{A}) \Rightarrow 1\right] - \Pr\left[\mathbf{Exp}^2(\mathcal{A}) \Rightarrow 1\right] \leqslant \mathsf{Adv}_{\mathsf{H}}^{\mathsf{SCR}}(\mathcal{C}).$$

The adversary $\mathcal{C}$ implements $\mathbf{Exp}^2$ and thus processes at most $q_R$ queries to the oracle $rRO$, at most $q_Q$ queries to the oracle $qRO$, at most $q_{BRO} + 2q_{sign} + 1$ queries to the oracle $BRO$, at most $q_{BRO^{-1}}$ queries to the oracle $BRO^{-1}$ and at most $q_{sign}$ queries to the oracles $NewSign$, at most 1 query to the oracle $KGen$, checks the collision condition and verifies the forgeries obtained from $\mathcal{A}$. Adversary $\mathcal{C}$ uses at most $3T$ computational resources since it needs to simulate signing oracle (at most $q_{sign} \leqslant T$ queries) and check the forgeries ($(q_{sign} + 1)$ pairs).

**$\mathbf{Exp}^2(\mathcal{A})$**

1 : $(\Pi^O, \Pi^S) \leftarrow (\emptyset, \emptyset), \Pi \leftarrow \Pi^O \cup \Pi^S$
2 : $\Pi_R \leftarrow \emptyset$
3 : $\Pi_Q \leftarrow \emptyset$
4 : $l \leftarrow 0,\ SESS \leftarrow \emptyset$
5 : $sid \leftarrow -1$
6 : $round_{kg} \leftarrow 0,\ ctx_{kg} \leftarrow \emptyset$
7 : $p \leftarrow \mathcal{A}()$
8 : **if** $(p \neq 1 \wedge p \neq 2)$ : **return** $\perp$
9 : $(Q, d_p) \leftarrow (\varepsilon, \varepsilon)$
10 : $\{(m_i, \langle r_i, s_i\rangle)\}_{i=1}^{l+1} \xleftarrow{\$} \mathcal{A}^{KGen, NewSign, Sign, BRO, BRO^{-1}, rRO, qRO}(p)$
11 : $\forall i \neq j \in \{1, \ldots, l+1\}, m_i \neq m_j$ :
12 : **if** $(\mathrm{H}(m_i) = \pm\mathrm{H}(m_j))$ : **abort**
13 : **return** $((\forall i \neq j \in \{1, \ldots, l+1\}$ :
14 : $(m_i, \langle r_i, s_i\rangle) \neq (m_j, \langle r_j, s_j\rangle)) \wedge$
15 : $\wedge\ (\forall i \in \{1, \ldots, l+1\} : \mathsf{Verify}(Q, m_i, \langle r_i, s_i\rangle)))$

Figure 10: $\mathbf{Exp}^2(\mathcal{A})$ for the 2p-GOST scheme in the sOMUF-PCA model.

There are two cases in experiment $\mathbf{Exp}^2(\mathcal{A})$, depending on which $p$ value the adversary $\mathcal{A}$ chooses:

1. the party $\mathbf{P_2}$ is compromised, i.e. $p = 1$;
2. the party $\mathbf{P_1}$ is compromised, i.e. $p = 2$.

Thus,

$$\begin{aligned}\Pr\left[\mathbf{Exp}^2(\mathcal{A}) \to 1\right] &= \Pr\left[\mathbf{Exp}^2(\mathcal{A}) \to 1 | p = 1\right] \Pr[p = 1] + \\ &+ \Pr\left[\mathbf{Exp}^2(\mathcal{A}) \to 1 | p = 2\right] \Pr[p = 2] \leqslant \\ &\leqslant \max\left\{\Pr\left[\mathbf{Exp}^2(\mathcal{A}) \to 1 | p = 1\right], \Pr\left[\mathbf{Exp}^2(\mathcal{A}) \to 1 | p = 2\right]\right\}.\end{aligned}$$

Let's consider both of these cases separately.

**The party $\mathbf{P_2}$ is compromised.** Consider the $\mathbf{Exp}^2(\mathcal{A})$ under the assumption that $p = 1$. In the further experiments we change the $\mathsf{ExecKGen}^1$ and $\mathsf{ExecSign}^1$ functions behaviour only (see Figure 11).

The $\mathsf{ExecKGen}^1$ function in $\mathbf{Exp}^3$ is the modification of the $\mathsf{ExecKGen}^1$ function in $\mathbf{Exp}^2$ (same as in $\mathbf{Exp}^1$) by adding the abort condition in case of choosing $op_Q$ that already belongs to set $\Pi_Q$ (line 9). Note that on round 0 we only select $comm_Q$ uniformly without querying random oracle $qRO$. We fix the values $Q_1$ and $op_Q$ on the round 1 and verify if $op_Q$ belongs to set $\Pi_Q$ or not. Thus, we preserve the ability of the adversary to receive $comm_Q$ and conduct an exhaustive search using the random oracle $qRO$.

We should estimate the probability of this event to estimate the difference between the $\mathbf{Exp}^2$ and $\mathbf{Exp}^3$. The value $op_Q$ is uniformly distributed in a set $\{0,1\}^\kappa$ of cardinality $2^\kappa$. In the worst case the adversary $\mathcal{A}$ has already made all queries to the $qRO$ oracle and thus $\Pi_Q$ contains at least $q_Q$ elements. The abort condition is met if the value $op_Q$ hits one of elements in $\Pi_Q$. We can estimate this probability as $\frac{q_Q}{2^\kappa}$. As oracle $KGen$ is executed once, the overall probability can be bounded by $\frac{q_Q}{2^\kappa}$.

Similarly, the $\mathsf{ExecSign}^1$ function in $\mathbf{Exp}^3$ is the modification of the $\mathsf{ExecSign}^1$ function in $\mathbf{Exp}^2$ (same as in $\mathbf{Exp}^1$) by adding the abort condition in case of choosing $op_R$ that already belongs to set $\Pi_R$ (line 12). Note that on round 0 we only select $comm_R$ uniformly without using random oracle $rRO$. We fix the values $R_1$ and $op_R$ on the round 1 and verify if $op_R$ belongs to set $\Pi_Q$ or not. Thus, we preserve the ability of the adversary to receive $comm_R$ and conduct an exhaustive search using the random oracle $rRO$.

We should estimate the probability of this event to estimate the difference between the $\mathbf{Exp}^2$ and $\mathbf{Exp}^3$. The value $op_R$ is uniformly distributed in a set $\{0,1\}^\kappa$ of cardinality $2^\kappa$. In the worst case the adversary $\mathcal{A}$ has already made all queries to the $rRO$ and $Sign$ oracles and thus $\Pi_R$ contains at least $q_R + q_{sign}$ elements. The abort condition is met if the value $op_R$ hits one of

elements in $\Pi_R$. We can estimate this probability as $\dfrac{q_R + q_{sign}}{2^\kappa}$. As oracle $Sign$ is executed at most $q_{sign}$ times, the overall probability can be bounded by $q_{sign} \cdot \dfrac{q_R + q_{sign}}{2^\kappa}$.

Thus,

$$\Pr\left[\mathbf{Exp}^2(\mathcal{A}) \to 1\right] - \Pr\left[\mathbf{Exp}^3(\mathcal{A}) \to 1\right] \leqslant \frac{q_Q}{2^\kappa} + q_{sign} \cdot \frac{q_R + q_{sign}}{2^\kappa}.$$

$\mathsf{ExecKGen}^1(msg)$ $(\mathbf{Exp}^3)$

1: **if** $(round_{kg} = 0)$ :
2: $comm_Q \xleftarrow{\mathcal{U}} \{0,1\}^n$
3: $msg' \leftarrow \{comm_Q\}$
4: **else if** $(round_{kg} = 1)$ :
5: $Q_2 \leftarrow msg$
6: $d_1 \xleftarrow{\mathcal{U}} \mathbb{Z}_q^*$
7: $Q_1 \leftarrow d_1 P$
8: $op_Q \xleftarrow{\mathcal{U}} \{0,1\}^\kappa$
9: **if** $((op_Q, \cdot, \cdot) \in \Pi_Q)$ : **abort**
10: $\Pi_Q \leftarrow \Pi_Q \cup \{op_Q, Q_1, comm_Q\}$
11: **if** $(Q_2 = -Q_1)$ : **return** $\perp$
12: $Q \leftarrow Q_1 + Q_2$
13: $msg' \leftarrow \{op_Q, Q_1\}$
14: **else** :
15: $msg' \leftarrow \varepsilon$
16: $round_{kg} \leftarrow round_{kg} + 1$
17: // Update the $ctx_{kg}$ value
18: **return** $msg'$

$\mathsf{ExecSign}^1(state, msg)$ $(\mathbf{Exp}^3)$

1: $round \leftarrow state.ctx.round$
2: **if** $(round = 0)$ :
3: $e \leftarrow \mathrm{H}(state.m)$
4: **if** $(e = 0) : e \leftarrow 1$
5: $comm_R \xleftarrow{\mathcal{U}} \{0,1\}^l$
6: $msg' \leftarrow \{comm_R\}$
7: **else if** $(round = 1)$ :
8: $R_2 \leftarrow msg$
9: $k_1 \xleftarrow{\mathcal{U}} \mathbb{Z}_q$
10: $R_1 \leftarrow k_1 P$
11: $op_R \xleftarrow{\mathcal{U}} \{0,1\}^\kappa$
12: **if** $((op_R, \cdot, \cdot) \in \Pi_R)$ : **abort**
13: $\Pi_R \leftarrow \Pi_R \cup \{op_R, R_1, comm_R\}$
14: **if** $(R_2 = -R_1)$ :
15: **return** $(state, \perp)$
16: $R \leftarrow R_1 + R_2$
17: **if** $((\phi(R), \cdot) \in \Pi)$ :
18: $r \leftarrow \psi(\Pi(\phi(R)))$
19: **else** :
20: $\beta \xleftarrow{\mathcal{U}} \{0, \ldots, 2^N - 1\}$
21: **if** $((\cdot, \beta) \in \Pi)$ : **abort**
22: $\Pi^S \leftarrow \Pi^S \cup \{(\phi(R), \beta)\}$
23: $\Pi \leftarrow \Pi^S \cup \Pi^O$
24: $r \leftarrow \psi(\beta)$
25: **if** $(r = 0)$ : **return** $(state, \perp)$
26: $s_1 \leftarrow k_1 \cdot e + d_1 \cdot r$
27: $state.flag \leftarrow 1$
28: $msg' \leftarrow \{op_R, R_1, s_1\}$
29: **else if** $(round = 2)$ :
30: $s_2 \leftarrow msg$
31: $s \leftarrow s_1 + s_2$
32: $msg' \leftarrow \varepsilon$
33: **if** $(\mathsf{Verify}(Q, state.m, \langle r, s \rangle) = 0)$ :
34: **return** $(state, \perp)$
35: **else**
36: $msg' \leftarrow \varepsilon$
37: // Update the $state.ctx$ value
38: **return** $(state, msg')$

$\mathsf{ExecSign}^1(state, msg)$ $(\mathbf{Exp}^4)$

1: $round \leftarrow state.ctx.round$
2: **if** $(round = 0)$ :
3: $e \leftarrow \mathrm{H}(state.m)$
4: **if** $(e = 0) : e \leftarrow 1$
5: $comm_R \xleftarrow{\mathcal{U}} \{0,1\}^l$
6: $msg' \leftarrow \{comm_R\}$
7: **else if** $(round = 1)$ :
8: $R_2 \leftarrow msg$
9: $\beta \xleftarrow{\mathcal{U}} \{0, \ldots, 2^N - 1\}$
10: $r \leftarrow \psi(\beta)$
11: $s_1 \xleftarrow{\mathcal{U}} \mathbb{Z}_q$
12: $R_1 \leftarrow e^{-1} s_1 P - e^{-1} r Q_1$
13: $op_R \xleftarrow{\mathcal{U}} \{0,1\}^\kappa$
14: **if** $((op_R, \cdot, \cdot) \in \Pi_R)$ : **abort**
15: $\Pi_R \leftarrow \Pi_R \cup \{op_R, R_1, comm_R\}$
16: **if** $(R_2 = -R_1)$ :
17: **return** $(state, \perp)$
18: $R \leftarrow R_1 + R_2$
19: **if** $(r = 0)$ : **return** $(state, \perp)$
20: **if** $((\phi(R), \cdot) \in \Pi)$ : **abort**
21: **if** $((\cdot, \beta) \in \Pi)$ : **abort**
22: $\Pi^S \leftarrow \Pi^S \cup \{(\phi(R), \beta)\}$
23: $\Pi \leftarrow \Pi^S \cup \Pi^O$
24: $state.flag \leftarrow 1$
25: $msg' \leftarrow\!\leftarrow \{op_R, R_1, s_1\}$
26: **else if** $(round = 2)$ :
27: $s_2 \leftarrow msg$
28: $s \leftarrow s_1 + s_2$
29: $msg' \leftarrow \varepsilon$
30: **if** $(\mathsf{Verify}(Q, state.m, \langle r, s \rangle) = 0)$ :
31: **return** $(state, \perp)$
32: **else**
33: $msg' \leftarrow \varepsilon$
34: // Update the $state.ctx$ value
35: **return** $(state', msg')$

Figure 11: The $\mathsf{ExecKGen}^1$ and $\mathsf{ExecSign}^1$ functions in $\mathbf{Exp}^3(\mathcal{A})$, $\mathbf{Exp}^4(\mathcal{A})$

The signing oracle in the $\mathbf{Exp}^4$ gets along with only public information. Values $\beta$ and $s_1$ are randomly chosen from the relevant sets and then point

$R_1$ is constructed. We define the corresponding pair in $\Pi$ implementation by saving this pair in the $\Pi^S$ set. Note that if we couldn't do so (i.e., $\beta$ already belongs to the $\Pi$), the abort condition is met like in the $\mathbf{Exp}^3$.

Consider the distribution on $R_1$ and $s_1$, that are returned by the $Sign$ oracle on the round 1. In the $\mathbf{Exp}^3$ value $k_1$ is distributed uniformly on $\mathbb{Z}_q$, thus $R_1$ is uniformly distributed. The value $r$ is independent on $k_1$ (due to bijective random oracle) and thus $s_1$ value is also uniformly distributed on $\mathbb{Z}_q$.

In the $\mathbf{Exp}^4$ values $R_1$ and $s_1$ are also distributed uniformly on the corresponding sets except of the values that lead to $\phi(R)$ that already belongs to $\Pi$. Let's estimate the probability of these «bad» event. The values $s_1$ and $r$ are uniformly distributed on a set $\mathbb{Z}_q$ and are chosen independently. Then the value $R_1$ (and thus $R$) is uniformly distributed on a set of cardinality $q$. In the worst case the adversary $\mathcal{A}$ has already made all queries to the $BRO$, $BRO^{-1}$, $Sign$ oracles and thus $\Pi$ contains at least $(q_{BRO} + q_{BRO^{-1}} + 2q_{sign})$ elements. Note that here we do not take into account the queries to the $BRO$ oracle made during finalizing the experiment (verifying the forgeries), since they are made after all queries to the $Sign$ oracle. The abort condition is met if the value $\phi(R)$ hits one of elements in $\Pi$. We can estimate this probability as $\dfrac{q_{BRO} + q_{BRO^{-1}} + 2q_{sign}}{q}$. As oracle $Sign$ is executed at most $q_{sign}$ times, the overall probability can be bounded by $q_{sign} \cdot \dfrac{q_{BRO} + q_{BRO^{-1}} + 2q_{sign}}{q}$.

Thus, we conclude that

$$\Pr\left[\mathbf{Exp}^3(\mathcal{A}) \to 1\right] - \Pr\left[\mathbf{Exp}^4(\mathcal{A}) \to 1\right] \leqslant q_{sign} \cdot \frac{q_{BRO} + q_{BRO^{-1}} + 2q_{sign}}{q}.$$

Let construct the adversary $\mathcal{B}$ for the GOST scheme in the sUF-KO model that uses $\mathcal{A}$ as the black box (see Figure 12).

$\mathcal{B}^{BRO^*,BRO^{*-1}}(Q,\mathcal{A})$

$(\Pi^O,\Pi^S) \leftarrow (\emptyset,\emptyset), \Pi \leftarrow \Pi^O \cup \Pi^S$
$\Pi_R \leftarrow \emptyset$
$\Pi_Q \leftarrow \emptyset$
$l \leftarrow 0,\ SESS \leftarrow \emptyset$
$sid \leftarrow -1$
$round_{kg} \leftarrow 0,\ ctx_{kg} \leftarrow \emptyset$
$p \leftarrow \mathcal{A}()$
**if** $(p \neq 1 \wedge p \neq 2)$ : **return** $\perp$
$\{(m_i, \langle r_i, s_i\rangle)\}_{i=1}^{l+1} \xleftarrow{\$} \mathcal{A}^{KGen,NewSign,Sign,BRO,BRO^{-1},rRO,qRO}(p)$
$\forall i \neq j \in \{1,\ldots,l+1\}, m_i \neq m_j$ :
  **if** $(\mathrm{H}(m_i) = \pm\mathrm{H}(m_j))$ : **abort**
**if** $((\exists i \neq j \in \{1,\ldots,l+1\} : (m_i, \langle r_i, s_i\rangle) = (m_j, \langle r_j, s_j\rangle))$ :
  **abort**
**for** $i \in \{1,\ldots,l+1\}$ :
  $e_i \leftarrow \mathrm{H}(m_i)$
  **if** $(e_i = 0) : e_i \leftarrow 1$
  $R_i \leftarrow e_i^{-1}(s_iP - r_iQ)$
  **if** $\psi(SimBRO(\phi(R_i))) \neq r_i$ : **abort**
  **if** $(\phi(R_i), \cdot) \in \Pi^O$ : **return** $(m_i, \langle r_i, s_i\rangle)$
Find $i,j : ((\phi(R_i), \cdot) \in \Pi^S) \wedge (\phi(R_i) = \phi(R_j))$
Compute $d$
$(m, \langle r, s\rangle) \xleftarrow{\$} \mathsf{GOST.Sign}(d, m)$
**return** $(m, \langle r, s\rangle)$

Oracle $SimBRO(\alpha)$

**if** $(\alpha, \cdot) \in \Pi$ : **return** $\Pi(\alpha)$
$\beta \leftarrow BRO^*(\alpha)$
**if** $((\cdot\beta) \in \Pi)$ : **abort**
$\Pi^O \leftarrow \Pi^O \cup \{(\alpha, \beta)\}$
$\Pi \leftarrow \Pi^O \cup \Pi^S$
**return** $\beta$

Oracle $SimBRO^{-1}(\beta)$

**if** $(\cdot, \beta) \in \Pi$ : **return** $\Pi^{-1}(\beta)$
$\alpha \leftarrow BRO^{*-1}(\beta)$
**if** $((\alpha, \cdot) \in \Pi)$ : **abort**
$\Pi^O \leftarrow \Pi^O \cup \{(\alpha, \beta)\}$
$\Pi \leftarrow \Pi^O \cup \Pi^S$
**return** $\beta$

$\mathsf{ExecKGen}^1(msg)$

**if** $(round_{kg} = 0)$ :
  $comm_R \xleftarrow{\mathcal{U}} \{0,1\}^n$
  $msg' \leftarrow \{comm_Q\}$
**else if** $(round_{kg} = 1)$ :
  $Q_2 \leftarrow msg$
  $Q_1 \leftarrow Q - Q_2$
  $op_Q \xleftarrow{\mathcal{U}} \{0,1\}^\kappa$
  **if** $((op_Q, \cdot, \cdot) \in \Pi_Q)$ : **abort**
  $\Pi_Q \leftarrow \Pi_Q \cup \{op_Q, Q_1, comm_Q\}$
  **if** $(Q_2 = -Q_1)$ : **return** $\perp$
  $Q \leftarrow Q_1 + Q_2$
  $msg' \leftarrow \{op_Q, Q_1\}$
**else** :
  $msg' \leftarrow \varepsilon$
$round_{kg} \leftarrow round_{kg} + 1$
// Update the $ctx_{kg}$ value
**return** $msg'$

Figure 12: The adversary $\mathcal{B}$ for the GOST scheme in the sUF-KO model that uses $\mathcal{A}$ as the black box

Adversary $\mathcal{B}$ simulates the $rRO$, $qRO$, $NewSign$ and $Sign$ oracles to answer the $\mathcal{A}$ queries as the corresponding oracles in the $\mathbf{Exp}^4$. Adversary $\mathcal{B}$ simulates the $BRO$, $BRO^{-1}$ oracles by translating the queries to its own oracle (see $SimBRO$ and $SimBRO^{-1}$). Adversary $\mathcal{B}$ simulates $KGen$ oracle similar to the oracle $KGen$ in the $\mathbf{Exp}^4$ with the modification of the $\mathsf{ExecKGen}^1$ function. Adversary $\mathcal{B}$ sets $Q_1$ value (after receiving $Q_2$) in such a way that the resulting public key iis equal to the public key $Q$, provided by its challenger.

After receiving $l+1$ forgeries from $\mathcal{A}$, $\mathcal{B}$ finds the suitable forgery relative to its own challenger. Assume that $\mathcal{A}$ delivers valid pairs $\{(m_i, \langle r_i, s_i\rangle)\}_{i=1}^{l+1}$. This means that the set $\Pi$ contains all corresponding pairs $(\phi(R_i), \beta), i = 1, \ldots, l+1$: either these pairs were already in the $\Pi$ before verification check in line 18 or were saved after $SimBRO$ call during this check. There are two possible cases. If there exists at least one pair $(\phi(R_i), \beta) \in \Pi^O$, then it is already a valid forgery with respect to the oracles $BRO^*$, $BRO^{*-1}$ and $\mathcal{B}$ can simply forward it to its own challenger. If all pairs $(\phi(R_i), \beta) \in \Pi^S, i = 1, \ldots, l+1$, $\mathcal{B}$ can recover the signing key $d$ as described below and construct the new forgery for an arbitrary message.

Note that there are at least $l$ pairs in $\Pi^S$, because adding a pair to the set $\Pi^S$ is performed only during the $Sign$ oracle execution simultaneously with incrementing the counter $l$ of successful sessions. Thus, if pairs $(\phi(R_i), \beta) \in \Pi^S, i = 1, \ldots, l+1$, then there are two of them with indexes $i, j$ such as $\phi(R_i) = \phi(R_j)$. This means that $r_i = r_j = \psi(\beta) = r$ in the corresponding forgeries. The adversary $\mathcal{B}$ knows the corresponding $e_i = \mathrm{H}(m_i)$, $e_j = \mathrm{H}(m_j)$.

The equations $\phi(R_i) = \phi(R_j)$ implies $R_i = \pm R_j$ and thus $k_i = \pm k_j = k$. So the following linear equation system holds:

$$\begin{cases} s_i &= ke_i + dr; \\ s_j &= \pm ke_j + dr; \end{cases}$$

There are two unknown variables $k$ and $d$ in the system above. This system has a unique solution whenever $e_i \neq \pm e_j$. Observe that case $e_i = \pm e_j$ and thus $\mathrm{H}(m_i) = \pm\mathrm{H}(m_j)$ is excluded by lines 10, 11, if $m_i \neq m_j$. The $m_i = m_j$ condition (together with $r_i = r_j$ condition) implies either $(m_i, \langle r_i, s_i\rangle) = (m_j, \langle r_j, s_j\rangle)$, that is excluded by lines 12, 13, or $k_i = -k_j$, that still allows to compute $d$ from the system equation. Summing all, we can always compute $d$ if all pairs $(\phi(R_i), \beta), i = 1, \ldots, l+1$, belongs to $\Pi^S$.

We conclude that if $\mathcal{A}$ delivers valid $l+1$ forgeries, $\mathcal{B}$ delivers a valid forgery to its own challenger and

$$\Pr\left[\mathbf{Exp}^4(\mathcal{A}) \to 1\right] = \Pr\left[\mathbf{Exp}^{\mathsf{sUF\text{-}KO}}_{\mathsf{GOST}}(\mathcal{B}) \to 1\right].$$

Note that the number of queries made by $\mathcal{B}$ to the $BRO^*$ and $BRO^{*-1}$ oracles is at most $q_{BRO} + 2q_{sign} + 1$ and $q_{BRO^{-1}}$ respectively. The adversary $\mathcal{B}$ needs the same amount of computational resources as $\mathcal{C}$.

Thus, we summarize the obtained bounds in case the party $P_2$ is compro-

mised:

$$\begin{aligned}\Pr\left[\mathbf{Exp}^2(\mathcal{A}) \to 1\right] &= \left(\Pr\left[\mathbf{Exp}^2(\mathcal{A}) \to 1\right] - \Pr\left[\mathbf{Exp}^3(\mathcal{A}) \to 1\right]\right) + \\ &+ \left(\Pr\left[\mathbf{Exp}^3(\mathcal{A}) \to 1\right] - \Pr\left[\mathbf{Exp}^4(\mathcal{A}) \to 1\right]\right) + \Pr\left[\mathbf{Exp}^4(\mathcal{A}) \to 1\right] \leqslant \\ &\leqslant \frac{q_Q}{2^\kappa} + q_{sign} \cdot \frac{q_R + q_{sign}}{2^\kappa} + q_{sign} \cdot \frac{q_{BRO} + q_{BRO^{-1}} + 2q_{sign}}{q} + \\ &+ \Pr\left[\mathbf{Exp}^{\mathsf{sUF\text{-}KO}}_{\mathsf{GOST}}(\mathcal{B}) \to 1\right] = \frac{q_Q}{2^\kappa} + q_{sign} \cdot \frac{q_R + q_{sign}}{2^\kappa} + \\ &+ q_{sign} \cdot \frac{q_{BRO} + q_{BRO^{-1}} + 2q_{sign}}{q} + \mathsf{Adv}^{\mathsf{sUF\text{-}KO}}_{\mathsf{GOST}}(\mathcal{B}).\end{aligned}$$

**The party $\mathbf{P_1}$ is compromised.** Consider the $\mathbf{Exp}^2$ under the assumption that $p = 2$. In the further experiments we change the $\mathsf{ExecKGen}^2$ and $\mathsf{ExecSign}^2$ functions behaviour only (see Figure 13).

The $\mathsf{ExecKGen}^2$ function in $\mathbf{Exp}^3$ is the modification of the $\mathsf{ExecKGen}^2$ function in $\mathbf{Exp}^2$ (same as in $\mathbf{Exp}^1$) by adding the abort condition in case of receiving $comm_Q$ that does not belong to set $\Pi_Q$ (lines 4, 5, 12). Note that on round 0 we only set $flag_Q$ and abort on the round 1. Thus, we preserve the ability of the adversary to receive $Q_2$.

We should estimate the probability of this event to estimate the difference between the $\mathbf{Exp}^2$ and $\mathbf{Exp}^3$. The value $comm_Q$ belongs to the set $\{0,1\}^n$ of cardinality $2^n$. In the worst case the adversary $\mathcal{A}$ has already made all queries to the $qRO$ oracle and thus $\Pi_Q$ contains at least $q_Q$ elements. The abort condition is met if the value $comm_Q$ hits one of elements in $\Pi_Q$. We can estimate this probability as $\frac{q_Q}{2^n}$. As oracle $KGen$ is executed only once, the overall probability can be bounded by $\frac{q_Q}{2^n}$.

Similarly, the $\mathsf{ExecSign}^2$ function in $\mathbf{Exp}^3$ is the modification of the $\mathsf{ExecSign}^2$ function in $\mathbf{Exp}^2$ (same as in $\mathbf{Exp}^1$) by adding the abort condition in case of receiving $comm_R$ that does not belong to set $\Pi_R$ (lines 7, 8, 16). Note that on round 0 we only set $flag_R$ and abort on the round 1. Thus, we preserve the ability of the adversary to receive $R_2$.

We should estimate the probability of this event to estimate the difference between the $\mathbf{Exp}^2$ and $\mathbf{Exp}^3$. The value $comm_R$ belongs to the set $\{0,1\}^n$ of cardinality $2^n$. In the worst case the adversary $\mathcal{A}$ has already made all queries to the $rRO$ and $Sign$ oracles and thus $\Pi_R$ contains at least $q_R{+}q_{sign}$ elements. The abort condition is met if the value $comm_R$ hits one of elements in $\Pi_R$. We can estimate this probability as $\frac{q_R + q_{sign}}{2^n}$. As oracle $Sign$ is executed at

most $q_{sign}$ times, the overall probability can be bounded by $q_{sign} \cdot \dfrac{q_R + q_{sign}}{2^n}$.

Thus,

$$\Pr\left[\mathbf{Exp}^2(\mathcal{A}) \to 1\right] - \Pr\left[\mathbf{Exp}^3(\mathcal{A}) \to 1\right] \leqslant \frac{q_Q}{2^n} + q_{sign} \cdot \frac{q_R + q_{sign}}{2^n}.$$

$\mathsf{ExecKGen}^2(msg)$ $(\mathbf{Exp}^3)$

1 : **if** $(round_{kg} = 0)$ :
2 : $\quad flag_Q \leftarrow 0$
3 : $\quad comm_Q \leftarrow msg$
4 : $\quad$ **if** $((\cdot, \cdot, comm_Q) \notin \Pi_Q)$ :
5 : $\quad\quad flag_Q \leftarrow 1$
6 : $\quad d_2 \xleftarrow{\mathcal{U}} \mathbb{Z}_q^*$
7 : $\quad Q_2 \leftarrow d_2 P$
8 : $\quad msg' \leftarrow \{Q_2\}$
9 : **else if** $(round_{kg} = 1)$ :
10 : $\quad op_Q, Q_1 \leftarrow msg$
11 : $\quad$ **if** $(comm_Q \neq qRO(op_Q, Q_1))$ : **return** $\perp$
12 : $\quad$ **if** $flag_Q$ : **abort**
13 : $\quad$ **if** $(Q_2 = -Q_1)$ : **return** $\perp$
14 : $\quad Q \leftarrow Q_1 + Q_2$
15 : $\quad msg' \leftarrow \varepsilon$
16 : **else** :
17 : $\quad msg' \leftarrow \varepsilon$
18 : $round_{kg} \leftarrow round_{kg} + 1$
19 : // Update the $ctx_{kg}$ value
20 : **return** $msg'$

$\mathsf{ExecSign}^2(state, msg)$ $(\mathbf{Exp}^3)$

1 : $round \leftarrow state.ctx.round$
2 : **if** $(round = 0)$ :
3 : $\quad flag_R \leftarrow 0$
4 : $\quad e \leftarrow \mathrm{H}(state.m)$
5 : $\quad$ **if** $(e = 0) : e \leftarrow 1$
6 : $\quad comm_R \leftarrow msg$
7 : $\quad$ **if** $((\cdot, \cdot, comm_R) \notin \Pi_R)$ :
8 : $\quad\quad flag_R \leftarrow 1$
9 : $\quad k_2 \xleftarrow{\mathcal{U}} \mathbb{Z}_q$
10 : $\quad R_2 \leftarrow k_2 P$
11 : $\quad msg' \leftarrow \{R_2\}$
12 : **else if** $(round = 1)$ :
13 : $\quad (op_R, R_1, s_1) \leftarrow msg$
14 : $\quad$ **if** $(comm_R \neq rRO(op_R, R_1))$ :
15 : $\quad\quad$ **return** $(state, \perp)$
16 : $\quad$ **if** $flag_R$ : **abort**
17 : $\quad$ **if** $(R_1 = -R_2)$ :
18 : $\quad\quad$ **return** $(state, \perp)$
19 : $\quad R \leftarrow R_1 + R_2$
20 : $\quad$ **if** $((\phi(R), \cdot) \in \Pi)$ :
21 : $\quad\quad r \leftarrow \psi(\Pi(\phi(R)))$
22 : $\quad$ **else**
23 : $\quad\quad \beta \xleftarrow{\mathcal{U}} \{0, \ldots, 2^N - 1\}$
24 : $\quad\quad$ **if** $((\cdot, \beta) \in \Pi)$ : **abort**
25 : $\quad\quad \Pi^S \leftarrow \Pi^S \cup \{(\phi(R), \beta)\}$
26 : $\quad\quad \Pi \leftarrow \Pi^S \cup \Pi^O$
27 : $\quad\quad r \leftarrow \psi(\beta)$
28 : $\quad s_2 \leftarrow k_2 \cdot e + d_2 \cdot r$
29 : $\quad s \leftarrow s_1 + s_2$
30 : $\quad state.flag \leftarrow 1$
31 : $\quad msg' \leftarrow \{s_2\}$
32 : $\quad$ **if** $(\mathsf{Verify}(Q, state.m, \langle r, s \rangle) = 0)$ :
33 : $\quad\quad$ **return** $(state, \perp)$
34 : **else** :
35 : $\quad msg' \leftarrow \varepsilon$
36 : // Update the $state.ctx$ value
37 : **return** $(state, msg')$

$\mathsf{ExecSign}^2(state, msg)$ $(\mathbf{Exp}^4)$

1 : $round \leftarrow state.ctx.round$
2 : **if** $(round = 0)$ :
3 : $\quad flag_R \leftarrow 0$
4 : $\quad e \leftarrow \mathrm{H}(state.m)$
5 : $\quad$ **if** $(e = 0) : e \leftarrow 1$
6 : $\quad comm_R \leftarrow msg$
7 : $\quad$ **if** $((\cdot, \cdot, comm_R) \notin \Pi_R)$ :
8 : $\quad\quad flag_R \leftarrow 1$
9 : $\quad \beta \xleftarrow{\mathcal{U}} \{0, \ldots, 2^N - 1\}$
10 : $\quad r \leftarrow \psi(\beta)$
11 : $\quad s_2 \xleftarrow{\mathcal{U}} \mathbb{Z}_q$
12 : $\quad R_2 \leftarrow e^{-1} s_2 P - e^{-1} r Q_2$
13 : $\quad msg' \leftarrow \{R_2\}$
14 : **else if** $(round = 1)$ :
15 : $\quad (op_1, R_1, s_1) \leftarrow msg$
16 : $\quad$ **if** $(comm_R \neq rRO(op_R, R_1))$ :
17 : $\quad\quad$ **return** $(state, \perp)$
18 : $\quad$ **if** $flag_R$ : **abort**
19 : $\quad$ **if** $(R_1 = -R_2)$ :
20 : $\quad\quad$ **return** $(state, \perp)$
21 : $\quad R \leftarrow R_1 + R_2$
22 : $\quad$ **if** $((\phi(R), \cdot) \in \Pi)$ : **abort**
23 : $\quad$ **if** $((\cdot, \beta) \in \Pi)$ : **abort**
24 : $\quad \Pi^S \leftarrow \Pi^S \cup \{(\phi(R), \beta)\}$
25 : $\quad \Pi \leftarrow \Pi^S \cup \Pi^O$
26 : $\quad s \leftarrow s_1 + s_2$
27 : $\quad state.flag \leftarrow 1$
28 : $\quad msg' \leftarrow \{s_2\}$
29 : $\quad$ **if** $(\mathsf{Verify}(Q, state.m, \langle r, s \rangle) = 0)$ :
30 : $\quad\quad$ **return** $(state, \perp)$
31 : **else** :
32 : $\quad msg' \leftarrow \varepsilon$
33 : // Update the $state.ctx$ value
34 : **return** $(state, msg')$

Figure 13: The $\mathsf{ExecKGen}^2$ and $\mathsf{ExecSign}^2$ functions in $\mathbf{Exp}^3(\mathcal{A})$, $\mathbf{Exp}^4(\mathcal{A})$

The signature oracle in the $\mathbf{Exp}^4$ gets along with only public information. Values $\beta$ and $s_2$ are randomly chosen from the relevant sets and then point $R_2$ is constructed. We define the corresponding pair in $\Pi$ implementation by

saving this pair in the $\Pi^S$ set. Note that if we couldn't do so (i.e., $\beta$ already belongs to the $\Pi$), the abort condition is met like in the $\mathbf{Exp}^3$.

Consider the distribution on $R_2$ and $s_2$, that are returned by the $Sign$ oracle on the rounds 1 and 2 respectively. In the $\mathbf{Exp}^3$ value $k_2$ is distributed uniformly on $\mathbb{Z}_q$, thus $R_2$ is uniformly distributed. The value $r$ is independent on $k_2$ (due to bijective random oracle) and thus $s_2$ value is also uniformly distributed on $\mathbb{Z}_q$.

In the $\mathbf{Exp}^4$ values $R_2$ and $s_2$ are also distributed uniformly on the corresponding sets except of the values that lead to $\phi(R)$ that already belongs to $\Pi$. Let's estimate the probability of these «bad» event. The values $s_2$ and $r$ are uniformly distributed on a set $\mathbb{Z}_q$ and are chosen independently. Then the value $R_2$ (and thus $R$) is uniformly distributed on a set of cardinality $q$. In the worst case the adversary $\mathcal{A}$ has already made all queries to the $BRO$, $BRO^{-1}$, $Sign$ oracles and thus $\Pi$ contains at least $(q_{BRO} + q_{BRO^{-1}} + 2q_{sign})$ elements. Note that here we do not take into account the queries to the $BRO$ oracle made during finalizing the experiment (verifying the forgeries), since they are made after all queries to the $Sign$ oracle. The abort condition is met if the value $\phi(R)$ hits one of elements in $\Pi$. We can estimate this probability as $\dfrac{q_{BRO} + q_{BRO^{-1}} + 2q_{sign}}{q}$. As oracle $Sign$ is executed at most $q_{sign}$ times, the overall probability can be bounded by $q_{sign} \cdot \dfrac{q_{BRO} + q_{BRO^{-1}} + 2q_{sign}}{q}$.

Thus, we conclude that

$$\Pr\left[\mathbf{Exp}^3(\mathcal{A}) \to 1\right] - \Pr\left[\mathbf{Exp}^4(\mathcal{A}) \to 1\right] \leqslant q_{sign} \cdot \frac{q_{BRO} + q_{BRO^{-1}} + 2q_{sign}}{q}.$$

Let construct the adversary $\mathcal{B}$ for the GOST scheme in the sUF-KO model that uses $\mathcal{A}$ as the black box (see Figure 14).

$\mathcal{B}^{BRO^*, BRO^{*-1}}(Q, \mathcal{A})$

$(\Pi^O, \Pi^S) \leftarrow (\emptyset, \emptyset), \Pi \leftarrow \Pi^O \cup \Pi^S$
$\Pi_R \leftarrow \emptyset$
$\Pi_Q \leftarrow \emptyset$
$l \leftarrow 0,\ SESS \leftarrow \emptyset$
$sid \leftarrow -1$
$round_{kg} \leftarrow 0,\ ctx_{kg} \leftarrow \emptyset$
$p \leftarrow \mathcal{A}()$
**if** $(p \neq 1 \wedge p \neq 2)$ : **return** $\perp$
$\{(m_i, \langle r_i, s_i \rangle)\}_{i=1}^{l+1} \xleftarrow{\$} \mathcal{A}^{KGen, NewSign, Sign, BRO, BRO^{-1}, rRO, qRO}(p)$
$\forall i \neq j \in \{1, \ldots, l+1\}, m_i \neq m_j$ :
  **if** $(\mathrm{H}(m_i) = \pm \mathrm{H}(m_j))$ : **abort**
**if** $((\exists i \neq j \in \{1, \ldots, l+1\} : (m_i, \langle r_i, s_i \rangle) = (m_j, \langle r_j, s_j \rangle)))$ :
  **abort**
**for** $i \in \{1, \ldots, l+1\}$ :
  $e_i \leftarrow \mathrm{H}(m_i)$
  **if** $(e_i = 0) : e_i \leftarrow 1$
  $R_i \leftarrow e_i^{-1}(s_i P - r_i Q)$
  **if** $\psi(SimBRO(\phi(R_i))) \neq r_i$ : **abort**
  **if** $(\phi(R_i), \cdot) \in \Pi^O$ : **return** $(m_i, \langle r_i, s_i \rangle)$
Find $i, j : ((\phi(R_i), \cdot) \in \Pi^S) \wedge (\phi(R_i) = \phi(R_j))$
Compute $d$
$(m, \langle r, s \rangle) \xleftarrow{\$} \mathsf{GOST.Sign}(d, m)$
**return** $(m, \langle r, s \rangle)$

Oracle $SimBRO(\alpha)$

**if** $(\alpha, \cdot) \in \Pi$ : **return** $\Pi(\alpha)$
$\beta \leftarrow BRO^*(\alpha)$
**if** $((\cdot\beta) \in \Pi)$ : **abort**
$\Pi^O \leftarrow \Pi^O \cup \{(\alpha, \beta)\}$
$\Pi \leftarrow \Pi^O \cup \Pi^S$
**return** $\beta$

Oracle $SimBRO^{-1}(\beta)$

**if** $(\cdot, \beta) \in \Pi$ : **return** $\Pi^{-1}(\beta)$
$\alpha \leftarrow BRO^{*-1}(\beta)$
**if** $((\alpha, \cdot) \in \Pi)$ : **abort**
$\Pi^O \leftarrow \Pi^O \cup \{(\alpha, \beta)\}$
$\Pi \leftarrow \Pi^O \cup \Pi^S$
**return** $\beta$

$\mathsf{ExecKGen}^2(msg)$

**if** $(round_{kg} = 0)$ :
  $flag_Q \leftarrow 0$
  $comm_Q \leftarrow msg$
  **if** $((\cdot, \cdot, comm_Q) \notin \Pi_Q)$ :
    $flag_Q \leftarrow 1$
    $d_2 \xleftarrow{\mathcal{U}} \mathbb{Z}_q$
    $Q_2 \leftarrow d_2 P$
**else**
    $Q_1 \leftarrow \Pi_Q[comm_Q]$
    $Q_2 \leftarrow Q - Q_1$
  $msg' \leftarrow \{Q_2\}$
  **else if** $(round_{kg} = 1)$ :
  $op_Q, Q_1 \leftarrow msg$
  **if** $(comm_Q \neq qRO(op_Q, Q_1))$ : **return** $\perp$
  **if** $flag_Q$ : **abort**
  **if** $(Q_2 = -Q_1)$ : **return** $\perp$
  $Q \leftarrow Q_1 + Q_2$
  $msg' \leftarrow \varepsilon$
**else** :
  $msg' \leftarrow \varepsilon$
$round_{kg} \leftarrow round_{kg} + 1$
// Update the $ctx_{kg}$ value
**return** $msg'$

Figure 14: The adversary $\mathcal{B}$ for the GOST scheme in the sUF-KO model that uses $\mathcal{A}$ as the black box

Adversary $\mathcal{B}$ simulates the $rRO$, $qRO$, $NewSign$ and $Sign$ oracles to answer the $\mathcal{A}$ queries as the corresponding oracles in the $\mathbf{Exp}^4$. Adversary $\mathcal{B}$ simulates the $BRO$, $BRO^{-1}$ oracles by translating the queries to its own oracle (see $SimBRO$ and $SimBRO^{-1}$). Adversary $\mathcal{B}$ simulates $KGen$ oracle similar to the oracle $KGen$ in the $\mathbf{Exp}^4$ with the modification of the $\mathsf{ExecKGen}^1$ function. It uses the $\Pi_Q$ set to know the $Q_1$ value from the re-

ceived commitment $comm_Q$ and then sets $Q_2$ in such a way that the resulting public key is equal to the public key $Q$, provided by its challenger.

Receiving the forgeries from $\mathcal{A}$, the adversary $\mathcal{B}$ constructs the forgery for its own challenger in the same way as defined in case when party $P_2$ is compromised. So, if $\mathcal{A}$ delivers a valid $l+1$ pairs, $\mathcal{B}$ delivers a valid forgery to its own challenger and

$$\Pr\left[\mathbf{Exp}^4(\mathcal{A}) \to 1\right] = \Pr\left[\mathbf{Exp}^{\mathsf{sUF\text{-}KO}}_{\mathsf{GOST}}(\mathcal{B}) \to 1\right].$$

The number of queries made by $\mathcal{B}$ to the $BRO^*$ and $BRO^{*-1}$ oracles is at most $q_{BRO} + 2q_{sign} + 1$ and $q_{BRO^{-1}}$ respectively. The adversary $\mathcal{B}$ needs the same amount of computational resources as $\mathcal{C}$.

Thus, we summarize the obtained bounds in case the party $P_1$ is compromised:

$$\begin{aligned}
\Pr\left[\mathbf{Exp}^2(\mathcal{A}) \to 1\right] &= \left(\Pr\left[\mathbf{Exp}^2(\mathcal{A}) \to 1\right] - \Pr\left[\mathbf{Exp}^3(\mathcal{A}) \to 1\right]\right) + \\
&+ \left(\Pr\left[\mathbf{Exp}^3(\mathcal{A}) \to 1\right] - \Pr\left[\mathbf{Exp}^4(\mathcal{A}) \to 1\right]\right) + \Pr\left[\mathbf{Exp}^4(\mathcal{A}) \to 1\right] \leqslant \\
&\leqslant \frac{q_Q}{2^n} + q_{sign} \cdot \frac{q_R + q_{sign}}{2^n} + q_{sign} \cdot \frac{q_{BRO} + q_{BRO^{-1}} + 2q_{sign}}{q} + \\
&+ \Pr\left[\mathbf{Exp}^{\mathsf{sUF\text{-}KO}}_{\mathsf{GOST}}(\mathcal{B}) \to 1\right] = \frac{q_Q}{2^n} + q_{sign} \cdot \frac{q_R + q_{sign}}{2^n} + \\
&+ q_{sign} \cdot \frac{q_{BRO} + q_{BRO^{-1}} + 2q_{sign}}{q} + \mathsf{Adv}^{\mathsf{sUF\text{-}KO}}_{\mathsf{GOST}}(\mathcal{B}).
\end{aligned}$$

All in all we prove:

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{sOMUF\text{-}PCA}}_{\mathsf{2p\text{-}GOST}}(\mathcal{A}) &= \Pr\left[\mathbf{Exp}^0(\mathcal{A}) \to 1\right] = \\
&= \left(\Pr\left[\mathbf{Exp}^0(\mathcal{A}) \to 1\right] - \Pr\left[\mathbf{Exp}^1(\mathcal{A}) \to 1\right]\right) + \\
&+ \left(\Pr\left[\mathbf{Exp}^1(\mathcal{A}) \to 1\right] - \Pr\left[\mathbf{Exp}^2(\mathcal{A}) \to 1\right]\right) + \Pr\left[\mathbf{Exp}^2(\mathcal{A}) \to 1\right] \leq \\
&\leqslant \frac{(q_{BRO} + q_{BRO^{-1}} + 3q_{sign} + 1)^2}{2^N} + \mathsf{Adv}^{\mathsf{SCR}}_{\mathrm{H}}(\mathcal{C}) + \frac{q_Q}{2^{\min\{\kappa,n\}}} + \\
&+ q_{sign} \cdot \frac{q_R + q_{sign}}{2^{\min\{\kappa,n\}}} + q_{sign} \cdot \frac{q_{BRO} + q_{BRO^{-1}} + 2q_{sign}}{q} + \mathsf{Adv}^{\mathsf{sUF\text{-}KO}}_{\mathsf{GOST}}(\mathcal{B}) \leqslant \\
&\leqslant \mathsf{Adv}^{\mathsf{SCR}}_{\mathrm{H}}(\mathcal{C}) + \mathsf{Adv}^{\mathsf{sUF\text{-}KO}}_{\mathsf{GOST}}(\mathcal{B}) + \frac{q_Q + q_{sign} \cdot (q_R + q_{sign})}{2^{\min\{\kappa,n\}}} + \\
&+ \frac{2(q_{BRO} + q_{BRO^{-1}} + 3q_{sign} + 1)^2}{q}.
\end{aligned}$$

□

# Probing the security landscape for authenticated key establishment protocols

Evgeny Alekseev and Sergey Kyazhin

CryptoPro LLC, Moscow, Russia
{alekseev, kyazhin}@cryptopro.ru

**Abstract**

The paper takes another step towards improving the systematization of the potential adversary capabilities and threats relevant for authenticated key establishment protocols. To do this, we use a new informal concept of the *security landscape* and begin to probe the security landscape for two party authenticated key establishment protocols. The results obtained do not describe all possible threats to authenticated key establishment protocols; on the contrary, they «highlight» some cases and problems that need to be investigated in further work. The results obtained are another attempt to understand the concept of cryptographic system strength.

**Keywords:** authenticated key establishment, security landscape, security model.

*There are a lot of similarities between cryptology and physics.*
*Both use a lot of mathematics, but neither is part of mathematics.*
Phong Q. Nguyen [27]

## 1 Introduction

The task of cryptanalysis is to evaluate the security of cryptosystems. Not so long ago, most cryptosystems were aimed at ensuring only confidentiality. Now, with the significant complication of information systems and the improvement of cryptography, the tasks of cryptography have significantly expanded. In the words of Wenbo Mao, «nowadays, however, cryptography has a modernized role in addition to keeping secrecy of information: ensuring fair play of 'games' by a much enlarged population of 'game players'» [24]. At the same time, the accuracy of the formulation of security properties, which must be evaluated for this cryptosystem, has become critical.

## 1.1 What is a secure cryptosystem?

The statement that some cryptosystem is secure is meaningless if no security model is specified. A classic example of the fact that conclusions about the cryptosystem security depends on the security model is the Diffie-Hellman protocol, secure relative to a passive adversary and not secure relative to an active adversary. The security model consists of the following three components: the adversary model, the threat model, and the adversary resources. The most interesting and difficult to define are the first two components, which are discussed in this paper. At the same time, without determining the quantitative adversary capabilities, that is, its resources (both computational and informational), it is impossible to draw a conclusion about the cryptosystem security. Thus, both for the analysis and for the synthesis of any cryptosystem, it is necessary to define the security model.

## 1.2 How to make a security model for a cryptosystem?

The definition of a security model can be divided into two stages. At the first stage, the existing or proposed conditions for the use of the cryptosystem are analyzed to identify factors important from the point of view of its target properties. This analysis is mostly informal. The second stage consists in the formalization of the identified factors and their mathematical description. The second stage is of particular interest, various approaches to formalization are discussed in a number of papers (see, for example, [3, 5, 8, 9]), but in this paper we focus exclusively on the first stage.

### 1.2.1 Adversary model

The adversary model determines the principal capabilities of the adversary to interact with the analyzed cryptosystem. Note that the quantitative parameters of the adversary are not determined here. The adversary model determines exactly the characteristics whose values will need to be set when determining its resources. For example, if the adversary can passively read messages in the channel, then there is such a parameter as the number of intercepted messages, and if the adversary can control the user acting, then it will be necessary to determine the parameter «number of users whose acting can be controlled».

When listing the adversary capabilities, first of all, the practical aspects of its use should be taken into account. For example, if the cryptosystem is implemented by weakly secured devices, then in some cases it is necessary to

consider the adversary capability to receive part of the intermediate protocol parameters (due to the lack of their secure storage). At the same time, even those capabilities that seem very insignificant should be taken into account: this will avoid situations, an example of which is described in the paper [10]. Note that the justification of the relevance of certain capabilities is not always trivial and is sometimes of particular interest (see, for example, [1, 28]).

However, there is another non-obvious source of the adversary capabilities, which can be called the maximization principle: the adversary's influence on some aspects of the system operation, which in practice he cannot influence. This is usually due to the need to evaluate the cryptosystem security in various scenarios of its use. For example, if the encryption mode requires uniqueness of the initialization value (IV), then there are the following questions: how is this value selected? is the cryptosystem secure for any method of choosing the value? Traditionally, to answer such questions, the adversary is given the capability to set an IV, preserving only its uniqueness [30]. A number of examples of the application of this principle to authenticated key establishment (AKE) protocols will be discussed in this paper.

#### 1.2.2 Threat model

The threat model lists the results of the cryptosystem operation, which lead to damage to the applied information system. At the same time, similarly to the adversary model, a separate value in cryptography is the justification that certain situations can lead to damage (see, for example, [4, 18]).

It should be especially noted that the threat model depends on the adversary model. This is due to the fact that some capabilities allow the adversary to trivially implement some of the threats. For example, having received the long-term key of one of the AKE protocol participants, the adversary can, regardless of the protocol design, impersonate this compromised party. In this situation, it only makes sense to minimize the damage that such a capability will cause. In the considered example, for the compromised party, the threat of impersonation is modified by excluding the trivially implemented part of it (sometimes it is said that «trivial attacks» are excluded), namely, only impersonation «on it» should be as a threat to this party.

### 1.3 How to make sure that all necessary security models are taken into consideration when analyzing the cryptosystem?

It seems impossible to build an absolutely complete security model that takes into account all aspects of the cryptosystem operation (as for any sim-

ulation of real objects). This fact is noted by many, for example, in the work [27]. However, the following two principles can, from our point of view, help to take into account more aspects. The first principle is atomization, i.e. splitting the adversary capability or threat into as small parts as possible.

The second principle is the search for dependencies between the adversary capabilities and threats. Note that there are dependencies between adversary models and threat models. For example, the more capabilities or threats are specified in the model, the more aspects of the cryptosystem operation it allows you to take into account. At the same time, there may be less obvious, but more practical dependencies. An example of such a relationship between the adversary capabilities for AKE protocols is discussed in Section 5. Also, the dependence of the adversary model and the threat model has already been discussed above. However, the main interest from the point of view of cryptosystem security is the dependence between security models, not their components. It is said that one security model is stronger than another if, for an arbitrary cryptosystem (naturally, we are talking about models relevant to the corresponding class of cryptosystems), its security in the first model implies security in the second one. However, such relations can be established exclusively for formally described models (see, for example, [13]). The remarks given in this paper relate to the informal stage and are intended to help better understand the object that will be formalized later.

### 1.4 Related work

This paper is not the first in the field of systematization of the potential adversary capabilities and threats for cryptographic systems. Here are some examples of recent papers containing the results of research in this direction for authenticated key establishment protocols.

In 2022, a report [2] on an attempt to systematize the adversary capabilities for two party AKE protocols was presented at the RusCrypto conference. For each of the capabilities, relevance was justified and examples of application were given.

Also in 2022, a paper [26] containing a list of security properties for cryptographic protocols was published. Among them, according to the authors, the AKE protocols include the following:

– Peer entity authentication (confirmation by one participant of the authenticity of the other, as well as obtaining a guarantee that the participant whose authenticity is confirmed really participates in the protocol);

– Message authentication (confirmation of the authenticity of the message source and the integrity of the transmitted message);

– Replay protection (once correctly accepted by the participant, the message should not be accepted again);

– Key secrecy (during the interaction, the key cannot become known to the adversary, as well as to users for whom this key is not intended);

– Key authentication (the participant receives confirmation that no other participant, except the second one, can know the secret key generated during the protocol execution);

– Key confirmation (the participant receives confirmation that the second participant really knows the secret key);

– Derived key compromise security (compromise of derived keys does not lead to disruption of other security properties both within the current and in other sessions of the protocol);

– Key compromise impersonation resilience (it is impossible for the adversary who has compromised a participant's long-term secret key to impersonate any other participant in front of him);

– Uknown key share resilience (the impossibility of a situation in which participants establish the key, but one of the participants believes that he has established the key with an another participant).

In our opinion, the use of this list in the analysis of protocols is difficult due to, for example, the following circumstances:

– 'Message authentication' and 'Replay protection' properties are not the final security properties of the AKE protocol, since their disruption does not directly lead to damage to the applied information system, i.e. these properties contradict the definition of the threat discussed above;

– it is not clear how one of the participants can get confirmation that no other participant, except the second one, can know the private key, since this confirmation can only be obtained by cryptographic analysis of the protocol; thus, 'Key secrecy' and 'Key authentication' properties appear to be the same;

– the 'Derived key compromise security' property is not atomic, since it means «disruption of other properties».

In addition, the authors included the following properties in the list, but did not refer them to the security properties of AKE protocols:

- Long-term key compromise security (compromise of long-term keys does not lead to disruption of the confidentiality of information transmitted before the key was compromised) — this property, despite the similarity, differs from the 'Derived key compromise security' property, since, in relation to AKE protocols, it means that it is impossible to implement a specific threat of disruption of the established key secrecy if there is a capability of compromising the authentication key, i.e. the so-called perfect forward secrecy property;
- Forcing the established key resilience;
- Peer entity anonymity.

It can also be noted that there are still papers describing new (previously undescribed) threats to AKE protocols. For example, the paper [4] presents a previously undescribed threat of forcing the identical roles.

Thus, the problem of forming a list of security properties, even for two party AKE protocols, can hardly be considered solved.

### 1.5 Our contribution

In this paper, we do not propose new attacks and do not systematize known attacks on AKE protocols. This work is about improving the systematization of the potential adversary capabilities and threats relevant to AKE protocols. In this paper, based on the principles described above, we started probing the security landscape for two party AKE protocols, formed by the following steps.

**Remark 1.** *Note that all the steps are supposed to be done informally. We sometimes use mathematical notation, but we do it not for formalization, but in order to increase the convenience of perception of the results. Mathematical statements should also be interpreted as informal.*

1. Define the interface, i.e. define the set of inputs ($\mathcal{I}$) and outputs ($\mathcal{R}$) of the protocol.
2. Determine a subset of outputs that are considered correct from the point of view of its use in an applied information system ($\mathcal{N} \subseteq \mathcal{R}$).
3. Systematize a subset of outputs that are not considered correct ($\mathcal{S} = \mathcal{R} \setminus \mathcal{N}$).

4. Form and systematize a set of potential adversary capabilities ($\mathcal{A}$).
5. Systematize information about inputs and outputs, the knowledge of which by the adversary may be undesirable from the point of view of using the protocol in the applied information system ($\mathcal{M}$).
6. Form a set of threats ($\mathcal{T} \subseteq \mathcal{S} \times \mathcal{M}$) by analyzing the potential damage to the applied information system that may be caused by abnormal outputs or knowledge of information about outputs.
7. Analyze all pairs of $(\mathsf{a}, \mathsf{t}) \in 2^{\mathcal{A}} \times 2^{\mathcal{T}}$, where $2^M$ is the set of all subsets of the set $M$: if, with the $\mathsf{a}$ adversary capabilities, the $\mathsf{t}$ threat is implemented in a trivial way, then, in order to minimize potential damage, define a modified threat that is not implemented trivially.

## 2 Step 1. Interface of the protocol (sets $\mathcal{I}$, $\mathcal{R}$)

A two party authenticated key establishment (AKE) protocol (between parties $\mathsf{A}$ and $\mathsf{B}$) is a cryptographic protocol for establishing a common secret key for these parties by exchange of messages using unprotected communication channel. Also $\mathsf{A}$ ($\mathsf{B}$) gets confidence that an established common key is unknown for anyone except $\mathsf{B}$ ($\mathsf{A}$).

Consider one of the approaches to defining an AKE protocol. Let $\mathbf{E}$ be the set of identifiers that participants received during registration (together with public keys), let $\mathbf{R}$ be the set of roles of the protocol participants ("Initiator" or "Responder"). The input of the protocol is a pair of the participants identifiers and long-term keys. The output of the protocol is a tuple of values calculated by the parties, which were given as input:

– the output of $\mathsf{A}$: $(S_{\mathsf{A}}, K_{\mathsf{A}}, R_{\mathsf{A}})$, where $S_{\mathsf{A}} = \{\mathsf{A}, P_{\mathsf{A}}\}$ is the set of interacting parties from the $\mathsf{A}$'s point of view ($P_{\mathsf{A}} \in \mathbf{E}$ is a partner of $\mathsf{A}$), $K_{\mathsf{A}}$ is the common key calculated by $\mathsf{A}$, $R_{\mathsf{A}} \in \mathbf{R}$ is the role of $\mathsf{A}$;

– the output of $\mathsf{B}$: $(S_{\mathsf{B}}, K_{\mathsf{B}}, R_{\mathsf{B}})$, where $S_{\mathsf{B}} = \{\mathsf{B}, P_{\mathsf{B}}\}$ is the set of interacting parties from the $\mathsf{B}$'s point of view ($P_{\mathsf{B}} \in \mathbf{E}$ is a partner of $\mathsf{B}$), $K_{\mathsf{B}}$ is the common key calculated by $\mathsf{B}$, $R_{\mathsf{B}} \in \mathbf{R}$ is the role of $\mathsf{B}$.

## 3 Step 2. 'Normal' outputs (a set $\mathcal{N}$)

It is assumed that after successful execution of the protocol, the following statements are fulfill: $S_{\mathsf{A}} = S_{\mathsf{B}}$, $K_{\mathsf{A}} = K_{\mathsf{B}}$, $R_{\mathsf{A}} \neq R_{\mathsf{B}}$. Any output for which at least one equation is not fulfilled is not considered 'normal'.

Recall Remark 1: despite the fact that the description of the input and output of the protocol, as well as these properties look like formal, they should be interpreted as informal.

To more correctly define 'abnormal' outputs, let's assume that each protocol session has some ID that is not part of the protocol. We will assume that the outputs of the participants are associated with this ID. If, according to the participant view, the protocol was not successfully completed (including it was not started), then the output of this participant is an empty string. If both participants have an empty string as an output, then this output is also normal, if only one, then it is not. Then, for example, the situation '$S_{\mathsf{A}} \neq S_{\mathsf{B}}$' includes the situation '$S_{\mathsf{A}}$ is undefined, $S_{\mathsf{B}}$ is defined'.

**Remark 2.** *Note that all the components of the participant's output* $(S_{\mathsf{A}}, K_{\mathsf{A}}, R_{\mathsf{A}})$ *are simultaneously defined or undefined.*

## 4 Step 3. 'Abnormal' outputs (a set $\mathcal{S}$)

Let's systematize abnormal outputs by defining the following properties of outputs for specifying the set $\mathcal{S}$:

S1 $S_{\mathsf{A}} \neq S_{\mathsf{B}}$ (i.e. $P_{\mathsf{A}} \neq \mathsf{B}$ or $P_{\mathsf{B}} \neq \mathsf{A}$);
S2 $K_{\mathsf{A}} \neq K_{\mathsf{B}}$;
S3 $R_{\mathsf{A}} = R_{\mathsf{B}}$.

## 5 Step 4. Adversary capabilities (a set $\mathcal{A}$)

Let's systematize the adversary capabilities for two party AKE protocols. We propose to define the following four classes of the adversary capabilities, determined by the object of his influence: channel (**C**), registration of the adversary (**AR**), registration of users (**UR**), user acting after registration (**UA**).

Registration refers to the inclusion of a user or adversary among the legitimate users of the system. So, we assume that after registration in case of using protocols with authentication based on key pairs, the public key of the registered participant is delivered in a trusted manner to all participants with which it interacts (using a PKI or any other mechanism).

In Tables 1, 2, 3, we have listed the adversary capabilities from the corresponding classes. For some capabilities, we have given examples of papers containing attacks using these capabilities.

| No | Name | Description | Attacks |
|---|---|---|---|
| C1 | Passive network adversary | to receive information transmitted through the channel and determine the place of the transmitted message in the protocol | [7, 14] |
| C2 | Interaction with users | to send messages to users | [7, 33] |
| C3 | Active network adversary | to delay, modify, replace, delete and generate messages transmitted through the channel | [18, 25] |

Table 1: Adversary capabilities from the class C

| No | Name | Description | Attacks |
|---|---|---|---|
| AR0 | Adversary registration | to register the adversary | [7, 25] |
| AR1/ UR1 | Registration during operation (for adversary/user) | to register the adversary/user after the start of at least one protocol session | [6, 25] |
| AR2/ UR2 | One key for initiator and responder (for adversary/user) | to use the same authentication information (key pair) for the adversary/user to participate in the protocol as an initiator and as a responder | [18] |
| AR3/ UR3 | Choosing/forcing a key | to select a key pair when registering the adversary/user | [6, 25] |
| AR4/ UR4 | Choosing/forcing an ID | to select an ID when registering the adversary/user | [25] |
| AR5 | No verification of sk knowledge | to register the adversary without verification of knowledge of the private key corresponding to the public key | [25] |
| AR6 | No verification of pk uniqueness | to register the adversary without verification of key pair uniqueness | [25] |
| UR5 | Forcing an inconsistent key | to register the user with a key pair where the private key does not match the public key | |
| UR6 | Repeating a key | to register the user with a key pair equal to the key pair of an already registered user (without knowing the private key) | |
| UR7 | Repeating a key with modification (related key) | to register the user with a key pair that differs from the key pair of an already registered user in a known way (to use 'related keys' without knowing the private keys) | |

Table 2: Adversary capabilities from the classes AR, UR

| No | Name | Description | Attacks |
|---|---|---|---|
| UA1 | Parallel sessions | to establish multiple protocol sessions simultaneously for the user | [18] |
| UA2 | Forcing user interaction | to force the interaction of two users | [29, 33] |
| UA3 | Session key compromise | to compromise the established key | [7] |
| UA4/ UA5 | Long-term key compromise (before/after the session) | to compromise the private key of the user before/after the session | [14, 33] |
| UA6/ UA7 | Intermediate private value compromise (before/after the session) | to compromise intermediate secret values generated or calculated by a participant during protocol execution, before/after their use | [21, 33] |
| UA8/ UA9 | Intermediate public value compromise (before/after the session) | to compromise intermediate public values generated or calculated by a participant during protocol execution, before/after using these values or corresponding secret values | [32] |
| UA10/ UA11 | Forcing an intermediate private/public value | to choose intermediate private/public values generated or calculated by the participant during protocol execution | |
| UA12/ UA13 | Repeating an intermediate private/public value | to use for the user intermediate private/public values generated or calculated during protocol execution, which are equal to previously used (without knowing these values) | |
| UA14/ UA15 | Repeating an intermediate secret/public value with modification | to use for the user intermediate private/public values generated or calculated during protocol execution, which differ from previously used in a known way (without knowing these values) | |

Table 3: Adversary capabilities from the class UA

The tables 4, 5 show the results of additional systematization of the adversary capabilities for compromising, forcing and repeating keys and intermediate values. For some cells, the adversary capabilities are not defined for the appropriate reasons:

$\star$ there is no session key before the session;

$\star\star$ the session key has no «public part»;

$\star\star\star$ knowledge of the public long-term key is included in the minimum adversary capabilities.

| | Session key | | Long-term key | | Intermediate value | |
|---|---|---|---|---|---|---|
| | before session | after session | before session | after session | before session | after session |
| Secret | $\star$ | UA3 | UA4 | UA5 | UA6 | UA7 |
| Public | $\star\star$ | | $\star\star\star$ | | UA8 | UA9 |

Table 4: Systematized adversary capabilities of key compromise

| | Long-term key | | Intermediate value | |
|---|---|---|---|---|
| | Forcing | Repeating | Forcing | Repeating |
| Secret | UR3 | UR6 | UA10 | UA12 |
| Public | $\star\star\star$ | | UA11 | UA13 |

Table 5: Systematized adversary capabilities of forcing and repeating the key

The figure 1 shows the result of additional systematization of capabilities from the classes C, AR, UR using a directed graph. The black dot is the so-called minimum adversary capabilities, including knowledge of user IDs and public keys, as well as a full description of the protocol. The dots of green (red, yellow) are the adversary capabilities from class C (AR, UR). The presence of the $(x, y)$ edge means that if the adversary has the ability $x$, then he also has the ability $y$. The presence of a set of dashed edges $\{(x, z), (y, z)\}$ means that if the intruder has both the $x$ possibility and the $y$ possibility, then he also has the $z$ possibility.

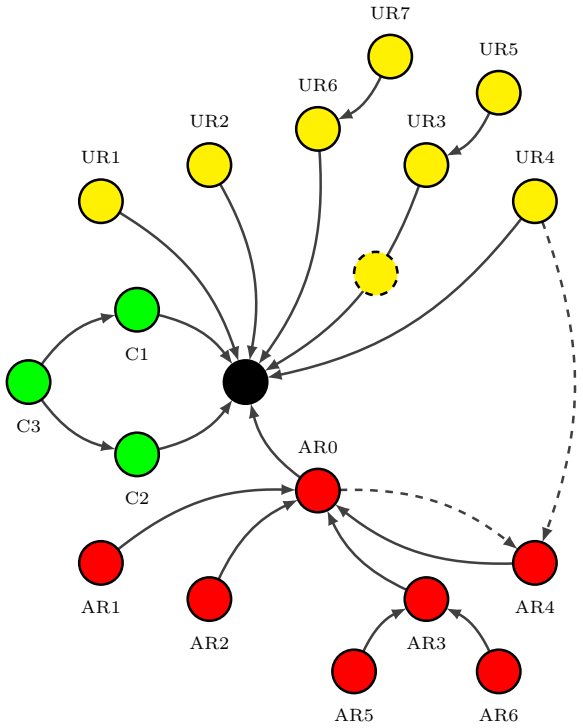



Figure 1: Adversary capabilities subgraph (for the classes C, AR, UR)

The list of the adversary capabilities is informal and is the result of finding a tradeoff between practice and usability. Therefore, this list may contain inaccuracies or seem insufficiently «atomic». For example, the list contains the 'Forcing a key' capability, implying the forcing any key pair chosen by the adversary. However, sometimes in practice, the adversary can only force a key pair with some property. In this case, for completeness, it is necessary to consider another capability (in the Figure 1 it is depicted as a yellow dot with a dotted contour) between the 'Forcing a key' capability and minimal capabilities.

## 6 Step 5. Information that cannot be known (a set $\mathcal{M}$)

It is undesirable for the adversay to know non-trivial information about any of the elements of the protocol output. In order to systematize this information, let's define the appropriate properties M1 (M2, M3) for specifying the set $\mathcal{M}$: $S_{\mathsf{A}}$ or $S_{\mathsf{B}}$ ($K_{\mathsf{A}}$ or $K_{\mathsf{B}}$, $R_{\mathsf{A}}$ or $R_{\mathsf{B}}$) is distinguishable from random for anyone other than $\mathsf{A}$, $\mathsf{B}$.

## 7 Step 6. Threats (a set $\mathcal{T}$)

First of all, it should be noted that there are security properties (threats) that cannot be described using the properties listed in step 3. An example of such a security property is deniability [15]. The threat corresponding to this property is that the output of the protocol (together with its transcript) allows the user to prove participation in the protocol session. The existence of such a property once again underlines the complexity of taking into account all threats when analyzing the strength of a cryptographic protocol.

However, most of the threats to two party AKE protocols can be described using the properties defined above. Some of the properties (for example, S3 or M2) directly describe subsets of outputs that are threats, that is, they can directly lead to damage to the applied information system. However, more often threats are described by special cases of these properties or their combinations. Here are examples of possible special cases of these properties:

| | |
|---|---|
| S1(a) | $P_{\mathsf{A}} \neq \mathsf{B}$ and $P_{\mathsf{B}} \neq \mathsf{A}$; |
| S1(b) | $P_{\mathsf{A}} = \mathsf{B}$, but $P_{\mathsf{B}} \neq \mathsf{A}$; |
| S2(a) | both keys are calculated, but not equal; |
| S2(b) | one of the keys is not calculated; |
| M2(a) | $K_{\mathsf{A}}$ and $K_{\mathsf{B}}$ are distinguishable from random; |
| M2(b) | $K_{\mathsf{A}}$ and $K_{\mathsf{B}}$ are known. |

The table 6 provides examples of known authentication disruption threats and their corresponding protocol output properties.

| **Threat name** | **Threat description** | **Output properties** |
|---|---|---|
| Impersonation | $\mathsf{A}$ believes that the key with $\mathsf{B}$ is established, but $\mathsf{B}$ believes that the key with $\mathsf{A}$ is not established | S1(b) |
| Unknown key share (UKS), version 1 [12] | $\mathsf{A}$ and $\mathsf{B}$ have established the key, but $\mathsf{A}$ or $\mathsf{B}$ believes that the key is established with another participant | S1 and not S2 |
| UKS, version 2 [6] | $\mathsf{A}$ and $\mathsf{B}$ have established the key, $\mathsf{A}$ believes that the key with $\mathsf{B}$ is established, but $\mathsf{B}$ believes that the key is established with another participant | S1(b) and not S2 |
| Bilateral unknown key share (BUKS) [11] | $\mathsf{A}$ and $\mathsf{B}$ have established the key, but $\mathsf{A}$ and $\mathsf{B}$ believe that the key is established with another participant (not $\mathsf{A}$, $\mathsf{B}$) | S1(a) and not S2 |

Table 6: Examples of authentication disruption threats

We can also consider a special case of the property S1(a): $P_{\mathsf{A}} = P_{\mathsf{B}} = \mathsf{C} \notin \{\mathsf{A}, \mathsf{B}\}$. This property may be a separate threat. For example, there is an attack [18] that implements this threat for $\mathsf{A} = \mathsf{B}$.

## 8 Step 7. Pairs $(\mathsf{a}, \mathsf{t}) \in 2^{\mathcal{A}} \times 2^{\mathcal{T}}$

If the adversary has a certain set of capabilities, the implementation of the threat may become trivial. For example, if the adversary has the C1 capability, then the threat of roles anonymity disruption, described by the M3 property, is realized. In general, quoting the title of the paper [23], «defining trivial attacks for security protocols is not trivial».

The table 7 provides examples of known security properties corresponding to threats modified due to trivial attacks:

– if the adversary has the UA3 capability, then the key secrecy disruption threat, described by the M2 property, is implemented in a trivial way (modified threat: the key $K_{\mathsf{A}}$ or $K_{\mathsf{B}}$ established in a session in which the session key was not compromised is distinguishable from random);

– if the adversary has the UA5 capability and some others capabilities (for example, C3), then the key secrecy disruption threat is implemented in a trivial way for most existing two party AKE protocols (modified threat:

the key $K_{\mathsf{A}}$ established before compromising the long-term key of the party $\mathsf{A}$ is distinguishable from random);

– if the adversary has the UA4 capability, then in case of compromise of the key of the participant $\mathsf{A}$, the threat of impersonation of the participant $\mathsf{A}$, described by the S1(b) property, is implemented in a trivial way; however, the threat of impersonation of another participant for the participant $\mathsf{A}$ remains relevant.

| **Security property name** | **Base threat (property of output)** | **Critical adversary capability** |
|---|---|---|
| Key Freshness [31] | M2 | UA3 |
| Perfect Forward Secrecy (PFS) [16, 19] | M2 | UA5 |
| Key Compromise Impersonation (KCI) Resilience [20] | S1(b) | UA4 |

Table 7: Examples of known security properties corresponding to threats modified due to trivial attacks

## 9 Additional remark on interdependence of threats

Consider the M2 property. It assumes that at least one of the keys (for example, $K_{\mathsf{A}}$) is distinguishable from a random one. Recall that a necessary condition for assigning the subset to 'abnormal' is that at least one of the parties believes that the protocol has been completed successfully. Therefore, the M2 property also makes sense in the case when the key $K_{\mathsf{A}}$ is defined, and the key $K_{\mathsf{B}}$ is not defined (i.e. the S2(b) property is fulfilled). According to the remark 2, this means that $S_{\mathsf{A}}$ is defined, and $S_{\mathsf{B}}$ is not defined. Therefore, the S1(b) property is fulfilled.

Thus, the implementation of the authentication disruption threat may be a necessary condition for the implementation of the key secrecy disruption threat. As an example demonstrating such a dependency, we can give an example of an attack from the paper [17], which leads to the PFS property disruption using the implementation of the KCI threat. Note that this attack uses the same special case of a threat to disrupt the PFS property: as a result of an attack implementing the KCI threat, one of the keys is not defined.

## 10 Conclusion

In this paper, we started probing the security landscape for two party AKE protocols in order to improve the systematization of the potential adver-

sary capabilities and threats relevant to AKE protocols. The results obtained do not describe all possible threats to AKE protocols; on the contrary, they «highlight» some cases and problems that need to be investigated in further work. Among them:

– the usual systematization of threats described in this paper is not complete (there are such security properties as deniability);

– there are subsets of AKE protocol outputs that can be considered as separate threats, but they were not previously classified as such.

– seemingly independent threats can be interdependent (for example, a KCI threat may follow from a PFS disruption).

# References

[1] Akhmetzyanova L., Alekseev E., Smyshlyaeva E., Sokolov A., "On post-handshake authentication and external PSKs in TLS 1.3", *Journal of Computer Virology and Hacking Techniques*, **16**:4 (2020), 269–274.

[2] Alekseev E.K., Akhmetzyanova L.R., Bozhko A.A., Kutsenok K.O., Kyazhin S.N., *On adversary capabilities for attacks on a certain class of authenticated key establishment protocols*, RusCrypto 2022, 2022, In Russian.

[3] Alekseev E.K., Akhmetzyanova L.R., Zubkov A.M., Karpunin G.A., Smyshlyaev S.V., "On one approach to formalizing cryptographic analysis tasks", *Matematicheskie Voprosy Kriptografii*, To be published, In Russian.

[4] Alekseev E., Kyazhin S., Smyshlyaev S., "The Threat of Forcing the Identical Roles for Authenticated Key Establishment Protocols", *Journal of Computer Virology and Hacking Techniques*, 2023.

[5] Bellare M., Rogaway P., "Code-Based Game-Playing Proofs and the Security of Triple Encryption", Cryptology ePrint Archive, Paper 2004/331, 2004.

[6] Blake-Wilson S., Menezes A., "Unknown Key-Share Attacks on the Station-to-Station (STS) Protocol", *Public Key Cryptography*, Springer, Berlin, Heidelberg, 1999, 154–170.

[7] Burmester M., "On the Risk of Opening Distributed Keys", *Advances in Cryptology — CRYPTO'94*, Springer, Berlin, Heidelberg, 1994, 308–317.

[8] Burrows M., Abad M., Needham R.M., "A Logic of Authentication", *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, **426**:1871 (1989), 233–271.

[9] Canetti R., "Universally composable security: a new paradigm for cryptographic protocols", *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, 2001, 136–145.

[10] Canvel B., Hiltgen A., Vaudenay S., Vuagnoux M., "Password Interception in a SSL/TLS Channel", *Advances in Cryptology — CRYPTO 2003*, Springer, Berlin, Heidelberg, 2003, 583–599.

[11] Chen L., Tang Q., "Bilateral Unknown Key-Share Attacks in Key Agreement Protocols", *Journal of Universal Computer Science*, **14**:3 (2008), 416–440.

[12] Cheng Z., Chen L., Comley R., Tang Q., "Identity-Based Key Agreement with Unilateral Identity Privacy Using Pairings", *Information Security Practice and Experience*, Springer, Berlin, Heidelberg, 2006, 202–213.

[13] Cremers C.J.F., "Session-state Reveal Is Stronger Than Ephemeral Key Reveal: Attacking the NAXOS Authenticated Key Exchange Protocol", *Applied Cryptography and Network Security*, Springer, Berlin, Heidelberg, 2009, 20–33.

[14] Cremers C., Horvat M., "Improving the ISO/IEC 11770 standard for key management techniques", *International Journal of Information Security*, **15**:6 (2016), 659–673.

[15] Di Raimondo M., Gennaro R., Krawczyk H., "Deniable Authentication and Key Exchange", *Proceedings of the 13th ACM Conference on Computer and Communications Security*, Association for Computing Machinery, New York, NY, USA, 2006, 400–409.

[16] Diffie W., van Oorschot P.C., Wiener M.J., "Authentication and authenticated key exchanges", *Designs, Codes and Cryptography*, **2** (1992), 107–125.

[17] Dowling B., Paterson, K.G., "A Cryptographic Analysis of the WireGuard Protocol", *Applied Cryptography and Network Security*, Springer International Publishing, Cham, 2018, 3–21.

[18] Drucker N., Gueron S., "Selfie: Reflections on TLS 1.3 with PSK", *Journal of Cryptology*, **34**:3 (2021).

[19] Günther C.G., "An Identity-Based Key-Exchange Protocol", *Advances in Cryptology — EUROCRYPT '89*, Springer, Berlin, Heidelberg, 1990, 29–37.

[20] Just M., Vaudenay S., "Authenticated multi-party key agreement", *Advances in Cryptology — ASIACRYPT '96*, Springer, Berlin, Heidelberg, 1996, 36–49.

[21] Krawczyk H., "HMQV: A High-Performance Secure Diffie-Hellman Protocol", *Advances in Cryptology – CRYPTO 2005*, Springer, Berlin, Heidelberg, 2005, 546–566.

[22] Krawczyk H., "SIGMA: The 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman and Its Use in the IKE Protocols", *Advances in Cryptology – CRYPTO 2003*, Springer, Berlin, Heidelberg, 2003, 400–425.

[23] Li Y., Schäge S., "No-Match Attacks and Robust Partnering Definitions: Defining Trivial Attacks for Security Protocols is Not Trivial", *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, Association for Computing Machinery, New York, NY, USA, 2017, 1343–1360.

[24] Mao W., *Modern Cryptography: Theory and Practice*, Pearson Education, 2003, ISBN: 9780134171845.

[25] Menezes A., Ustaoglu B., "Comparing the Pre- and Post-specified Peer Models for Key Agreement", *Information Security and Privacy*, Springer, Berlin, Heidelberg, 2008, 53–68.

[26] Nesterenko A.Yu., Semenov A.M., "Methodology for assessing the security of cryptographic protocols", *Prikladnaya Diskretnaya Matematika*, **56** (2022), 33–82, In Russian.

[27] Nguyen P.Q., "Cryptanalysis vs. Provable Security", *Information Security and Cryptology*, Springer, Berlin, Heidelberg, 2012, 22–23.

[28] Razali E., Phan R.C.-W., "On the Existence of Related-Key Oracles in Cryptosystems Based on Block Ciphers", *On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops*, Springer, Berlin, Heidelberg, 2006, 425–438.

[29] Ristenpart T., Yilek S., *When Good Randomness Goes Bad: Virtual Machine Reset Vulnerabilities and Hedging Deployed Cryptography*, Network and Distributed System Security Symposium, 2010.

[30] Rogaway P., "Nonce-Based Symmetric Encryption", *Fast Software Encryption*, Springer, Berlin, Heidelberg, 2004, 348–358.

[31] Rueppel R.A., van Oorschot P.C., "Modern key agreement techniques", *Computer Communications*, **17**:7 (1994), 458–465.

[32] Seye P.B., Sarr A.P., "Enhanced Modelling of Authenticated Key Exchange Security", *Security and Trust Management*, Springer International Publishing, Cham, 2017, 36–52.

[33] Tang Q., Chen L., "Extended KCI attack against two-party key establishment protocols", *Information Processing Letters*, **111**:15 (2011), 744–747.

# On the confidentiality and integrity of ECIES scheme

Kirill Tsaregorodtsev

JSC "NPK Kryptonite", Moscow, Russia
k.tsaregorodtsev@kryptonite.ru

**Abstract**

In the paper we analyze ECIES scheme in the provable security framework. We show that if ECIES scheme is instantiated with the key exchange scheme $\mathcal{KE}$ and authenticated encryption scheme $\mathcal{AE}$, then the insecurity of ECIES scheme in the (standard) LOR-CCA and INT-CTXT models can be upper bounded by the insecurities of $\mathcal{KE}$ (in the MODH model, Oracle Diffie-Hellman Model with multiple queries) and $\mathcal{AE}$ (in the LOR-CCA and INT-CTXT models respectively).

**Keywords:** ECIES, provable security

## Introduction

ECIES is widely standardized ([1, 2, 3]) hybrid encryption scheme that provides confidentiality and integrity of messages. The security of ECIES was analyzed in the "provable security" framework in [4, 5]. However, the treatment in [3, 4, 5] has some drawbacks:

– only the confidentiality of the ECIES scheme is analyzed; integrity of the scheme (either in the INT-CTXT model, see Section 2.2, or in the INT-PTXT model, see [6, Section 2]) is not reviewed; in general, it is known that LOR-CCA security does not imply integrity (see, e.g.,[6, Section 3], [7, Section 5.2.2]);

– the confidentiality model in these articles (LOR-CCA-fg/IND-CCA2) allows only one encryption challenge query to the $\mathcal{O}^b_{\text{enc}}$ oracle (see Section 2.1 for more details); generalization to the case of $q_e$ queries to the encryption oracle seems not to be the immediate consequence (cf. [7, Theorem 12.6], where each pair of messages is processed under the same private/public key pair); however, the possibility to ask a number of queries instead of a single one can make a difference in practice (see [8]);

– the analysis could be slightly more general: it allows any AE(AD)-scheme (see Definition 1) to be used instead of concrete Encrypt-then-MAC approach (for instance, MGM mode [9, 10] can be used).

In this work we give a formal analysis of ECIES in the "usual" LOR-CCA and INT-CTXT models (with multiple queries). The security in these models implies the following informal properties:

– the adversary is unable to extract any useful information about plaintext from the given ciphertext (except for its length);

– if the adversary is given some ephemeral public key (chosen by the honest party), it is unable to form the ciphertext that correctly decrypts under this key (for instance, it cannot modify messages formed by honest senders).

We stress out that these properties does not imply the authenticity of the sender (i.e., the receiver cannot securely identify the origin of the received message, only the message content), which means that additional mechanisms are needed to ensure it. However, the scheme can be used as a building block of more involved protocols (e.g., as a part of user anonymous authentication in 5G-AKA protocol [11]).

The paper is structured as follows. In Section 1 we introduce necessary definitions and notation to be used in the paper; Section 2 is devoted to the formal definitions of security models for the algorihms of interest. In Section 3 we show how to reduce multi-user models to the standard single-user case. Section 4 deals with the security reduction of ECIES scheme. Finally, Section 5 shortly lists the results of the paper.

# 1 Preliminaries

## 1.1 Notations

The length of the message $m$ (in blocks) is denoted as $|m|$. By $x \leftarrow y$ we denote assigning value $y$ to a variable $x$; $x \xleftarrow{\$} \mathcal{O}$ denotes the process of running the probabilistic algorithm $\mathcal{O}$ and assigning the resulting value to $x$. Uninitialized associative array (dictionary) is denoted as $[\,]$. Symbol $\perp$ denotes either an error message (e.g. decryption failure), or an empty (unitialized) element of associative array.

### 1.2 Authenticated encryption scheme

**Definition 1.** *Authenticated encryption (AE) scheme is the following triplet* $\mathcal{AE} = (\texttt{KeyGen}, \texttt{Enc}, \texttt{Dec})$ *of (probabilistic) algorithms:*

- *key generation algorithm* `KeyGen`*; it takes no input and returns a randomly chosen key* $k$ *from the key space (from now on we will assume that the key is uniformly random over the set* $\{0,1\}^{klen}$*);*
- *encryption algorithm* `Enc`*; it takes the key* $k$ *and the message* $m$ *to be encrypted and returns a ciphertext* $ct \xleftarrow{\$} \mathcal{AE}.\texttt{Enc}(k, m)$*;*
- *decryption algorithm* `Dec`*; it takes the key* $k$ *and the ciphertext* $ct$ *and returns* $m \leftarrow \mathcal{AE}.\texttt{Dec}(k, ct)$*, which is either some message, or the special decryption error symbol* $\perp$*.*

*The standard requirement of correct decryption must hold: for any* $m$ *and any* $k \xleftarrow{\$} \mathcal{AE}.\texttt{KeyGen}$ *it is true that* $\mathcal{AE}.\texttt{Dec}(k, \mathcal{AE}.\texttt{Enc}(k, m)) = m$*.*

### 1.3 Key exchange scheme

In this paper by "key exchange scheme" (see also [12]) we mean the cryptographic mechanism that allows two parties to derive secret key based on two key pairs (ephemeral $(esk, epk)$ and long-term $(sk, pk)$) in two steps:

- party A generates ephemeral key pair $(esk, epk)$ and sends $epk$ to the party B which has a long-term key pair $(sk, pk)$;
- party A and party B are now able to generate shared secret value $\texttt{Combine}(esk, pk) = \texttt{Combine}(sk, epk) = k$.

An example of key exchange scheme (denoted as "key exchange algorithm" VKO) is given in [12, Section 3.7].

**Definition 2.** *Key exchange scheme is the pair of algorithms* $\mathcal{KE} = (\texttt{KeyPairGen}, \texttt{Combine})$*:*

- `KeyPairGen` *is a private-public key pair generation algorithm; it takes no input and returns a randomly chosen key pair* $(sk, pk)$*;*
- `Combine` *is a shared secret value generation algorihtm; it takes some private key* $sk$ *and public key* $pk$ *and generates shared secret* $k$*.*

*The standard requirement of correct shared secret generation must hold: for any two key pairs* $(sk, pk) \xleftarrow{\$} \mathcal{KE}.\texttt{KeyPairGen}$ *and* $(esk, epk) \xleftarrow{\$} \mathcal{KE}.\texttt{KeyPairGen}$ *it holds that* $\mathcal{KE}.\texttt{Combine}(sk, epk) = \mathcal{KE}.\texttt{Combine}(esk, pk)$*.*

### 1.4 ECIES scheme

The object of study (ECIES) is an asymmetric (hybrid) authenticated encryption scheme based on the key exchange scheme $\mathcal{KE}$ and AE(AD)-scheme $\mathcal{AE}$. The encryption process consists of two steps:

- generating ephemeral pair and session secret key $k$ using $\mathcal{KE}$;
- encrypting the message $m$ under the key $k$ using $\mathcal{AE}$ and sending $(epk, ct)$ to the recipient.

The pseudocode description of the scheme is given in Fig. 1.

| ECIES.$\texttt{Enc}(pk, m)$ | ECIES.$\texttt{Dec}(epk, sk, ct)$ |
|---|---|
| $(esk, epk) \xleftarrow{\$} \mathcal{KE}.\texttt{KeyPairGen}()$ | $k \leftarrow \mathcal{KE}.\texttt{Combine}(sk, epk)$ |
| $k \leftarrow \mathcal{KE}.\texttt{Combine}(esk, pk)$ | **return** $\mathcal{AE}.\texttt{Dec}(k, ct)$ |
| $ct \leftarrow \mathcal{AE}.\texttt{Enc}(k, m)$ | |
| **return** $(epk, ct)$ | |

Figure 1: Pseudocode of ECIES scheme

**Remark 1.** *Note that a fresh pair of ephemeral keys* $(esk, epk)$ *is generated during each invocation of* ECIES.`Enc` *algorithm.*

**Remark 2.** *It is possible to use AEAD-schemes (see, e.g., [7, Section 5.2.1]) instead of AE-schemes (i.e. to have some additional data to be attached to the ciphertext).*

## 2 Security models

In this section we describe security models by means of Experiments, each of which formalizes some intuitive notion of security (for AE-scheme, key exchange scheme, ECIES scheme) in the "provable security" framework [7, 13]. We specify concrete oracles (interfaces) and determine success measure (advantage) of the adversary (some probabilistic algorithm) in each of the Experiments. Later we obtain explicit estimates of advantages in terms of adversarial time complexity and query complexity (e.g., number of oreacle queries, total (maximal) length of the queries).

**Remark 3.** *By time complexity of the adversary we mean the sum of the number of steps in some fixed model of computations and the code size of the program description of the adversary (in order to avoid trivial attacks based on lookup tables).*

## 2.1 Confidentiality model

Let us introduce the LOR-CCA model (Left-or-Right, Chosen Ciphertext Attack) for the AE-scheme $\mathcal{AE}$ in the multi-user setting. Let $D \in \mathbb{N}$ be the number of users in the model. The adversary $\mathcal{A}$ interacts with two oracles $\mathcal{O}^b_{\text{enc}}$ and $\mathcal{O}_{\text{dec}}$:

- $\mathcal{O}^b_{\text{enc}}$ takes as input a triple $(i, m_0, m_1)$ consisting of a message pair $(m_0, m_1)$ of equal length ($|m_0| = |m_1|$) and a number $i \in \{1, \ldots, D\}$; it encrypts the message $m_b$ using $\mathcal{AE}.\texttt{Enc}$ under the key $k_i$ and returns the resulting ciphertext $ct$ to $\mathcal{A}$;
- $\mathcal{O}_{\text{dec}}$ takes as input a message (ciphertext) $ct$ and a number $i \in \{1, \ldots, D\}$; if $ct$ was not returned as an answer to the $\mathcal{O}_{\text{enc}}$ query of the type $(i, \cdot, \cdot)$ before, then $\mathcal{O}_{\text{dec}}$ decrypts $ct$ using $\mathcal{AE}.\texttt{Dec}$ under the key $k_i$ and returns the result to $\mathcal{A}$.

The adversary's goal is to predict the bit $b$ fixed in the $\mathcal{O}^b_{\text{enc}}$-oracle using answers to its queries. If $\mathcal{A}$ guesses $b$ correctly with high probability, then it means that $\mathcal{A}$ is able to retrieve some useful information from adaptive plaintext/ciphertext queries.

**Definition 3.** *The advantage of the adversary $\mathcal{A}$ in the* LOR-CCA *model with $D$ parties (users) is the following quantity:*

$$\mathbf{Adv}^{\text{LOR-CCA}}_{\mathcal{AE}}(\mathcal{A}) = \mathbb{P}\big[\mathbf{Exp}^{\text{LOR-CCA-1}}_{\mathcal{AE}}(\mathcal{A}) \to 1\big] - \mathbb{P}\big[\mathbf{Exp}^{\text{LOR-CCA-0}}_{\mathcal{AE}}(\mathcal{A}) \to 1\big],$$

*the pseudocode of $\mathbf{Exp}^{\text{LOR-CCA-}b}_{\mathcal{AE}}$, $b \in \{0, 1\}$, is given in Fig. 2.*

| $\mathbf{Exp}^{\text{LOR-CCA-}b}_{\mathcal{AE}}(\mathcal{A})$ | $\mathcal{O}^b_{\text{enc}}(i, m_0, m_1)$ |
|---|---|
| **for** $1 \le i \le D$ **do** | $ct \xleftarrow{\$} \mathcal{AE}.\texttt{Enc}(k_i, m_b)$ |
| $\quad k_i \xleftarrow{\$} \mathcal{AE}.\texttt{KeyGen}()$ | $sent[i] \leftarrow sent[i] \cup \{ct\}$ |
| **endfor** | **return** $ct$ |
| $sent \leftarrow [\,]$ | $\mathcal{O}_{\text{dec}}(i, ct)$ |
| $b' \xleftarrow{\$} \mathcal{A}^{\mathcal{O}^b_{\text{enc}}, \mathcal{O}_{\text{dec}}}$ | **if** $(ct \in sent[i])$ |
| **return** $b'$ | $\quad$ **return** $\perp$ |
| | **fi** |
| | **return** $\mathcal{AE}.\texttt{Dec}(k_i, ct)$ |

Figure 2: Pseudocode of the LOR-CCA Experiment

**Definition 4.** *Let $\mathbf{InSec}^{\text{LOR-CCA}}_{\mathcal{AE}}(t, Q_e, Q_d, L_e, L_d, M_e, M_d; D)$ be the maximal advantage $\mathbf{Adv}^{\text{LOR-CCA}}_{\mathcal{AE}}(\mathcal{A})$, where the maximum is taken over the adversaries $\mathcal{A}$ whose time complexity is at most $t$ and with the following restrictions on oracle queries ($1 \le i \le D$):*

– *the number of queries of the type* $(i, \cdot, \cdot)$ *to the* $\mathcal{O}^b_{\text{enc}}$ *oracle (*$(i, \cdot)$ *to the* $\mathcal{O}_{\text{dec}}$ *oracle) does not exceed* $Q_e[i]$ *(*$Q_d[i]$ *resp.);*

– *the total length of the queries* $\sum |m_0| = \sum |m_1|$ *among queries of the type* $(i, m_0, m_1)$ *to the* $\mathcal{O}^b_{\text{enc}}$ *oracle (*$\sum |ct|$ *among queries of the type* $(i, ct)$ *to the* $\mathcal{O}_{\text{dec}}$ *oracle) does not exceed* $L_e[i]$ *(*$L_d[i]$ *resp.);*

– *the maximal length of the query* $\max |m_0| = \max |m_1|$ *among queries of the type* $(i, m_0, m_1)$ *to the* $\mathcal{O}^b_{\text{enc}}$ *oracle (*$\max |ct|$ *among queries of the type* $(i, ct)$ *to the* $\mathcal{O}_{\text{dec}}$ *oracle) does not exceed* $M_e[i]$ *(*$M_d[i]$ *resp.).*

**Remark 4.** *For the case* $D = 1$ *we obtain the usual* LOR-CCA *model for the confidentiality of authenticated encryption [6]. If additionally the adversary does not have oracle access to the* $\mathcal{O}_{\text{dec}}$*-oracle, we obtain the* LOR-CPA*-model for the encryption scheme.*

## 2.2 Integrity model

Let us introduce the INT-CTXT model (Integrity of Ciphertexts) for the AE-scheme $\mathcal{AE}$ in the multi-user setting. Let $D \in \mathbb{N}$ be the number of users in the model. The adversary $\mathcal{A}$ interacts with two oracles $\mathcal{O}_{\text{enc}}$ and $\mathcal{O}_{\text{verify}}$:

– $\mathcal{O}_{\text{enc}}$ takes as input a message $m$ and a number $i \in \{1, \ldots, D\}$; it encrypts the message $m$ using $\mathcal{AE}.\texttt{Enc}$ under the key $k_i$ and returns the resulting ciphertext $ct$ to $\mathcal{A}$;

– $\mathcal{O}_{\text{verify}}$ takes as input a message (ciphertext) $ct$ and a number $i \in \{1, \ldots, D\}$; it decrypts $m \leftarrow \mathcal{AE}.\texttt{Dec}(k_i, ct)$ and returns $m$ to $\mathcal{A}$; if $ct$ was not returned as an answer to the $\mathcal{O}_{\text{enc}}$ query of the type $(i, \cdot)$ before and $m \neq \perp$ (correct decryption), then $\mathcal{O}_{\text{verify}}$ sets flag $win \leftarrow \texttt{true}$.

The adversary's goal is to forge fresh ciphertext $ct$ that is decrypted to the correct plaintext (i.e. to set the flag $win \leftarrow \texttt{true}$).

**Definition 5.** *The advantage of the adversary* $\mathcal{A}$ *in the* INT-CTXT *model with* $D$ *parties (users) is the following quantity:*

$$\mathbf{Adv}^{\text{INT-CTXT}}_{\mathcal{AE}}(\mathcal{A}) = \mathbb{P}\big[\mathbf{Exp}^{\text{INT-CTXT}}_{\mathcal{AE}}(\mathcal{A}) \to 1\big],$$

*the pseudocode of* $\mathbf{Exp}^{\text{INT-CTXT}}_{\mathcal{AE}}$*,* $b \in \{0, 1\}$*, is given in Fig. 3.*

$\mathbf{Exp}^{\text{INT-CTXT}}_{\mathcal{AE}}(\mathcal{A})$

**for** $1 \le i \le D$ **do**
  $k_i \xleftarrow{\$} \mathcal{AE}.\texttt{KeyGen}$
**endfor**
$sent \leftarrow [\,]$
$win \leftarrow \texttt{false}$
$\mathcal{A}^{\mathcal{O}_{\text{enc}},\mathcal{O}_{\text{verify}}}$
**return** $win$

$\mathcal{O}_{\text{enc}}(i, m)$

$ct \xleftarrow{\$} \mathcal{AE}.\texttt{Enc}(k_i, m)$
$sent[i] \leftarrow sent[i] \cup \{ct\}$
**return** $ct$

$\mathcal{O}_{\text{verify}}(i, ct)$

$m \leftarrow \mathcal{AE}.\texttt{Dec}(k_i, ct)$
**if** $(ct \notin sent[i])\,\&\,(m \neq \perp)$
  $win \leftarrow 1$
**fi**
**return** $m$

Figure 3: Pseudocode of the INT-CTXT Experiment

**Definition 6.** *Let* $\mathbf{InSec}^{\text{INT-CTXT}}_{\mathcal{AE}}(t, Q_e, Q_v, L_e, L_v, M_e, M_v; D)$ *be the maximal advantage* $\mathbf{Adv}^{\text{INT-CTXT}}_{\mathcal{AE}}(\mathcal{A})$*, where the maximum is taken over the adversaries* $\mathcal{A}$ *whose time complexity is at most* $t$ *and with the following restrictions on oracle queries* ($1 \le i \le D$)*:*

- *the number of queries of the type* $(i, \cdot)$ *to the* $\mathcal{O}_{\text{enc}}$ *oracle* ($(i, \cdot)$ *to the* $\mathcal{O}_{\text{verify}}$ *oracle) does not exceed* $Q_e[i]$ ($Q_v[i]$ *resp.);*
- *the total length of the queries* $\sum |m|$ *among queries of the type* $(i, m)$ *to the* $\mathcal{O}_{\text{enc}}$ *oracle* ($\sum |ct|$ *among queries of the type* $(i, ct)$ *to the* $\mathcal{O}_{\text{verify}}$ *oracle) does not exceed* $L_e[i]$ ($L_v[i]$ *resp.);*
- *the maximal length of the query* $\max |m|$ *among queries of the type* $(i, m)$ *to the* $\mathcal{O}_{\text{enc}}$ *oracle* ($\max |ct|$ *among queries of the type* $(i, ct)$ *to the* $\mathcal{O}_{\text{verify}}$ *oracle) does not exceed* $M_e[i]$ ($M_v[i]$ *resp.);*

**Remark 5.** *For the case* $D = 1$ *we obtain the usual* INT-CTXT *model for the integrity of authenticated encryption [6].*

## 2.3 Diffie-Hellman assumptions

In the MODH model (multiple oracle Diffie-Hellman, see [4] for the single-query case) for the key exchange scheme $\mathcal{KE}$ an adversary $\mathcal{A}$ has an access to two oracles $\mathcal{O}^b_{\text{kgen}}$ and $\mathcal{O}_{\text{comb}}$:

- oracle $\mathcal{O}^b_{\text{kgen}}$ generates either random keys of a given length (in case of $b = 0$) or keys generated via key exchange scheme (in case of $b = 1$) with some restrictions that exclude trivial attacks, see below;
- oracle $\mathcal{O}_{\text{comb}}(epk)$ generates a key via $\mathcal{KE}.\texttt{Combine}$ function using the ephemeral key $epk$.

The goal of $\mathcal{A}$ is to guess which bit $b$ is fixed within the $\mathcal{O}^b_{\text{kgen}}$-oracle. The inability of $\mathcal{A}$ to predict $b$ correctly means that $\mathcal{KE}$-keys are indistinguishable from random ones.

**Definition 7.** *The advantage of the adversary $\mathcal{A}$ in the* MODH *is the following quantity:*

$$\mathbf{Adv}^{\text{MODH}}_{\mathcal{KE}}(\mathcal{A}) = \mathbb{P}\big[\mathbf{Exp}^{\text{MODH-1}}_{\mathcal{KE}}(\mathcal{A}) \to 1\big] - \mathbb{P}\big[\mathbf{Exp}^{\text{MODH-0}}_{\mathcal{KE}}(\mathcal{A}) \to 1\big],$$

*the pseudocode of $\mathbf{Exp}^{\text{MODH-}b}_{\mathcal{KE}}$, $b \in \{0,1\}$, is given in Fig. 4.*

$\mathbf{Exp}^{\text{MODH-}b}_{\mathcal{KE}}(\mathcal{A})$

$(sk, pk) \xleftarrow{\$} \mathcal{KE}.\texttt{KeyPairGen}()$
$Keys \leftarrow [\,]$
$b' \xleftarrow{\$} \mathcal{A}^{\mathcal{O}^b_{\text{kgen}}, \mathcal{O}_{\text{comb}}}(pk)$
**return** $b'$

$\mathcal{O}_{\text{comb}}(epk)$

**if** $Keys[epk] = \perp$
  **return** $\mathcal{KE}.\texttt{Combine}(sk, epk)$
**else**
  **return** $Keys[epk]$
**fi**

$\mathcal{O}^b_{\text{kgen}}()$

$(esk, epk) \xleftarrow{\$} \mathcal{KE}.\texttt{KeyPairGen}()$
**if** $Keys[epk] = \perp$
  $k \leftarrow \mathcal{KE}.\texttt{Combine}(sk, epk)$
  **if** $(b = 0)$
    $k \xleftarrow{\$} \{0,1\}^{|k|}$
  **fi**
  $Keys[epk] \leftarrow k$
**fi**
**return** $(epk, Keys[epk])$

Figure 4: Pseudocode of the MODH Experiment

**Definition 8.** *Let* $\mathbf{InSec}^{\text{MODH}}_{\mathcal{KE}}(t, q_{gen}, q_{com})$ *be the maximal advantage* $\mathbf{Adv}^{\text{MODH}}_{\mathcal{KE}}(\mathcal{A})$*, where the maximum is taken over the adversaries $\mathcal{A}$ whose time complexity is at most $t$, making at most $q_{gen}$ queries to $\mathcal{O}^b_{\text{kgen}}$, $q_{com}$ queries to $\mathcal{O}_{\text{comb}}$ oracles.*

## 2.4 Models for ECIES scheme

In this section we introduce models for confidentiality and integrity for the ECIES scheme. The models are essentially the same as for AE-schemes with the addition of ephemeral key generation.

**Definition 9.** *The advantage of the adversary $\mathcal{A}$ in the* LOR-CCA *model for the* ECIES *scheme is defined as:*

$$\mathbf{Adv}^{\text{LOR-CCA}}_{\text{ECIES}}(\mathcal{A}) = \mathbb{P}\big[\mathbf{Exp}^{\text{LOR-CCA-1}}_{\text{ECIES}}(\mathcal{A}) \to 1\big] - \mathbb{P}\big[\mathbf{Exp}^{\text{LOR-CCA-0}}_{\text{ECIES}}(\mathcal{A}) \to 1\big],$$

*the pseudocode of $\mathbf{Exp}^{\text{LOR-CCA-}b}_{\text{ECIES}}$, $b \in \{0,1\}$, is given in Fig. 5.*

$\mathbf{Exp}^{\text{LOR-CCA-}b}_{\text{ECIES}}(\mathcal{A})$

$(sk, pk) \xleftarrow{\$} \mathcal{KE}.\texttt{KeyPairGen}()$
$sent \leftarrow [\,]$
$b' \xleftarrow{\$} \mathcal{A}^{\mathcal{O}^b_{\text{enc}}, \mathcal{O}_{\text{dec}}}(pk)$
**return** $b'$

$\mathcal{O}^b_{\text{enc}}(m_0, m_1)$

$(epk, esk) \xleftarrow{\$} \mathcal{KE}.\texttt{KeyPairGen}()$
$k \leftarrow \mathcal{KE}.\texttt{Combine}(sk, epk)$
$ct \xleftarrow{\$} \mathcal{AE}.\texttt{Enc}(k, m_b)$
$sent \leftarrow sent \cup \{(epk, ct)\}$
**return** $(epk, ct)$

$\mathcal{O}_{\text{dec}}(epk, ct)$

**if** $(epk, ct) \in sent$
  **return** $\perp$
**fi**
$k \leftarrow \mathcal{KE}.\texttt{Combine}(sk, epk)$
**return** $\mathcal{AE}.\texttt{Dec}(k, ct)$

Figure 5: Pseudocode of the LOR-CCA Experiment for ECIES

**Definition 10.** *Let* $\mathbf{InSec}^{\text{LOR-CCA}}_{\text{ECIES}}(t, q_e, q_d, l_e, l_d, \mu_e, \mu_d)$ *be the maximal advantage* $\mathbf{Adv}^{\text{LOR-CCA}}_{\text{ECIES}}(\mathcal{A})$*, where the maximum is taken over the adversaries* $\mathcal{A}$ *whose time complexity is at most* $t$ *and with the following restrictions on the oracle queries:*

- *the number of queries to the* $\mathcal{O}^b_{\text{enc}}$ *oracle (to the* $\mathcal{O}_{\text{dec}}$ *oracle) does not exceed* $q_e$ *(*$q_d$ *resp.);*
- *the total length of the queries* $\sum |m_0| = \sum |m_1|$ *to the* $\mathcal{O}^b_{\text{enc}}$ *oracle (*$\sum |ct|$ *to the* $\mathcal{O}_{\text{dec}}$ *oracle) does not exceed* $l_e$ *(*$l_d$ *resp.);*
- *the maximal length of the query* $\max |m_0| = \max |m_1|$ *among queries to the* $\mathcal{O}^b_{\text{enc}}$ *oracle (*$\max |ct|$ *among queries to the* $\mathcal{O}_{\text{dec}}$ *oracle) does not exceed* $\mu_e$ *(*$\mu_d$ *resp.);*

**Definition 11.** *The advantage of the adversary* $\mathcal{A}$ *in the* INT-CTXT *model for the* ECIES *scheme is the following quantity:*

$$\mathbf{Adv}^{\text{INT-CTXT}}_{\text{ECIES}}(\mathcal{A}) = \mathbb{P}\big[\mathbf{Exp}^{\text{INT-CTXT}}_{\text{ECIES}}(\mathcal{A}) \to 1\big],$$

*the pseudocode of* $\mathbf{Exp}^{\text{INT-CTXT}}_{\text{ECIES}}$*,* $b \in \{0, 1\}$*, is given in Fig. 6.*

$\mathbf{Exp}^{\text{INT-CTXT}}_{\text{ECIES}}(\mathcal{A})$

$(sk, pk) \xleftarrow{\$} \mathcal{KE}.\texttt{KeyPairGen}()$
$sent \leftarrow [\,]$
$win \leftarrow \texttt{false}$
$\mathcal{A}^{\mathcal{O}_{\text{enc}}, \mathcal{O}_{\text{verify}}}(pk)$
**return** $win$

$\mathcal{O}_{\text{enc}}(m)$

$(epk, esk) \xleftarrow{\$} \mathcal{KE}.\texttt{KeyPairGen}()$
$k \leftarrow \mathcal{KE}.\texttt{Combine}(sk, epk)$
$ct \xleftarrow{\$} \mathcal{AE}.\texttt{Enc}(k, m)$
$sent[epk] \leftarrow sent[epk] \cup \{ct\}$
**return** $(epk, ct)$

$\mathcal{O}_{\text{verify}}(epk, ct)$

$k \leftarrow \mathcal{KE}.\texttt{Combine}(sk, epk)$
$m \leftarrow \mathcal{AE}.\texttt{Dec}(k, ct)$
$t_1 \leftarrow (m \neq \perp)$
$t_2 \leftarrow (sent[epk] \neq \perp)$
$t_3 \leftarrow (ct \notin sent[epk])$
**if** $t_1 \& t_2 \& t_3$
  $win \leftarrow \texttt{true}$
**fi**
**return** $m$

Figure 6: Pseudocode of the INT-CTXT Experiment for ECIES

**Definition 12.** *Let* $\mathbf{InSec}^{\text{INT-CTXT}}_{\text{ECIES}}(t, q_e, q_v, l_e, l_v, \mu_e, \mu_v)$ *be the maximal advantage* $\mathbf{Adv}^{\text{INT-CTXT}}_{\text{ECIES}}(\mathcal{A})$*, where the maximum is taken over the adversaries* $\mathcal{A}$ *whose time complexity is at most* $t$ *and with the following restrictions on the oracle queries:*

- *the number of queries to the* $\mathcal{O}_{\text{enc}}$ *oracle (to the* $\mathcal{O}_{\text{verify}}$ *oracle) does not exceed* $q_e$ *(*$q_v$ *resp.);*
- *the total length of the queries* $\sum |m_0| = \sum |m_1|$ *to the* $\mathcal{O}_{\text{enc}}$ *oracle (*$\sum |ct|$ *to the* $\mathcal{O}_{\text{verify}}$ *oracle) does not exceed* $l_e$ *(*$l_v$ *resp.);*
- *the maximal length of the query* $\max |m_0| = \max |m_1|$ *among queries to the* $\mathcal{O}_{\text{enc}}$ *oracle (*$\max |ct|$ *among queries to the* $\mathcal{O}_{\text{verify}}$ *oracle) does not exceed* $\mu_e$ *(*$\mu_v$ *resp.);*

# 3 Reducing multi-user setting to the single-user case

In this section we show that multi-user setting for LOR-CCA, INT-CTXT models (and $q_{gen}$-queries case for MODH model) can be reduced to the single-user case for LOR-CCA, INT-CTXT (and one query-case for MODH). The very basic form of hybrid argument is used for each of the reductions. In essence, we show that for any adversary $\mathcal{A}$ in the $D$-user setting we can construct a series of the adversaries $\mathcal{B}_1, \ldots, \mathcal{B}_D$ in the single-user model and bound the advantage of $\mathcal{A}$ in terms of $\mathcal{B}_i$-advantages.

**Proposition 1.** *The following inequality holds:*

$$\mathbf{InSec}^{\text{LOR-CCA}}_{\mathcal{AE}}(t, Q_e, Q_d, L_e, L_d, M_e, M_d; D) \leq \\ \leq D \cdot \mathbf{InSec}^{\text{LOR-CCA}}_{\mathcal{AE}}(t + T, q_e, q_d, l_e, l_d, \mu_e, \mu_d; 1),$$

*where the following notation is used:*

- $T = D + \sum_{i=1}^{D} (Q_e[i] + Q_d[i] + L_e[i] + L_d[i])$,
- $l_x = \max_{1 \leq i \leq D} L_x[i]$, $\mu_x = \max_{1 \leq i \leq D} M_x[i]$, $q_x = \max_{1 \leq i \leq D} Q_x[i]$, $x \in \{e, d\}$.

*Proof.* Let $\mathcal{A}^{b_1 b_2 \ldots b_D}$, $b_i \in \{0, 1\}$, be the adversary in the $\mathbf{Exp}^{\text{LOR-CCA}}_{\mathcal{AE}}$-experiment, such that on the queries of the type $(i, \cdot, \cdot)$ the oracle $\mathcal{O}_{\text{enc}}$ chooses message $m_{b_i}$ to be encrypted. Then $\mathcal{A}$'s advantage can be written as:

$$\mathbf{Adv}^{\text{LOR-CCA}}_{\mathcal{AE}}(\mathcal{A}) = \mathbb{P}\left[\mathcal{A}^{11\ldots1} \to 1\right] - \mathbb{P}\left[\mathcal{A}^{00\ldots0} \to 1\right] = \\ = \left(\mathbb{P}\left[\mathcal{A}^{11\ldots1} \to 1\right] - \mathbb{P}\left[\mathcal{A}^{11\ldots0} \to 1\right]\right) + \ldots \\ \ldots + \left(\mathbb{P}\left[\mathcal{A}^{\underbrace{1\ldots1}_{s}\underbrace{0\ldots0}_{D-s}} \to 1\right] - \mathbb{P}\left[\mathcal{A}^{\underbrace{1\ldots1}_{s-1}\underbrace{0\ldots0}_{D-s+1}} \to 1\right]\right) + \ldots \\ \ldots + \left(\mathbb{P}\left[\mathcal{A}^{00\ldots1} \to 1\right] - \mathbb{P}\left[\mathcal{A}^{00\ldots0} \to 1\right]\right).$$

Each of the summands of the form

$$\left(\mathbb{P}\left[\mathcal{A}^{\underbrace{1\ldots1}_{s}\underbrace{0\ldots0}_{D-s}} \to 1\right] - \mathbb{P}\left[\mathcal{A}^{\underbrace{1\ldots1}_{s-1}\underbrace{0\ldots0}_{D-s+1}} \to 1\right]\right)$$

can be upper bounded by the value

$$\mathbf{InSec}^{\text{LOR-CCA}}_{\mathcal{AE}}(t_{\mathcal{A}} + T, Q_e[s], Q_d[s], L_e[s], L_d[s], M_e[s], M_d[s]; 1), \quad (1)$$

$T$ is the time needed to simulate LOR-CCA-experiment with $D$ parties. In order to show this we construct adversaries $\mathcal{B}_s$, $1 \leq s \leq D$, in the LOR-CCA model with $D = 1$ user. $\mathcal{B}_s$ generates keys $k_i$, $1 \leq i \leq D$, $i \neq s$, responds to the $\mathcal{A}$-queries (see Table 1, decryption queries are processed similarly) and returns the same bit $b'$ as $\mathcal{A}$ at the end of the Experiment.

Table 1: Encryption query simulations in hybrid experiments

| $(1, m_0, m_1), \ldots, (s-1, m_0, m_1)$ | $(s, m_0, m_1)$ | $(s+1, m_0, m_1), \ldots, (D, m_0, m_1)$ |
|---|---|---|
| Simulated via choosing $m_1$ and encrypting under $k_i$ | Oracle queries | Simulated via choosing $m_0$ and encrypting under $k_i$ |
| $1 \leq i \leq s-1$ | to $\mathcal{O}^b_{\text{enc}}(m_0, m_1)$ | $s+1 \leq i \leq D$ |

By definition of the Experiments we have:

$$\mathbb{P}[\mathcal{B}_s \to 1] = \mathbb{P}\left[\mathcal{A}^{\underbrace{1\ldots1}_{s}\underbrace{0\ldots0}_{D-s}} \to 1\right], \quad \mathbb{P}[\mathcal{B}_s \to 0] = \mathbb{P}\left[\mathcal{A}^{\underbrace{1\ldots1}_{s-1}\underbrace{0\ldots0}_{D-s+1}} \to 1\right],$$

hence the bound (1). The time $T$ can be upper bounded as follows (generating $D-1$ keys, query processing):

$$T \le D + \sum_{i=1}^{D} \left(Q_e[i] + Q_d[i] + L_e[i] + L_d[i]\right).$$

Collecting all of the estimates and setting $l_x = \max_{1\le i\le D} L_x[i]$, $\mu_x = \max_{1\le i\le D} M_x[i]$, $q_x = \max_{1\le i\le D} Q_x[i]$, $x \in \{e, d\}$, we obtain the final result:

$$\begin{aligned}\mathbf{InSec}^{\text{LOR-CCA}}_{\mathcal{AE}}(t, Q_e, Q_d, L_e, L_d, M_e, M_d; D) \le \\ \le D \cdot \mathbf{InSec}^{\text{LOR-CCA}}_{\mathcal{AE}}(t+T, q_e, q_d, l_e, l_d, \mu_e, \mu_d; 1),\end{aligned}$$

where $T = D + \sum_{i=1}^{D} \left(Q_e[i] + Q_d[i] + L_e[i] + L_d[i]\right)$. □

**Proposition 2.** *The following inequality holds:*

$$\begin{aligned}\mathbf{InSec}^{\text{INT-CTXT}}_{\mathcal{AE}}(t, Q_e, Q_v, L_e, L_v, M_e, M_v; D) \le \\ \le D \cdot \mathbf{InSec}^{\text{INT-CTXT}}_{\mathcal{AE}}(t+T, q_e, q_v, l_e, l_v, \mu_e, \mu_v; 1),\end{aligned}$$

*where the following notation is used:*

- $T = D + \sum_{i=1}^{D} \left(Q_e[i] + Q_v[i] + L_e[i] + L_v[i]\right)$,
- $l_x = \max_{1\le i\le D} L_x[i]$, $\mu_x = \max_{1\le i\le D} M_x[i]$, $q_x = \max_{1\le i\le D} Q_x[i]$, $x \in \{e, v\}$.

*Proof.* Let $\mathcal{A}$ be the adversary in the INT-CTXT model with $D$ parties. Let us construct the adversary $\mathcal{B}$ in the INT-CTXT model in the single-user setting that uses $\mathcal{A}$ as a subroutine and achieves advantage $\frac{1}{D}\mathbf{Adv}^{\text{INT-CTXT}}_{\mathcal{AE}}(\mathcal{A})$.

The adversary $\mathcal{B}$ performs the following actions:

- chooses $s \xleftarrow{\$} \{1, \ldots, D\}$;
- generates $k_i \xleftarrow{\$} \texttt{KeyGen}$, $1 \le i \le D$, $i \ne s$;
- processes $\mathcal{A}$-queries of the form $(i, \cdot)$, $1 \le i \le D$, $i \ne s$ as it is described in the INT-CTXT-Experiment; queries of the form $(s, \cdot)$ are redirected to the $\mathcal{B}$'s oracles.

The adversary $\mathcal{B}$ forges successfully if and only if $\mathcal{A}$ succesfully forges $ct$ for the $s$-th party. Since $s$ was chosen independently uniformly at random, and keys $k_i$, $1 \le i \le D$, are I.I.D., we have that:

$$\mathbf{Adv}^{\text{INT-CTXT}}_{\mathcal{AE}}(\mathcal{B}) = \frac{1}{D}\,\mathbf{Adv}^{\text{INT-CTXT}}_{\mathcal{AE}}(\mathcal{A}).$$

Number of $\mathcal{B}$-oracle queries does not exceed $l_x = \max_{1\le i\le D} L_x[i]$, $\mu_x = \max_{1\le i\le D} M_x[i]$, $q_x = \max_{1\le i\le D} Q_x[i]$, $x \in \{e, v\}$ respectively; time complexity of $\mathcal{B}$ does not exceed

$$t_{\mathcal{A}} + D + \sum_{i=1}^{D} \left(Q_e[i] + Q_v[i] + L_e[i] + L_v[i]\right),$$

hence the result. □

**Proposition 3.** *Assume that the distribution of ephemeral public keys epk generated by $\mathcal{KE}$.*`KeyPairGen` *is uniformly random on EpkSet. Then the following inequality holds:*

$$\mathbf{InSec}^{\text{MODH}}_{\mathcal{KE}}(t, q_{gen}, q_{com}) \le \\ \le q_{gen} \cdot \mathbf{InSec}^{\text{MODH}}_{\mathcal{KE}}(t + q_{gen} + q_{com}, 1, q_{com}) + \frac{2\, q_{gen}\, q_{com}}{|EpkSet|},$$

*Proof.* The idea is essentially the same as in Proposition 1, with the following addition. The problem may arise if the key *epk* generated inside $\mathcal{O}^b_{\text{kgen}}$ collides with one of the keys *epk* queried by $\mathcal{A}$ to $\mathcal{O}_{\text{comb}}$ oracle. Let us denote by **Coll** the probability that the collision of this type happened during the MODH-experiment. Then, we can bound the probability of such a collision as:

$$\mathbb{P}[\mathbf{Coll}] \le \frac{q_{gen} \cdot q_{com}}{|EpkSet|}.$$

The reasoning is the following: if the number of queries to $\mathcal{O}_{\text{comb}}$ does not exceed $q_{com}$, then the probability of collision when choosing *epk* at random does not exceed $\frac{q_{com}}{|EpkSet|}$, where $EpkSet$ is a set of ephemeral keys (assuming the uniform distribution of ephemeral keys).

Let us denote by $\widetilde{\text{MODH}}$ the modified model, where the Experiment MODH halts whenever the collision happens. Then it holds that:

$$|\mathbf{Adv}^{\widetilde{\text{MODH}}}_{\mathcal{KE}}(\mathcal{A}) - \mathbf{Adv}^{\text{MODH}}_{\mathcal{KE}}(\mathcal{A})| \le \mathbb{P}[\mathbf{Coll}].$$

Denote by $\mathcal{A}^{b_1 b_2 \ldots b_D}$, $b_i \in \{0, 1\}$, the adversary in the $\mathbf{Exp}^{\widetilde{\text{MODH}}}_{\mathcal{KE}}$-experiment, such that on the $i$-th query the oracle $\mathcal{O}_{\text{kgen}}$ behaves as if the bit

$b_i$ was fixed in it. Then the advantage of $\mathcal{A}$ can be expressed as follows:

$$\begin{aligned}\mathbf{Adv}^{\widetilde{\mathrm{MODH}}}_{\mathcal{KE}}(\mathcal{A}) &= \mathbb{P}\left[\mathcal{A}^{11\ldots1} \to 1\right] - \mathbb{P}\left[\mathcal{A}^{00\ldots0} \to 1\right] = \\ &= \left(\mathbb{P}\left[\mathcal{A}^{11\ldots1} \to 1\right] - \mathbb{P}\left[\mathcal{A}^{11\ldots0} \to 1\right]\right) + \ldots \\ \ldots + \left(\mathbb{P}\left[\mathcal{A}^{\underbrace{1\ldots1}_{s}\underbrace{0\ldots0}_{D-s}} \to 1\right] - \mathbb{P}\left[\mathcal{A}^{\underbrace{1\ldots1}_{s-1}\underbrace{0\ldots0}_{D-s+1}} \to 1\right]\right) &+ \ldots \\ \ldots + \left(\mathbb{P}\left[\mathcal{A}^{00\ldots1} \to 1\right] - \mathbb{P}\left[\mathcal{A}^{00\ldots0} \to 1\right]\right)&,\end{aligned}$$

where $D = q_{gen}$. We now bound each of the summands of the form

$$\left(\mathbb{P}\left[\mathcal{A}^{\underbrace{1\ldots1}_{s}\underbrace{0\ldots0}_{D-s}} \to 1\right] - \mathbb{P}\left[\mathcal{A}^{\underbrace{1\ldots1}_{s-1}\underbrace{0\ldots0}_{D-s+1}} \to 1\right]\right).$$

In order to do this we construct adversaries $\mathcal{B}_s$, $1 \le s \le D$, in the MODH model with $q_{gen} = 1$ query to $\mathcal{O}^b_{\mathrm{kgen}}$ oracle. $\mathcal{B}_s$ initializes empty associative array $Keys$ and responds to the $\mathcal{A}$-queries as follows:

1. $\mathcal{A}$'s query $epk$ to the $\mathcal{O}_{\mathrm{comb}}$ oracle:
   - if $Keys[epk] = \perp$: queries $\mathcal{B}$'s oracle $k \leftarrow \mathcal{O}_{\mathrm{comb}}(epk)$ and sets $Keys[epk] \leftarrow k$;
   - returns $Keys[epk]$.
2. $\mathcal{A}$'s $i$-th query to $\mathcal{O}^b_{\mathrm{kgen}}$, $i < s$:
   - $\mathcal{B}_s$ generates $(esk, epk) \xleftarrow{\$} \mathcal{KE}.\mathtt{KeyPairGen}$;
   - if $Keys[epk] \neq \perp$: the Experiment **halts**;
   - sets $Keys[epk] \leftarrow \mathcal{KE}.\mathtt{Combine}(esk, pk)$;
   - returns $Keys[epk]$.
3. $\mathcal{A}$'s $s$-th query to $\mathcal{O}^b_{\mathrm{kgen}}$: $\mathcal{B}_s$ queries $\mathcal{B}$'s oracle $(epk, k) \leftarrow \mathcal{O}^b_{\mathrm{kgen}}$; if $Keys[epk] \neq \perp$: the Experiment **halts**; otherwise, sets $Keys[epk] \leftarrow k$ and returns the result to the $\mathcal{A}$.
4. $\mathcal{A}$'s $i$-th query to $\mathcal{O}^b_{\mathrm{kgen}}$, $i > s$:
   - $\mathcal{B}_s$ generates $(esk, epk) \xleftarrow{\$} \mathcal{KE}.\mathtt{KeyPairGen}$;
   - if $Keys[epk] \neq \perp$: the Experiment **halts**;
   - sets $Keys[epk] \leftarrow \{0,1\}^{klen}$ and returns $Keys[epk]$ to the $\mathcal{A}$.

The adversary $\mathcal{B}_s$ returns the same bit $b'$ as $\mathcal{A}$ at the end of the Experiment. $\mathcal{B}_s^b$ simulates the Experiment $\mathcal{A}^{\underbrace{1\ldots1}_{s-1}b\underbrace{0\ldots0}_{D-s}}$, hence:

$$\left(\mathbb{P}\left[\mathcal{A}^{\underbrace{1\ldots1}_{s}\underbrace{0\ldots0}_{D-s}}\to 1\right]-\mathbb{P}\left[\mathcal{A}^{\underbrace{1\ldots1}_{s-1}\underbrace{0\ldots0}_{D-s+1}}\to 1\right]\right)\leq$$

$$\leq \mathbf{Adv}^{\widetilde{\mathrm{MODH}}}_{\mathcal{KE}}(\mathcal{B}_s)\leq \mathbf{Adv}^{\mathrm{MODH}}_{\mathcal{KE}}(\mathcal{B}_s)+\mathbb{P}[\mathbf{Coll}]\leq$$

$$\leq \mathbf{InSec}^{\mathrm{MODH}}_{\mathcal{KE}}(t+q_{gen}+q_{com},1,q_{com})+\frac{q_{com}}{|EpkSet|}.$$

Collecting all of the estimates, we obtain the desired bound. □

# 4 Security reduction for ECIES scheme

In this section we show that the security of ECIES scheme in LOR-CCA and INT-CTXT models follows from the security of $\mathcal{KE}$ in MODH model and $\mathcal{AE}$ in LOR-CCA and INT-CTXT models respectively. The basic strategy of the proof is the same as in [6]: we replace $\mathcal{KE}$-keys with independent random keys and then analyze ECIES scheme in the obtained model. Because of the multiple queries to the $\mathcal{O}_{\mathrm{enc}}$-oracles, we obtain LOR-CCA and INT-CTXT models with $D = q_e$ parties and MODH model, which can be reduced to the standard single-user setting (see Section 3).

## 4.1 Security reduction for the LOR-CCA model

**Theorem 1.** *Assume that the distribution of ephemeral public keys epk generated by* $\mathcal{KE}$.`KeyPairGen` *is uniformly random on* $EpkSet$*. Then the following inequality holds:*

$$\mathbf{InSec}^{\text{LOR-CCA}}_{\text{ECIES}}(t,q_e,q_d,l_e,l_d,\mu_e,\mu_d)\leq$$

$$\leq 2\cdot\mathbf{InSec}^{\mathrm{MODH}}_{\mathcal{KE}}(t+T_1,q_e,q_d)+$$
$$+\,q_e\cdot\mathbf{InSec}^{\text{LOR-CCA}}_{\mathcal{AE}}(t+T_2,q_e,q_d,l_e,l_d,\mu_e,\mu_d;1)+\frac{q_e\cdot q_d}{|EpkSet|},$$

*where* $T_1 = q_e + q_d + l_e + l_d$, $T_2 = q_d + l_d + q_e\,(q_e + q_d + l_e + l_d + 2)$.

*Proof.* Let $\mathcal{A}$ be the adversary for the ECIES scheme in the LOR-CCA

model, then its advantage can be decomposed as follows:

$$\begin{aligned}\mathbf{Adv}^{\text{LOR-CCA}}_{\text{ECIES}}(\mathcal{A}) &= \mathbb{P}\big[\mathcal{A}^1 \to 1\big] - \mathbb{P}\big[\mathcal{A}^0 \to 1\big] = \\ &= \big(\mathbb{P}\big[\mathcal{A}^1 \to 1\big] - \mathbb{P}\big[\mathcal{A}^1_{\$} \to 1\big]\big) + \big(\mathbb{P}\big[\mathcal{A}^1_{\$} \to 1\big] - \mathbb{P}\big[\mathcal{A}^0_{\$} \to 1\big]\big) + \\ &\qquad + \big(\mathbb{P}\big[\mathcal{A}^0_{\$} \to 1\big] - \mathbb{P}\big[\mathcal{A}^0 \to 1\big]\big),\end{aligned}$$

where the following notation is used:

- $\mathcal{A}^b$: the adversary $\mathcal{A}$ interacting with the Experiment $\mathbf{Exp}^{\text{LOR-CCA-}b}_{\text{ECIES}}$;
- $\mathcal{A}^b_{\$}$: the adversary $\mathcal{A}$ interacting with the Experiment $\widetilde{\mathbf{Exp}}^{\text{LOR-CCA-}b}_{\text{ECIES}}$ (see the pseudocode at Fig. 7, LOR-CCA-experiment with random keys).

In this decomposition we do the two-step reduction: firstly, we replace $\mathcal{KE}$-keys with randomly generated; secondly, we analyze the scheme with $q_e$ randomly chosen independent keys.

$\mathcal{O}^b_{\text{enc}}(m_0, m_1)$

$(epk, esk) \xleftarrow{\$} \mathcal{KE}.\texttt{KeyPairGen}()$
**if** $Keys[epk] = \perp$
  $Keys[epk] \xleftarrow{\$} \mathcal{AE}.\texttt{KeyGen}()$
**fi**
$k \leftarrow Keys[epk]$
$ct \xleftarrow{\$} \mathcal{AE}.\texttt{Enc}(k, m_b)$
$sent \leftarrow sent \cup \{(epk, ct)\}$
**return** $(epk, ct)$

$\mathcal{O}_{\text{dec}}(epk, ct)$

**if** $((epk, ct) \in sent)$
  **return** $\perp$
**fi**
**if** $Keys[epk] = \perp$
  $Keys[epk] \leftarrow \mathcal{KE}.\texttt{Combine}(epk, sk)$
**fi**
$k \leftarrow Keys[epk]$
**return** $\mathcal{AE}.\texttt{Dec}(k, ct)$

Figure 7: Pseudocode of the LOR-CCA Experiment with random keys ($\widetilde{\mathbf{Exp}}^{\text{LOR-CCA-}b}_{\text{ECIES}}$) for ECIES

The main difference from $\mathbf{Exp}^{\text{LOR-CCA-}b}_{\text{ECIES}}$ consists in choosing $Keys[epk]$ according to the $\mathcal{AE}.\texttt{KeyGen}$ algorithm.

Let us estimate the value $\varepsilon = \big(\mathbb{P}\big[\mathcal{A}^1 \to 1\big] - \mathbb{P}\big[\mathcal{A}^1_{\$} \to 1\big]\big)$. Define the adversary $\mathcal{B}$ in the MODH model as follows.

1. When $\mathcal{A}$ makes an $\mathcal{O}^1_{\text{enc}}$-query of the form $(m_0, m_1)$, $\mathcal{B}$ does the following:
   - queries $(epk, k) \xleftarrow{\$} \mathcal{O}^b_{\text{kgen}}$;
   - processes $m_1$ on the key $k$: $ct \xleftarrow{\$} \mathcal{AE}.\texttt{Enc}(k, m_1)$;
   - stores the values $sent \leftarrow sent \cup \{(epk, ct)\}$ and $Keys[epk] \leftarrow k$;

– returns $(epk, ct)$.

2. When $\mathcal{A}$ makes an $\mathcal{O}_{\text{dec}}$-query of the form $(epk, ct)$:
   – if $(epk, ct) \in sent$: returns $\perp$;
   – if $Keys[epk] = \perp$, asks $k \leftarrow \mathcal{O}_{\text{comb}}(epk)$ and stores $Keys[epk] \leftarrow k$;
   – decrypt $ct$ using $k \leftarrow Keys[epk]$: $m \leftarrow \mathcal{AE}.\texttt{Dec}(k, ct)$, returns $m$.

Adversary $\mathcal{B}$ returns the same bit as $\mathcal{A}$. Whenever the bit $b = 1$ is fixed in the MODH model (i.e., $\mathcal{B}$ queries the $\mathcal{O}^1_{\text{kgen}}$ oracle), $\mathcal{B}$ simulates the Experiment $\mathbf{Exp}^{\text{LOR-CCA-1}}_{\text{ECIES}}$ for $\mathcal{A}$, hence $\mathbb{P}\big[\mathcal{B}^1 \to 1\big] = \mathbb{P}\big[\mathcal{A}^1 \to 1\big]$. Analogously, if the bit $b = 0$ is fixed in the MODH model (i.e., $\mathcal{B}$ queries the $\mathcal{O}^0_{\text{kgen}}$ oracle), $\mathcal{B}$ simulates the Experiment $\widetilde{\mathbf{Exp}}^{\text{LOR-CCA-1}}_{\text{ECIES}}$ (i.e. LOR-CCA with random keys) for $\mathcal{A}$, hence $\mathbb{P}\big[\mathcal{B}^0 \to 1\big] = \mathbb{P}\big[\mathcal{A}^1_{\$} \to 1\big]$.

The same line of reasoning (replacing $m_1 \to m_0$) works for the term $\big(\mathbb{P}\big[\mathcal{A}^0_{\$} \to 1\big] - \mathbb{P}\big[\mathcal{A}^0 \to 1\big]\big)$. Hence, $\big|\mathbb{P}\big[\mathcal{A}^b \to 1\big] - \mathbb{P}\big[\mathcal{A}^b_{\$} \to 1\big]\big|$, $b \in \{0, 1\}$, can be upper bounded by $\mathbf{InSec}^{\text{MODH}}_{\mathcal{KE}}(t + q_e + q_d + l_e + l_d, q_e, q_d)$.

Now let us consider the last summand $\big(\mathbb{P}\big[\mathcal{A}^1_{\$} \to 1\big] - \mathbb{P}\big[\mathcal{A}^0_{\$} \to 1\big]\big)$. We will show that it can be upper bounded by the insecurity of $\mathcal{AE}$ in the LOR-CCA model with $D = q_e$ parties.

In order to do that let us consider the adversary $\mathcal{B}$ in LOR-CCA model acting as follows. It generates key pair $(sk, pk) \xleftarrow{\$} \mathcal{KE}.\texttt{KeyPairGen}$ and gives $pk$ to $\mathcal{A}$. Also it generates empty associative arrays $Keys$, $GenKeys$ and sets $i \leftarrow 1$.

1. When $\mathcal{A}$ makes an $\mathcal{O}^b_{\text{enc}}$ query of the form $(m_0, m_1)$:
   – $\mathcal{B}$ generates $(esk, epk) \xleftarrow{\$} \mathcal{KE}.\texttt{KeyPairGen}$;
   – if $GenKeys[epk] \neq \perp$, then the Experiment halts;
   – if $Keys[epk] = \perp$, it sets $Keys[epk] \leftarrow i$ and increments $i \leftarrow i + 1$;
   – it queries its encryption oracle $ct \xleftarrow{\$} \mathcal{O}^b_{\text{enc}}(Keys[epk], m_0, m_1)$;
   – $\mathcal{B}$ returns the pair $(epk, ct)$ to $\mathcal{A}$.

2. When $\mathcal{A}$ makes an $\mathcal{O}_{\text{dec}}$ query of the form $(epk, ct)$:
   – if $Keys[epk] \neq \perp$, $\mathcal{B}$ queries its own oracle $m \leftarrow \mathcal{O}_{\text{dec}}(Keys[epk], ct)$ and returns $m$;
   – if $epk$ was not the part of any query before, then $\mathcal{B}$ generates $GenKeys[epk] \leftarrow \mathcal{KE}.\texttt{Combine}(epk, sk)$, processes $ct$ using $GenKeys[epk]$ and returns the result to the $\mathcal{A}$.

The adversary $\mathcal{B}$ returns the same bit as $\mathcal{A}$. The caveat here is that the Experiment halts whenever $\mathcal{A}$ asks *epk* via $\mathcal{O}_{\text{dec}}$ *before* it was generated in $\mathcal{O}^b_{\text{enc}}$-oracle query (see also the proof of Proposition 3). This collision is undesirable, because $\mathcal{B}$ "does not know" which of the messages $m_0$ or $m_1$ it should process (it depends on the value of the bit $b$ in his own oracle). Let us denote by **Coll** the event that the Experiment halts. We have the following equality:

$$\mathbb{P}\big[\mathcal{B}^1 \to 1\big] = \mathbb{P}\big[\mathcal{A}^1_{\$} \to 1 \cap \overline{\mathbf{Coll}}\big], \quad \mathbb{P}\big[\mathcal{B}^0 \to 1\big] = \mathbb{P}\big[\mathcal{A}^0_{\$} \to 1 \cap \overline{\mathbf{Coll}}\big].$$

Hence, it holds that

$$\begin{aligned}&\big(\mathbb{P}\big[\mathcal{A}^1_{\$} \to 1\big] - \mathbb{P}\big[\mathcal{A}^0_{\$} \to 1\big]\big) \leq \\ &\leq \mathbf{InSec}^{\text{LOR-CCA}}_{\mathcal{AE}}\left(t + q_e + q_d + l_d, Q_e, Q_d, L_e, L_d, M_e, M_d; q_e\right) + \mathbb{P}[\mathbf{Coll}],\end{aligned}$$

where $Q_x[i] = q_x$, $L_x[i] = l_x$, $M_x[i] = \mu_x$, $x \in \{e, d\}$, $1 \leq i \leq q_e$.

Using the result from Proposition 1, we have the following inequality:

$$\begin{aligned}\mathbf{InSec}^{\text{LOR-CCA}}_{SE}\left(t + q_e + q_d + l_d, Q_e, Q_d, L_e, L_d, M_e, M_d; q_e\right) \leq \\ \leq q_e \cdot \mathbf{InSec}^{\text{LOR-CCA}}_{\mathcal{AE}}\left(t + T, q_e, q_d, l_e, l_d, \mu_e, \mu_d; 1\right),\end{aligned}$$

where $T = q_d + l_d + q_e\left(q_e + q_d + l_e + l_d + 2\right)$.

Probability $\mathbb{P}[\mathbf{Coll}]$ can be upper bounded using the following reasoning. If the number of queries to $\mathcal{O}_{\text{dec}}$ does not exceed $q_d$, then the probability of collision when choosing *epk* at random does not exceed $\frac{q_d}{|EpkSet|}$, where *EpkSet* is a set of ephemeral keys (assuming the uniform distribution of ephemeral keys), hence $\mathbb{P}[\mathbf{Coll}] \leq \frac{q_e \cdot q_d}{|EpkSet|}$. □

### 4.2 Security reduction for the INT-CTXT model

An analogous theorem with almost identical proof holds for INT-CTXT model.

**Theorem 2.** *Assume that the distribution of ephemeral public keys epk generated by* $\mathcal{KE}$.`KeyPairGen` *is uniformly random on EpkSet. The following inequality holds:*

$$\begin{aligned}\mathbf{InSec}^{\text{INT-CTXT}}_{\text{ECIES}}\left(t, q_e, q_v, l_e, l_v, \mu_e, \mu_v\right) \leq \\ \leq \mathbf{InSec}^{\text{MODH}}_{\mathcal{KE}}\left(t + T_1, q_e, q_v\right) + \\ + q_e \mathbf{InSec}^{\text{INT-CTXT}}_{\mathcal{AE}}\left(t + T_2, q_e, q_v, l_e, l_v, \mu_e, \mu_v; 1\right) + \frac{q_e \cdot q_v}{|EpkSet|},\end{aligned}$$

*where* $T_1 = q_e + q_v + l_e + l_v$, $T_2 = D + q_v + l_v + q_e \cdot (1 + q_e + q_v + l_e + l_v)$.

*Proof.* Let $\mathcal{A}$ be the adversary for ECIES scheme in INT-CTXT model. Decompose the advantage of $\mathcal{A}$ in the following way:

$$\mathbb{P}[\mathcal{A} \to 1] = \mathbb{P}[\mathcal{A} \to 1] - \mathbb{P}[\mathcal{A}_\$ \to 1] + \mathbb{P}[\mathcal{A}_\$ \to 1],$$

where $\mathcal{A}_\$$ — the adversary $\mathcal{A}$ interacting with the Experiment $\widetilde{\mathbf{Exp}}^{\text{INT-CTXT}}_{\text{ECIES}}$ (see the pseudocode at Fig. 8). In this decomposition we do the two-step reduction: firstly, we replace $\mathcal{KE}$-keys with randomly generated; secondly, we analyze the scheme with $q_e$ randomly chosen keys in the INT-CTXT model.

| $\mathcal{O}_{\text{enc}}(m)$ | $\mathcal{O}_{\text{verify}}(epk, ct)$ |
|---|---|
| $(esk, epk) \xleftarrow{\$} \mathcal{KE}.\texttt{KeyPairGen}$ | **if** $Keys[epk] = \perp$ |
| **if** $Keys[epk] = \perp$ | $\quad Keys[epk] \leftarrow \mathcal{KE}.\texttt{Combine}(epk, sk)$ |
| $\quad Keys[epk] \xleftarrow{\$} \mathcal{AE}.\texttt{KeyGen}$ | **fi** |
| **fi** | $k \leftarrow Keys[epk]$ |
| $k \leftarrow Keys[epk]$ | $m \leftarrow \mathcal{AE}.\texttt{Dec}(k, ct)$ |
| $ct \xleftarrow{\$} \mathcal{AE}.\texttt{Enc}(k, m)$ | $t_1 \leftarrow (m \neq \perp)$ |
| $sent[epk] \leftarrow sent[epk] \cup \{ct\}$ | $t_2 \leftarrow (sent[epk] \neq \perp)$ |
| **return** $(epk, ct)$ | $t_3 \leftarrow (ct \notin sent[epk])$ |
| | **if** $t_1 \& t_2 \& t_3$ |
| | $\quad win \leftarrow \texttt{true}$ |
| | **fi** |
| | **return** $m$ |

Figure 8: Pseudocode of the INT-CTXT Experiment for ECIES with random keys

The main difference from $\mathbf{Exp}^{\text{INT-CTXT}}_{\text{ECIES}}$ consists in choosing $Keys[epk]$ according to the $\mathcal{AE}.\texttt{KeyGen}$ algorithm.

Now we estimate the value $\varepsilon = (\mathbb{P}[\mathcal{A} \to 1] - \mathbb{P}[\mathcal{A}_\$ \to 1])$. Let us define the adversary $\mathcal{B}$ in the MODH model.

1. When $\mathcal{A}$ makes an $\mathcal{O}_{\text{enc}}$-query of the form $m$, $\mathcal{B}$ does the following:
   - queries $(epk, k) \xleftarrow{\$} \mathcal{O}^b_{\text{kgen}}$;
   - processes $m$ on the key $k$: $ct \xleftarrow{\$} \mathcal{AE}.\texttt{Enc}(k, m)$;
   - stores the ciphertext $sent[epk] \leftarrow sent[epk] \cup \{ct\}$ and the key $Keys[epk] \leftarrow k$;
   - returns $(epk, ct)$.
2. When $\mathcal{A}$ makes an $\mathcal{O}_{\text{verify}}$-query of the form $(epk, ct)$:
   - if $Keys[epk] = \perp$, queries $k \leftarrow \mathcal{O}_{\text{comb}}(epk)$ and stores $Keys[epk] \leftarrow k$;

– decrypts $ct$ using $k \leftarrow Keys[epk]$: $m \leftarrow \mathcal{AE}.\mathtt{Dec}(k, ct)$ and returns $m$;
– if conditions $t_1$, $t_2$ and $t_3$ are fulfilled, then sets $win \leftarrow \mathtt{true}$.

The adversary $\mathcal{B}$ returns the bit $win$. We have the following equality for the success probability of $\mathcal{B}$:

$$\mathbb{P}\left[\mathcal{B}^1 \to 1\right] = \mathbb{P}[\mathcal{A} \to 1], \quad \mathbb{P}\left[\mathcal{B}^0 \to 1\right] = \mathbb{P}[\mathcal{A}_\$ \to 1].$$

This quantity can be upper bounded by $\mathbf{InSec}^{\mathrm{MODH}}_{\mathcal{KE}}(t + q_e + q_d + l_e + l_d, q_e, q_d)$.

Now let us examine the value $\mathbb{P}[\mathcal{A}_\$ \to 1]$: we will show that it can be upper bounded by insecurity of $\mathcal{AE}$ in the INT-CTXT model. Let us construct $\mathcal{B}$ in INT-CTXT model with $D = q_e$ parties as follows. The adversary $\mathcal{B}$ generates $(sk, pk) \xleftarrow{\$} \mathcal{KE}.\mathtt{KeyPairGen}$ and gives $pk$ to the adversary $\mathcal{A}$. It also generates empty associative arrays $Keys$, $GenKeys$ and sets $i \leftarrow 1$.

1. When $\mathcal{A}$ makes an $\mathcal{O}_{\mathrm{enc}}$ query of the form $m$:
   – $\mathcal{B}$ generates $(esk, epk) \xleftarrow{\$} \mathcal{KE}.\mathtt{KeyPairGen}$;
   – if $GenKeys[epk] \neq \perp$, then the Experiment halts;
   – if $Keys[epk] = \perp$, $\mathcal{B}$ sets $Keys[epk] \leftarrow i$ and increments $i += 1$;
   – $\mathcal{B}$ queries its encryption oracle $ct \xleftarrow{\$} \mathcal{O}_{\mathrm{enc}}(Keys[epk], m)$;
   – the pair $(epk, ct)$ returns to $\mathcal{A}$.
2. When $\mathcal{A}$ makes an $\mathcal{O}_{\mathrm{verify}}$ query of the form $(epk, ct)$:
   – if $Keys[epk] \neq \perp$, $\mathcal{B}$ queries its own oracle $m \leftarrow \mathcal{O}_{\mathrm{dec}}(Keys[epk], ct)$ and returns $m$;
   – if $epk$ was not the part of any query before, then $\mathcal{B}$ generates $GenKeys[epk] \leftarrow \mathcal{KE}.\mathtt{Combine}(epk, sk)$, processes $ct$ using $GenKeys[epk]$ and returns the result to the $\mathcal{A}$.

The adversary $\mathcal{B}$ wins in $\mathbf{Exp}^{\text{INT-CTXT}}_{\mathcal{AE}}$ with $D = q_e$ parties if and only if there are no collisions of $epk$, and $\mathcal{A}$ wins in $\mathbf{Exp}^{\text{INT-CTXT}}_{\text{ECIES}}$, hence (using the Proposition 2):

$$\mathbb{P}[\mathcal{A}_\$ \to 1] \leq \\ \leq q_e \mathbf{InSec}^{\text{INT-CTXT}}_{\mathcal{AE}}(t + T, q_e, q_v, l_e, l_v, \mu_e, \mu_v; 1) + \frac{q_e \cdot q_v}{|EpkSet|},$$

where $T = D + q_v + l_v + q_e \cdot (1 + q_e + q_v + l_e + l_v)$. □

# 5 Concluding remarks

In the paper we study the security of ECIES scheme in the LOR-CCA and INT-CTXT models for confidentiality and integrity respectively. We show that the security of the scheme is based on the security of $\mathcal{KE}$ scheme in the MODH model and the security of AE-scheme $\mathcal{AE}$ in the LOR-CCA and INT-CTXT models.

Further directions of research may include the analysis of the (in)security of specific instances of key exchange schemes in MODH model (e.g., VKO scheme [12] with various idealizations, such as random oracle model [4], generic group model [20, 21], etc), as well as the exact analysis for the AEAD-schemes (see Remark 2).

# References


[1] V. Gayoso Martínez, L. Hernández Encinas, "A comparison of the standardized versions of ECIES", Sixth International Conference on Information Assurance and Security, 2010, 1–4.

[2] V. Gayoso Martínez, L. Hernández Encinas, A. Queiruga Dios, "Security and practical considerations when implementing the elliptic curve integrated encryption scheme", *Cryptologia*, **39**:3 (2015), 244–269.

[3] V. Shoup, "A Proposal for an ISO Standard for Public Key Encryption", *IACR Cryptology ePrint Archive, Paper 2001/112*, 2001, `https://eprint.iacr.org/2001/112`.

[4] M. Abdalla, M. Bellare, P. Rogaway, "The Oracle Diffie-Hellman assumptions and an analysis of DHIES", Topics in Cryptology—CT-RSA 2001: The Cryptographers' Track at RSA Conference, 2001, 143–158.

[5] N. Smart, "The exact security of ECIES in the generic group model", Cryptography and Coding: 8th IMA International Conference, 2001, 73–84.

[6] M. Bellare, C. Namprempre, "Authenticated encryption: Relations among notions and analysis of the generic composition paradigm", Advances in Cryptology—ASIACRYPT 2000: 6th International Conference on the Theory and Application of Cryptology and Information Security (Kyoto, Japan), 2000, 531–545.

[7] J. Katz, Y. Lindell, *Introduction to modern cryptography*, CRC press, Boca Raton, Florida, 2020, 626 pp.

[8] S. Chatterjee, A. Menezes, P. Sarkar, "Another look at tightness", Selected Areas in Cryptography: 18th International Workshop, SAC 2011, Toronto, ON, Canada, 2012, 293–319

[9] V. Nozdrunov, "Parallel and double block cipher mode of operation (PD-mode) for authenticated encryption", Proceedings of the 6th Workshop on Current Trends in Cryptology (CTCrypt 2017), 2017, 36–45

[10] *Standard R 1323565.1.026-2019. Information technology. Cryptographic data security. Authenticated encryption block cipher operation modes.*, 2019, In Russian.

[11] 3GPP, *Technical specification (TS). Security architecture and procedures for 5G System (3GPP TS 33.501 version 17.5.0 Release 17).*, 2022.

[12] E. Alekseev, I. Oshkin, V. Popov, S. Smyshlyaev, "On the cryptographic properties of algorithms accompanying the applications of standards GOST R 34.11-2012 and GOST R 34.10-2012", *Matematicheskie Voprosy Kriptografii [Mathematical Aspects of Cryptography]*, **7**:1 (2016), 5–38, In russian.

[13] F. Guo, W. Susilo, Y. Mu, *Introduction to security reduction*, Springer, Cham, Switzerland, 2018, 253 pp.

[14] L. Ahmetzyanova, E. Alekseev, I. Oshkin, S. Smyshlyaev, L. Sonina, "On the properties of the CTR encryption mode of Magma and Kuznyechik block ciphers with re-keying method based on CryptoPro Key Meshing", *Matematicheskie Voprosy Kriptografii [Mathematical Aspects of Cryptography]*, **8** (2017), 39–50

[15] M. Bellare, O. Goldreich, A. Mityagin, "The Power of Verification Queries in Message Authentication and Authenticated Encryption", *IACR Cryptology ePrint Archive, Paper 2004/309*, 2004, `https://eprint.iacr.org/2004/309`.

[16] P. Rogaway, "Evaluation of some blockcipher modes of operation", *Cryptography Research and Evaluation Committees (CRYPTREC) for the Government of Japan*, 2011.

[17] T. Iwata, K. Kurosawa, "Stronger security bounds for OMAC, TMAC, and XCBC", International Conference on Cryptology in India, 2003, 402–415.

[18] M. Nandi, "Improved security analysis for OMAC as a pseudorandom function", *Journal of Mathematical Cryptology*, **3**:2 (2009), 133–148.

[19] S. Chattopadhyay, A. Jha, M. Nandi, "Towards tight security bounds for OMAC, XCBC and TMAC", Advances in Cryptology–ASIACRYPT 2022: 28th International Conference on the Theory and Application of Cryptology and Information Security, 2023, 348–378.

[20] V. I. Nechaev, "Complexity of a determinate algorithm for the discrete logarithm", *Math. Notes*, **55**:2 (1994), 165–172.

[21] V. Shoup, "Lower bounds for discrete logarithms and related problems", Advances in Cryptology—EUROCRYPT'97: International Conference on the Theory and Application of Cryptographic Techniques Konstanz, Germany, 1997, 256–266.

# A Additional security models

In this Section we briefly discuss additional models needed for the evaluation of one concrete instantiation of ECIES scheme given in Section B.

## A.1 PRF-security of keyed functions

Let us consider a keyed function, i.e., a family of functions $F = \{F_k\}_{k\in\mathtt{Keys}}$, $F_k\colon \mathtt{Dom} \to \mathtt{Rng}$, indexed by some key $k$ from the set of keys $\mathtt{Keys}$. We give a formal definition of PRF model.

**Definition 13.** *The advantage of the adversary $\mathcal{A}$ in the* PRF*-model for the function family $F$ is defined as:*

$$\mathbf{Adv}_F^{\mathrm{PRF}}(\mathcal{A}) = \mathbb{P}\big[\mathbf{Exp}_F^{\mathrm{Left}}(\mathcal{A}) \to 1\big] - \mathbb{P}\Big[\mathbf{Exp}_F^{\mathrm{Right}}(\mathcal{A}) \to 1\Big],$$

*where $\mathbf{Exp}_F^{\mathrm{Left}}$ and $\mathbf{Exp}_F^{\mathrm{Right}}$ are defined as follows:*

| $\mathbf{Exp}_F^{\text{Left}}(\mathcal{A})$ | $\mathbf{Exp}_F^{\text{Right}}(\mathcal{A})$ |
|---|---|
| $k \xleftarrow{\$} \texttt{KeyGen}$ | $Asked \leftarrow [\,]$ |
| $b' \xleftarrow{\$} \mathcal{A}^{\mathcal{O}}$ | $b' \xleftarrow{\$} \mathcal{A}^{\mathcal{O}}$ |
| **return** $b'$ | **return** $b'$ |
| $\mathcal{O}(x)$ | $\mathcal{O}(x)$ |
| **return** $F_k(x)$ | **if** $Asked[x] = \perp$ |
| | $Asked[x] \xleftarrow{\$} \texttt{Rng}$ |
| | **fi** |
| | **return** $Asked[x]$ |

**Definition 14.** *Let* $\mathbf{InSec}_F^{\text{PRF}}(t, q, l, \mu)$ *be the maximal advantage* $\mathbf{Adv}_F^{\text{PRF}}(\mathcal{A})$*, where the maximum is taken over the adversaries* $\mathcal{A}$ *whose time complexity is at most* $t$ *and with the following restrictions on the oracle queries:*

- *the number of queries to the* $\mathcal{O}$ *oracle does not exceed* $q$*;*
- *the total length of the queries* $\sum |m|$ *to the* $\mathcal{O}$ *oracle does not exceed* $l$*;*
- *the maximal length of the query* $\max |m|$ *to the* $\mathcal{O}$ *oracle does not exceed* $\mu$*.*

**Remark 6.** *If the function family is a block cipher, i.e., each* $E_k \colon \{0,1\}^n \to \{0,1\}^n$*, then* $l = q$*,* $\mu = 1$*. It is usually assumed that for the modern secure block cipher is holds that:*

$$\mathbf{InSec}_E^{\text{PRF}}(t, q, q, 1) \approx \frac{q^2}{2^n},$$

*i.e., the best possible attack in the* PRF *model is the birthday-based attack.*

## A.2 SUF-CMA model for deterministic MAC function

**Brief description.** The standard SUF-CMA model of forging MAC tag for the message in **deterministic MAC function** setting (see [7, 15]) is usually used to investigate the integrity property. In the model the adversary $\mathcal{A}$ is given access to the MAC calculation oracle $\mathcal{O}_{\text{mac}}$ and MAC verification oracle $\mathcal{O}_{\text{verify}}$. It is able to adaptively choose messages $m$ and obtain MAC-tags $\tau$ for them (using $\mathcal{O}_{\text{mac}}$ queries) under a fixed (unknown to the adversary) key $k$.

The ultimate goal is to forge a tag $\tau$ for a message $m$, such that the pair $(m, \tau)$ was not outputted by $\mathcal{O}_{\text{mac}}$-oracle before.

**Remark 7.** *In case of deterministic MAC function* SUF-CMA *model is equivalent to* EUF-CMA *model. In the latter model $\mathcal{A}$'s goal is to forge a tag $\tau$ for a message $m$ that was not queried before, i.e. to obtain a pair $(m, \tau)$ such that $\tau$ is a valid tag for $m$ (under a key $k$), and $m$ does not belong to the set of input queries of $\mathcal{O}$.*

**Security model.** Let $MAC(k, m)$ be a function that computes MAC tag under a key $k$ for a message $m$.

**Definition 15.** *The advantage of adversary $\mathcal{A}$ in the* EUF-CMA *model for the deterministic MAC function $MAC$ is defined as:*

$$\mathbf{Adv}^{\text{EUF-CMA}}_{MAC}(\mathcal{A}) = \mathbb{P}\big[\mathbf{Exp}^{\text{EUF-CMA}}_{MAC}(\mathcal{A}) \to 1\big],$$

*where $\mathbf{Exp}^{\text{EUF-CMA}}_{MAC}$ is defined as follows:*

| $\mathbf{Exp}^{\text{EUF-CMA}}_{MAC}(\mathcal{A})$ | $\mathcal{O}_{\text{mac}}(m)$ | $\mathcal{O}_{\text{verify}}(m, \tau)$ |
|---|---|---|
| $k \xleftarrow{\$} \texttt{KeyGen}$ | $Sent \leftarrow Sent \cup \{m\}$ | $res \leftarrow (\tau = MAC(k, m))$ |
| $Sent = \emptyset$ | **return** $MAC(k, m)$ | **if** $(m \notin Sent)$ & $(res = \texttt{true})$ |
| $win \leftarrow \texttt{false}$ | | $win \leftarrow \texttt{true}$ |
| $\mathcal{A}^{\mathcal{O}_{\text{mac}}, \mathcal{O}_{\text{verify}}}$ | | **fi** |
| **return** $win$ | | **return** $res$ |

**Definition 16.** *Let $\mathbf{InSec}^{\text{EUF-CMA}}(t, q_e, q_v, l_e, l_v, \mu_e, \mu_v)$ be the maximal advantage $\mathbf{Adv}^{\text{EUF-CMA}}_{MAC}(\mathcal{A})$, where the maximum is taken over the adversaries $\mathcal{A}$ whose time complexity is at most $t$ and with the following restrictions on the oracle queries:*

- *the number of queries to the $\mathcal{O}_{\text{mac}}$ oracle (to the $\mathcal{O}_{\text{verify}}$ oracle) does not exceed $q_e$ ($q_v$ resp.);*
- *the total length of the queries $\sum |m|$ to the $\mathcal{O}_{\text{mac}}$ oracle (to the $\mathcal{O}_{\text{verify}}$ oracle) does not exceed $l_e$ ($l_v$ resp.);*
- *the maximal length of the query $\max |m|$ among queries to the $\mathcal{O}_{\text{mac}}$ oracle (to the $\mathcal{O}_{\text{verify}}$ oracle) does not exceed $\mu_e$ ($\mu_v$ resp.).*

**Concrete estimates.** For simplicity, we will omit some technical details that does not influence the final estimate under reasonable assumptions (e.g., we use the notation $t' = O(t)$ and drop the $\mathbf{InSec}^{\text{PRP}}_{E}(\cdot)$ term, which is negligible compared to other terms for the "good" block cipher).

For the estimation of EUF-CMA security of CMAC scheme one can use the results from [15]: from the proof of Theorem 5.1 in [15] it follows that for the deterministic MAC function (such as CMAC) it holds that:

$$\mathbf{InSec}^{\text{EUF-CMA}}_{CMAC}(t, q_e, q_v, l_e, l_v, \mu_e, \mu_v) \leq \\ \leq q_v \cdot \mathbf{InSec}^{\text{EUF-CMA}}_{CMAC}(t', q_e, 1, l_e, l_v, \mu_e, \mu_v), \quad (2)$$

i.e., only one verification query $q_v = 1$ is sufficient.

The results from [16] gives the following (implicit) estimate for CMAC function (with one verification query):

$$\mathbf{InSec}^{\text{EUF-CMA}}_{CMAC}(t, q_e, 1, l_e, l_v, \mu_e, \mu_v) \leq \\ \leq \mathbf{InSec}^{\text{PRF}}_{CMAC}(t', q_e + 1, l_e + l_v, \max(\mu_e, \mu_v)) + \frac{1}{2^{tlen}}, \quad (3)$$

where $tlen$ is the length of the MAC tag.

Finally, there are three (incomparable) results on the insecurity of $CMAC$ scheme (based on a block cipher with the block length $n$) in the PRF-model [17, 18, 19]:

$$\mathbf{InSec}^{\text{PRF}}_{CMAC}(t, q, l, \mu) \leq \frac{4l^2}{2^n}, \quad (4)$$

$$InSec^{\text{PRF}}_{CMAC}(t, q, l, \mu) \leq \frac{4ql}{2^n} + \frac{8q(q-1)\mu^4}{2^{2n}}, \quad (5)$$

$$InSec^{\text{PRF}}_{CMAC}(t, q, l, \mu) \leq \frac{4l + 16q^2 + q\mu^2}{2^n} + \\ + \frac{8q^2\mu^4 + 32q^3\mu^2 + 2q^2\mu^3}{2^{2n}} + \frac{3q^3\mu^5 + 143q^3\mu^6 + 11q^4\mu^3}{2^{3n}} + \\ + \frac{17q^4\mu^6 + 5462q^4\mu^8}{2^{4n}}. \quad (6)$$

For concrete parameters the minimum among the three can be used.

Hence, for the insecurity of CMAC in the EUF-CMA model the following inequality holds:

$$\mathbf{InSec}^{\text{EUF-CMA}}_{CMAC}(t, q_e, q_v, l_e, l_v, \mu_e, \mu_v) \leq \\ \leq \frac{q_v}{2^n} + q_v \cdot \min\Big(\frac{4l^2}{2^n}, \quad \frac{4ql}{2^n} + \frac{8q(q-1)\mu^4}{2^{2n}}, \\ \frac{4l + 16q^2 + q\mu^2}{2^n} + \frac{8q^2\mu^4 + 32q^3\mu^2 + 2q^2\mu^3}{2^{2n}} + \\ + \frac{3q^3\mu^5 + 143q^3\mu^6 + 11q^4\mu^3}{2^{3n}} + \frac{17q^4\mu^6 + 5462q^4\mu^8}{2^{4n}}\Big), \quad (7)$$

where $q = q_e + 1$, $l = l_e + l_v$, $\mu = \max(\mu_e, \mu_v)$, $tlen = n$.

# B Example of concrete instantiation for ECIES scheme

Let us now describe one concrete example of ECIES scheme instantiation, i.e., we specify concrete $\mathcal{KE}$ and $\mathcal{AE}$ schemes.

## B.1 $\mathcal{KE}$ scheme

The set of ephemeral keys is a cyclic subgroup $H = \langle P \rangle$ of the group of points of elliptic curve minus zero point $EpkSet = H - \{0\}$. Let $h$ be the size of the subgroup: $h \leftarrow |H| - 1$.

The algorithm $\mathcal{KE}$.`KeyPairGen` works as follows:

– generate ephemeral secret key $sk \leftarrow \{1, \ldots, h\}$;

– generate ephemeral public key $pk \leftarrow sk \cdot P$.

$\mathcal{KE}$.`Combine` algorithm (as well as the concrete elliptic curve and the group) can be specified as it is done in [12, Section 3.7] (VKO function with $UKM \leftarrow 1$, $t \leftarrow 512$).

## B.2 AE-scheme $\mathcal{AE}$

Key generation algorithm $\mathcal{AE}$.`KeyGen` generates two independent uniformly random keys $k_{enc} \xleftarrow{\$} \{0,1\}^{256}$, $k_{mac} \xleftarrow{\$} \{0,1\}^{256}$. To process the message $m$ under the key $k \leftarrow k_{enc} \| k_{mac}$, two steps are to be done:

– message $m$ is encrypted using CTR mode of operation $c \leftarrow CTR^{IV}[E](k_{enc}, m)$, where $IV \leftarrow 0^n$, $E$ is a block cipher to be used in CTR mode, $n$ is the block length;

– the tag $\tau$ is computed using CMAC function $\tau \leftarrow CMAC(k_{mac}, c)$;

The result of the encryption is the pair $ct \leftarrow (c, \tau)$. To decrypt the ciphertext $ct$ under the key $k \leftarrow k_{enc} \| k_{mac}$ one has to check the tag $\tau$ first: if it is invalid, then $\perp$ is returned; otherwise, decrypt $c$ using CTR mode of operation. This process follows the widespread Encrypt-then-MAC approach to the construction of AE-schemes [6, Section 4.3].

## B.3 Scheme evaluation

In this section we give a brief overview of the estimates for our concrete instantiation of ECIES scheme. Again, in order to simplify the estimates, we omit some technical details that does not influence the final result under

reasonable assumptions (e.g., we use the notation $t' = O(t)$, $t'' = O(t)$ and drop the $\mathbf{InSec}_E^{\mathrm{PRP}}(\cdot)$ term, which is negligible compared to other terms for the "good" block cipher).

As it is shown in [6], for the "Encrypt-then-MAC" schemes it holds that:

$$\mathbf{InSec}_{\mathcal{AE}}^{\text{INT-CTXT}}(t, q_e, q_v, l_e, l_v, \mu_e, \mu_v; 1) \leq \\ \leq \mathbf{InSec}_{CMAC}^{\text{SUF-CMA}}(t', q_e, q_v, l_e, l_v, \mu_e, \mu_v). \quad (8)$$

$$\mathbf{InSec}_{\mathcal{AE}}^{\text{LOR-CCA}}(t, q_e, q_d, l_e, l_d, \mu_e, \mu_d; 1) \leq \\ \leq \mathbf{InSec}_{\mathcal{AE}}^{\text{LOR-CPA}}(t', q_e, l_e, \mu_e) + \\ + 2 \cdot \mathbf{InSec}_{CMAC}^{\text{SUF-CMA}}(t'', q_e, q_d, l_e, l_d, \mu_e, \mu_d), \quad (9)$$

For the CTR mode the following estimate holds [14]:

$$\mathbf{InSec}_{\mathcal{AE}}^{\text{LOR-CPA}}(t, q_e, l_e, \mu_e) \leq 2 \cdot \mathbf{InSec}_E^{\text{PRF}}(t', l_e). \quad (10)$$

The insecurity of $CMAC$ in SUF-CMA model was analyzed in Section A.2. The insecurity of a block cipher $E$ in PRF model is briefly described in Section A.1.

The problem of analyzing (in)security of $VKO$ function in the MODH model is more elaborate. In [12, Remark 5.2] it is claimed that due to the fact that algebraic group and the hash fuction used in VKO are "unrelated", the complexity of CDH (Computational Diffie-Hellman) problem might be roughly equivalent to the complexity of certain problems tailored for VKO function. In [4] the problem is analyzed using various different idealizations (such as modelling hash function as a random oracle, or to restrict attention only to "generic" algorithms over group, see [20, 21] for more details). The bound of the form $O\left(\frac{q^4}{h}\right)$ for the ODH-RO-Generic-model (MODH-model with one query to the $\mathcal{O}^b_{\text{kgen}}$-oracle, hash function is treated as a random oracle, the adversary has only an oracle access to the operations in elliptic curve group) is obtained in [4]. The problem of analyzing (in)security of VKO function in the ODH model (with various idealizations) is out of scope of this paper and should be studied separately.

# On security aspects of CRISP

Vitaly Kiryukhin

LLC «SFB Lab», JSC «InfoTeCS», Moscow, Russia
vitaly.kiryukhin@sfblaboratory.ru

**Abstract**

Using the provable security approach, we analyze CRISP – a standardized Russian cryptographic protocol that aims to ensure confidentiality, integrity of transmitted messages, as well as protection against replay attacks. The protocol is considered as a specific mode of authenticated encryption with associated data (AEAD). We take into account that one key can be used by many protocol's participants and in different cipher suites. We impose requirements for the set of the cipher suites used in the protocol and show that the existing ones meet them. Estimates of the maximum allowable amount of data processed using a single key are also given.

**Keywords:** CRISP, provable security, AEAD

## 1 Introduction

CRISP (CRyptographic Industrial Security Protocol) [4] is a secure data transfer protocol designed for use in industrial systems. The security properties that should be provided by the protocol are confidentiality and integrity (or only integrity) of messages and protection against replay attacks.

Important features of the protocol include the following.

*Non-Interactivity.* Protocol participants do not establish a session, preshared keys are used. Each message contains all (or almost all) the necessary information for processing. Messages may be received out of order.

*Multicasting and shared keys.* One message from one sender can be intended for many receivers. All users of the information system can share the same secret key.

*Dynamic selection of a cipher suite.* For each message, the sender can choose any cipher suite from the available ones. Some of them provide confidentiality and integrity, while others provide only integrity.

In this paper, we analyze the cryptographic properties of the protocol by using the provable security approach [8, 9]. We take into account the declared security properties and the above-mentioned protocol features.

Non-Interactivity and simplicity of CRISP encourage us to consider the protocol as a specific *encryption mode*. The lack of authenticated key exchange (AKE) makes it completely irrelevant to use the Canetti-Krawczyk security models [12, 13]. Formal verification tools, such as AVISPA [11], are also useless here for the same reason. We also emphasize that the presented security proofs (based on the approach of Rogaway and Bellare) gives us not only qualitative, but also (more importantly) *quantitative* characteristics, including the "imperfection" of the used encryption algorithms. On the contrary, verification tools usually assume the unconditional ideality of all primitives, and also give only a qualitative result.

The results are presented as follows. In the section 2, the necessary notations and brief information about the provable security paradigm are presented. The third section describes the protocol.

The fourth section is devoted to the general analysis of the protocol's security. We begin with an informal discussion about the capabilities and goals of the adversary. Next, we introduce requirements for the set of the used cipher suites. We show that protocol can be considered as an authenticated encryption with associated data (AEAD) algorithm and then prove that with suitable cipher suites, the CRISP protocol is secure in the relevant threat model.

Section 5 contains the results of the analysis of the existing cipher suites used in CRISP. The known bounds for the cipher modes when used separately or jointly (as AEAD modes) are presented.

In conclusion, estimates of the key capacity (i.e. permissible amount of data processed with one key) and ways to increase them are given.

## 2 Notations and definitions

We use the following notations throughout the paper:

$n$ – block size in bits; $k$ – key size in bits; $\tau \leq n$ – tag size in bits; $\oplus$ – bitwise XOR operation; $||$ – concatenation of binary strings;

$V^*$ – the set of all binary strings of a finite length;

$V^n$ – the set of all $n$-bit strings;

$V^{\leq L}$ – the set of binary strings of length no more than $L$ bits;

$(V^n)^{\leq l}$ – the set of binary strings of length no more than $l \cdot n$ bits, the length of each string is a multiple of $n$;

$|X|$ – bit length of binary string $X$;

$\mathrm{Func}(\mathbf{X}, \mathbf{Y})$ – the set of all mappings from the set $\mathbf{X}$ to the set $\mathbf{Y}$;

$\mathrm{Perm}(\mathbf{X})$ – the set of all permutations on the set $\mathbf{X}$;

$X \xleftarrow{\mathrm{R}} \mathbf{X}$ – uniform and random selection of element $X$ from the set $\mathbf{X}$.

Transformations (including ciphers, cipher modes and protocols) are denoted in Sans Serif: $\mathsf{E}$, $\mathsf{CTR}$, $\mathsf{CRISP}$. If the transformation $\mathsf{A}$ uses the transformation $\mathsf{B}$ from the set of all possible parametrizations, we denote it by $\mathsf{A}[\mathsf{B}]$. The parameter $[\mathsf{B}]$ is omitted when it is clear based on the context.

The adversary is modeled by an interactive probabilistic algorithm that has access to other algorithms (oracles). We denote by $\mathrm{Adv}^{TM}_{\mathsf{Alg}}(\mathcal{A})$ a quantitative characterization (advantage) of the capabilities of the adversary $\mathcal{A}$ in realizing a certain threat, defined by the model $TM$, for the cryptographic scheme $\mathsf{Alg}$. The resources of $\mathcal{A}$ are measured in terms of time and query complexities. The time complexity $t$ includes the description size of $\mathcal{A}$ in some computation model. The query complexity $q$ is measured in the number of adaptively chosen input/output pairs. We assume that $\mathcal{A}$ always uses exactly $q$ unique queries (with no redundant or repeating queries). The algorithm of an oracle (or several oracles) is fixed in the definition of the threat model $TM$. The result of computations of $\mathcal{A}$ after interacting with oracles $\mathcal{O}_1$, $\mathcal{O}_2$, ... $\mathcal{O}_w$, $w \in \mathbb{N}$ is some binary value $x$, which is denoted as $\mathcal{A}^{\mathcal{O}_1,\mathcal{O}_2,\ldots,\mathcal{O}_w} \Rightarrow x$.

The maximum of the advantage among all resource constrained adversaries is denoted by

$$\mathrm{Adv}^{TM}_{\mathsf{Alg}}(t,q) = \max_{\mathcal{A}(t',q'):t'\le t,\ q'\le q,} \mathrm{Adv}^{TM}_{\mathsf{Alg}}(\mathcal{A}).$$

Some threat models, which would be addressed later, imply different types of resources, like the number of queries to different oracles, the length of these queries, etc. The advantage for such models is defined in similar way.

The cryptoalgorithm $\mathsf{Alg}$ is informally called secure in the threat model $TM$ ($TM$-secure) if $\mathrm{Adv}^{TM}_{\mathsf{Alg}}(t,q) < \varepsilon$, where $\varepsilon$ is some small value determined by the requirements for the strength of the cryptosystem and the resources $t$ and $q$ are comparable to those available to the adversary in practice.

To demonstrate the practical significance of the obtained results, we sometimes substitute heuristic estimates based on assumptions into derived security bounds. The resulting informal estimates are denoted by symbol "$\lessapprox$" meaning "less or equal if the assumptions are true", a slight loss due to omitting of insignificant addends may also occur.

Definitions of frequently used formal models are presented in Appendix A. Other essential definitions are given in the text.

# 3 Protocol description

## 3.1 Packet fields

The message (packet) in CRISP consists of the header, payload (`PayloadData`) and tag (`ICV`). The header consists of five fields: `ExternalKeyIdFlag`, `Version`, `CS`, `KeyId`, `SeqNum`. The sizes of the fields are shown in the table below, the total length of all seven fields does not exceed 2048 bytes.

| | Name | Symbol | Length in bits | |
|---|---|---|---|---|
| 1 | `ExternalKeyIdFlag` | – | 1 | Header $H$ |
| 2 | `Version` | – | 15 | |
| 3 | `CS` | $CS$ | 8 | |
| 4 | `KeyId` | – | from 8 to 1024 | |
| 5 | `SeqNum` | $SN$ | 48 | |
| 6 | `PayloadData` | $P$ and $C$ | variable | Payload |
| 7 | `ICV` | $T$ | variable | Tag |

Table 1: List of CRISP-packet fields

The `ExternalKeyIdFlag` and `KeyId` fields indicate the master key $K$ used to process the message. The length of the `KeyId` field is uniquely determined by the first byte of the field itself.

If the flag is zero (`ExternalKeyIdFlag` = 0), then the key is uniquely determined by the field `KeyId`. Otherwise, external information is used.

The field `Version` is fixed and reserved for possible future modifications.

The field `SeqNum` contains the sequence number $SN$ of the message.

The field `CS` contains the identifier $CS$ of the cipher suite. The latter includes:

– `EncryptionAlg` – the encryption/decryption algorithms Enc/Dec (can be set to NULL, meaning that no encryption is applied);

– `MACAlg` – the message authentication code Mac, which computes the $\tau$–bit (`MACLength`) tag $T$;

– `DeriveIV` – the algorithm DerIv for generating nonces;

– `DeriveKey` – the algorithm KDF for producing derived keys from the master key, and the subalgorithm DerIvKDF that takes $SN$ as input and produces a bit string that is suitable for use as a KDF parameter, thus making its output dependent on $SN$.

We also refer to the composition of Enc and Mac as AE (authenticated encryption).

The field `PayloadData` contains plaintext $P$ or ciphertext $C$, depending on the chosen cipher suite. The tag $T$ is computed for all data in fields 1–6

and is contained in the field `ICV`. The tag length is also defined by the cipher suite.

### 3.2 Common restrictions

The protocol assumes that the sender and the receiver(s) have the same pre-shared master key $K$ with the identifier $K_{ID}$. Each sender has its own unique identifier `SourceIdentifier` (we denote them by $S_{ID}$). In the system there is an injective correspondence of the form $K_{ID} \to (K, S_{ID})$, different $K_{ID}$ can correspond to the same $K$ (multiple senders use the same master key). The receiver determines $K_{ID}$ from the fields `ExternalKeyIdFlag`, `KeyId`, and possibly by some external data.

### 3.3 Initialization of the sequence number

Before using the specific master key $K$, the sender sets the initial value of $SN \in [0, 2^{48} - 1]$ in an unspecified way. The sequence number must be increasing (for each message from one sender using one key), which includes overflow protection. For each $(K, S_{ID})$ the receiver initializes the lower $\underline{SN}$ and the upper $\overline{SN}$ bounds of the window (binary vector) $W$ of received messages, $\underline{SN} = \overline{SN} = 0$. The $j$-th bit of $W$ is set to one if $j$-th message was received. The receiver stores only bits of $W$ from $\underline{SN}$-th to $\overline{SN}$-th inclusive. The window size is the predefined constant $1 \leq Size \leq 256$, $(\overline{SN} - \underline{SN}) \leq Size$.

### 3.4 Sender's algorithm

The sender with some $S_{ID}$ selects the master key $K$ (and corresponding $K_{ID}$), the plaintext $P$, and the $CS$-th cipher suite.

1) The sequence number $SN$ is determined by $K_{ID}$, the value of $SN$ increases by 1.

2) Derived keys $K_{MAC}$ and (if presented) $K_{ENC}$ are computed

$$(K_{ENC}, K_{MAC}) = \mathsf{KDF}(K, prms),$$

where the specific content of $prms$ is determined by the cipher suite and may include $CS$, $S_{ID}$, $\mathsf{DerIvKDF}(SN)$, and other parameters.

3) The header $H$ (fields 1-5) is generated, including $SN$ and $CS$.

4) If the cipher suite provides encryption, then the ciphertext is computed as $C = \mathsf{Enc}(K_{ENC}, IV, P)$, $IV = \mathsf{DerIv}(SN)$, otherwise, $C = P$ is set.

5) The tag $T = \mathsf{Mac}(K_{MAC}, H||C)$ is computed.

6) The message of the form $H||C||T$ is sent.

### 3.5 Receiver's algorithm

The receiver parses the received message as $H'||C'||T'$ (possibly modified or forged by an attacker) and processes it according to the following algorithm.

1) If the protocol version or the cipher suite specified in $H$ is not supported, then stop processing.

2) $K_{ID}$ is determined by the `KeyId`, `ExternalKeyIdFlag` fields and possibly by external data. Next, $K$, $S_{ID}$, $(\underline{SN}, \overline{SN})$, and $W$ are determined by the value of $K_{ID}$. If the key $K$ is not found, then stop processing.

3) The validity of the sequence number $SN$ is checked:

– if $SN < \underline{SN}$, then stop processing;

– if $SN$-th bit of $W$ is equal to one, then stop processing.

4) Derived keys $K_{MAC}$ and (if necessary) $K_{ENC}$ are computed $(K_{ENC}, K_{MAC}) = \mathsf{KDF}(K, prms)$.

5) The tag $T'' = \mathsf{Mac}(K_{MAC}, H'||C')$ is computed. If the received and computed tags are not equal ($T' \neq T''$), then stop processing.

6) The window of received messages is updated:

– if $\overline{SN} < SN$, then set $\overline{SN} = SN$ and $\underline{SN} = \min(SN - Size + 1, 0)$;

– the $SN$-th bit of $W$ is set to one.

7) If the cipher suite provides encryption, then the result is computed as $P' = \mathsf{Dec}(K_{ENC}, IV, C')$, $IV = \mathsf{DerIv}(SN)$, otherwise, $P' = C'$.

## 4 General security analysis

Mathematically rigorous proof of the security properties of any cryptoalgorithm is possible only in the formal model that includes the qualitative and quantitative capabilities of the adversary, as well as his goals. The discrepancy between the model and practice is a potential source of threats and attacks (see the well-known example of the inconsistency between the model [14] and the attack [15] on the SSL protocol).

The above considerations motivate: to carefully include in the model the capabilities available in practice; to establish the weakest possible goal(s); to stipulate the limitations of the formal model.

Obviously, the adversary knows everything except the keys. The attacker can adaptively chosen plaintexts $P$ and headers $H$, including the cipher suite $CS$, master key identifier $K_{ID}$, sender identifier $S_{ID}$, and the sequence number $SN$, but pairs $(S_{ID}, SN)$ are not repeated (i.e. each sender does not use the same sequence number twice with the same key). The adversary also can

drop, reorder, or modify any number of packets.

The adversary's goals are the follows.

1) To get any information about the plaintext $P$ from the ciphertext $C$ (except the length).

2) To make a forgery (create a new valid packet that has not been formed by any sender before).

3) To make a replay (some receiver recognizes the same packet as valid at least twice).

Next, we list the capabilities of the adversary, which he may potentially possess in reality, but which are *not* included in the formal models: changing the protocol version (it is assumed that there is only one); compromising protocol participants (i.e. leakage of the participants' keys); side-channel attacks; fault attacks. The security properties of the protocol when disclosing some keys are shortly discussed at the end of the section.

The non-interactivity of the protocol, along with the aforementioned features and the first two goals of the adversary, prompts us to consider CRISP in the well-established $NAE$ model (*Nonce-based Authenticated Encryption*), see, for example, [26]. This model is also similar to $IND\text{-}CCA3$ proposed in [22]. We prove that even stateless version of the protocol ensures confidentiality and integrity (with the caveat that the sender does not repeat the same $SN$). Storing states that include windows of the received messages and the sequence numbers provides simple protection against replays.

### 4.1 Requirements for the cipher suites

We define the cipher suite of the CRISP as the tuple of four algorithms

$$\mathsf{CS} = (\mathsf{KDF}, \mathsf{DerIvKDF}, \mathsf{AE}, \mathsf{DerIv}),$$

where AE can be either a composition of Enc and Mac, or only one algorithm Mac, or a dedicated authenticated encryption mode.

Here we briefly outline the requirements sufficient for the security proof.

Let the master key $K$ be used in several cipher suites. All of them must use the same KDF[1]. Different cipher suites can use keys of different lengths, therefore, KDF must be $PRF$-secure with variable length of the output (VO-PRF). The input of the KDF must include at least the sender ID $S_{ID}$ and the number $CS$ of the cipher suite. Due to this, different users and different cipher suites will have computationally independent keys. Some bits (we denote

[1]Concurrent usage of different KDFs (for example, CMAC-Magma and HMAC-Streebog) in different cipher suites may not immediately lead to efficient attacks, but when trying to prove formally, some poorly understood basic problems arise during the reduction.

them as $\mathsf{DerIvKDF}(SN)$) of the sequence number $SN$ can also be used as the part of the input. We demand the absence of collisions among nonces, for any $SN \neq SN'$, $\mathsf{DerIvKDF}(SN) \neq \mathsf{DerIvKDF}(SN')$ or/and $\mathsf{DerIv}(SN) \neq \mathsf{DerIv}(SN')$.

If the cipher suite is designed to ensure confidentiality and integrity, then $\mathsf{AE}$ must be a secure deterministic AEAD scheme (dedicated or combined). All this properties are also formalized in the $NAE$ model. For cipher suites, from which only integrity is expected, we make the same requirements, but in this case the length of the encrypted data is zero. For example, if $\mathsf{AE} = \mathsf{Mac}$, then $PRF$-security of $\mathsf{Mac}$ is sufficient. Nonce-based schemes, such as Carter-Wegman [6] construction in GCM [10] and UMAC [7] are also suitable.

### 4.2 Protocol in the $NAE$ model

Stateless version of $\mathsf{CRISP}$ is considered in scenario "many senders and one receiver have a single pre-shared key" within the following definitions.

**Definition**. *The deterministic nonce-based authenticated encryption is the pair of the algorithms*

$$\begin{aligned}\mathsf{AE} :& \mathbf{K} \times \mathbf{N} \times \mathbf{A} \times \mathbf{P} \quad \to \mathbf{C} \times \mathbf{T},\\ \mathsf{AE}^{-1} :& \mathbf{K} \times \mathbf{N} \times \mathbf{A} \times \mathbf{C} \times \mathbf{T} \to \mathbf{P} \cup \{\perp\},\end{aligned}$$

*where* $\mathbf{K}$, $\mathbf{N}$, $\mathbf{A}$, $\mathbf{P}$, $\mathbf{C}$, $\mathbf{T}$ *are sets of keys, nonces, associated data, plaintexts, ciphertexts, tags, respectively. For any* $(C, T) = \mathsf{AE}(K, N, A, P)$, $P = \mathsf{AE}^{-1}(K, N, A, C, T)$ *is true.*

**Definition**. *The advantage of* $\mathcal{A}$ *in the model* $NAE$ *for* $\mathsf{AE}$ *is*

$$\mathrm{Adv}^{NAE}_{\mathsf{AE}}(\mathcal{A}) = \Pr\left(K \xleftarrow{\mathrm{R}} \mathbf{K} : \mathcal{A}^{\mathsf{AE}_K(\cdot,\cdot,\cdot),\mathsf{AE}_K^{-1}(\cdot,\cdot,\cdot,\cdot)} \Rightarrow 1\right) - \Pr\left(\mathcal{A}^{\$(\cdot,\cdot,\cdot),\perp(\cdot,\cdot,\cdot,\cdot)} \Rightarrow 1\right).$$

*The oracle* \$ *receives the query* $(N, A, P)$ *and returns a random binary string of length* $|P|+ext(P)$ *bits. The extension function* $ext(P)$ *calculates the total length of the tag and padding. The oracle* $\perp$ *always returns error symbol "*$\perp$*". The queries from* $\mathcal{A}$ *to the left oracle (*$\mathsf{AE}$ *or* \$*) does not contain the same* $N$*.* $\mathcal{A}$ *does not resend to the right oracle (*$\mathsf{AE}^{-1}$ *or* $\perp$*) the answers of the left, that is, it does not query* $(N, A, C, T)$*, where* $(C, T)$ *is the answer of the left oracle to the query* $(N, A, P)$*.* $\mathcal{A}$ *makes* $q$ *(resp.* $\nu$*) queries to the left (resp. right) oracle of no more than* $l$ $n$*-bit blocks each.*

Everywhere else, $N \in \mathbf{N}$ is uniquely determined by the associated data $A \in \mathbf{A}$, hence, the set $\mathbf{N}$ is *implicit*. The algorithm $\mathsf{AE}$ can be defined on some *subset* of $\mathbf{A} \times \mathbf{P}$ (with similar changes in $\mathsf{AE}^{-1}$), not on the whole $\mathbf{A} \times \mathbf{P}$.

For CRISP we have:

– the set of all master keys $\mathbf{K} = V^k$;

– $\mathbf{T} = V^{\leq \tau_{\max}}$ (all possible values of the field `ICV`);

– $\mathbf{P} = \mathbf{C} = V^{\leq L_P}$ (`PayloadData`);

– $\mathbf{A} \subseteq \mathbf{A}_{\text{ext}} \times \mathbf{H} \times \mathbf{P}$ (external data plus all possible header values plus `PayloadData` field).

Here we consider external data $A_{\text{ext}} \in \mathbf{A}_{\text{ext}}$ as an "imaginary" packet field. The set $\mathbf{H} \subset V^{\leq L_H}$ contains all possible header values. The values of $L_H$ and $L_P$ do not exceed the packet length (excluding the tag length), $\tau_{\max}$ is the maximum length of the tag among all cipher suites.

The associated data $A \in \mathbf{A}$ explicitly contains the entire header $H \in \mathbf{H}$, and hence the sequence number $SN$, and the cipher suite number $CS$. `KeyId` and `ExternalKeyIdFlag` from $H$, and possibly empty external data $A_{\text{ext}} \in \mathbf{A}_{\text{ext}}$ implicitly correspond to the pair $(K, S_{ID})$. We assume that this mapping is *injective*, hence, changing the external data leads to change of the key or/and $S_{ID}$ [2]. The length of the `KeyId` field can be different, but the length used is uniquely determined by the first byte of the field. Therefore, changing the length does not violate the injectivity of encoding. The pair $(S_{ID}, SN) \in \mathbf{N}$ is considered as a nonce.

If the chosen cipher suite provides only integrity, then the input of CRISP is $((A_{\text{ext}}, H, P), \emptyset)$, the associated data $A$ consists of the external data $A_{\text{ext}}$, the header $H$, and the payload $P$. Otherwise, if both confidentiality and integrity are provided, then the input is $((A_{\text{ext}}, H, \emptyset), P)$. This constraints define the subset of $\mathbf{A} \times \mathbf{P}$ on which CRISP operates.

**Theorem 1.** *The advantage of the adversary in the NAE model attacking the* CRISP *that uses the cipher suites from the set* $\mathbf{CS} = \{\mathsf{CS}_1, ..., \mathsf{CS}_c\}$,

$$\mathsf{CS}_i = (\mathsf{KDF}, \mathsf{AE}_i, \mathsf{DerIvKDF}, \mathsf{DerIv}_i),\ i = 1, ..., c, \quad \textit{is bounded by}$$

$$\mathrm{Adv}^{NAE}_{\mathsf{CRISP}}(t, q, \nu) \leq \mathrm{Adv}^{VO-PRF}_{\mathsf{KDF}}(t', \kappa) + \sum_{j=1}^{\kappa} \mathrm{Adv}^{NAE}_{\mathsf{AE}^{(j)}}(t', q^{(j)}, \nu^{(j)}),$$

$$\textit{where } \kappa \leq q + \nu, \quad \sum_{j=1}^{\kappa} q^{(j)} = q, \quad \sum_{j=1}^{\kappa} \nu^{(j)} = \nu, \quad \mathsf{AE}^{(j)} \in \{\mathsf{AE}_1, ..., \mathsf{AE}_c\}.$$

*Provided that:*

*1) the input of* KDF *contains* $S_{ID}$, $CS$, $\mathsf{DerIvKDF}(SN)$;

*2) for any* $SN \neq SN'$: $\mathsf{DerIvKDF}(SN) \neq \mathsf{DerIvKDF}(SN')$ *or/and* $\mathsf{DerIv}_i(SN) \neq \mathsf{DerIv}_i(SN')$, $i = 1, ..., c$.

[2] The assumption is adequate to the practice. The opposite will require more complex definitions, but does not generate any vulnerabilities. In addition, external data is presented in the packet only "virtually".

The idea of the proof is simple. Restrictions on the use of KDF make it easy to "replace" it with an ideal primitive. Due to this, we get $\kappa$ *independent* cryptosystems. It remains to apply the "hybrid argument" corresponding to the sum in the estimate. The complete proof is presented in Appendix B.

**Corollary**. *Let the cipher suites in queries to the left (resp. right) oracle belong to the set* $\mathbf{CS}_{\mathcal{S}}$ *(resp.* $\mathbf{CS}_{\mathcal{R}}$*), and* $\mathbf{CS}_{\mathcal{S}} \cap \mathbf{CS}_{\mathcal{R}}$ *are shared, then*

$$\mathrm{Adv}^{NAE}_{\mathsf{CRISP}}(t,q,\nu) \le \mathrm{Adv}^{VO-PRF}_{\mathsf{KDF}}(t',\kappa=\kappa'+\kappa''+\kappa''')+$$
$$+\sum_{j=1}^{\kappa'}\mathrm{Adv}^{NAE}_{\mathsf{AE}^{(j)}}(t',q^{(j)},0)+\sum_{j=\kappa'+1}^{\kappa''}\mathrm{Adv}^{NAE}_{\mathsf{AE}^{(j)}}(t',q^{(j)},\nu^{(j)})+\sum_{j=\kappa''+1}^{\kappa'''}\mathrm{Adv}^{NAE}_{\mathsf{AE}^{(j)}}(t',0,\nu^{(j)}),$$

*where for* $j=1,...,\kappa'$, $j=\kappa'+1,...,\kappa''$, $j=\kappa''+1,...,\kappa'''$, $\mathsf{CS}^{(j)}$ *belongs to* $\mathbf{CS}_{\mathcal{S}}\backslash\mathbf{CS}_{\mathcal{R}}$, $\mathbf{CS}_{\mathcal{S}}\cap\mathbf{CS}_{\mathcal{R}}$, $\mathbf{CS}_{\mathcal{R}}\backslash\mathbf{CS}_{\mathcal{S}}$, *correspondingly.*

The security of CRISP against privacy attacks is determined by the weakest set of the sender ($\mathbf{CS}_{\mathcal{S}}$). Forgery attacks can achieve the greatest efficiency when a shared set from $\mathbf{CS}_{\mathcal{S}} \cap \mathbf{CS}_{\mathcal{R}}$ is used (by using $q^{(j)}$ packets protected with the same key), or when a "vulnerable" set supported only by the receiver ($\mathbf{CS}_{\mathcal{R}}\backslash\mathbf{CS}_{\mathcal{S}}$) is used. In the latter case, attempts to forge will essentially be carried out "blindly", this corresponds to zero in $\mathrm{Adv}^{NAE}_{\mathsf{AE}^{(j)}}(t',0,\nu^{(j)})$. A well-known example of the mentioned "vulnerability" is the short tag length, and the corresponding only possible attack is a simple guessing.

Recall that the results above describe a case where the participants have only one shared key. The "many keys" scenario can be reduced in a typical way to the analysis of many single-key independent systems using the "hybrid argument". Obtaining non-trivial results in such conditions is the subject of further research. It seems that this is possible when the protocol is used only to protect integrity, and the sender's ID is included in the packet explicitly.

Also note that considering many receivers instead of one does not seem to lead to meaningful changes in given proofs. Each receiver processes incoming messages independently of the others. The package does not contain a field with any receiver identifier, it is assumed that the receiver can be anyone who has a master key. In practice, the number of forgery attempts $\nu$ usually increases linearly with the number of users. In other words, the case of "many receivers" does not lead to a new threat model, but to an increase in adversary resources.

### 4.3 Replay protection

The security of the CRISP protocol to replay attacks is almost obvious. Indeed, each receiver has the window $W$ of received messages. If a message

with a certain sequence number $SN$ is accepted, then this fact is stored in the window. The second time a message with the same number will be rejected regardless of the content. Messages with numbers less than the lower bound $\underline{SN}$ are also rejected without consideration. The formal proof of this (including cumbersome definitions similar to those proposed in [23, 24, 25, 26]), is not so trivial and due to lack of space, we omit it here.

### 4.4 Security with leakage of keys

Thanks to the $PRF$-security of KDF, the CRISP protocol continues to provide some security properties in conditions when some keys become known to an attacker. Obviously, when the master key $K$ is leaked, no security properties are preserved.

If there is a leak of one encryption key $K_{ENC}$, then the confidentiality of $q'$ messages is violated. The maximum value of $q'$ depends on the algorithm DerIvKDF, that is, from the frequency of changing encryption keys.

If the adversary learns one authentication key $K_{MAC}$, then each receiver recognizes up to $q'$ forged packets as authentic. Note that in any case, the enemy cannot impose even two packages with the same $SN$, hence each forgery increases the counter by at least one. Consequently, several forgeries will lead to the key change.

If any number of derived keys is leaked, the adversary cannot efficiently determine the value of any other derived key (and even more so the master key). The opposite would mean that KDF is not $PRF$-secure.

## 5 Analysis of the existing cipher suites

The existing version of the CRISP specification contains four "paired" cipher suites (see the table below).

| $CS$ | Name | Integrity | Confidentiality | Tag length ($\tau$ bit) |
|---|---|---|---|---|
| 1 | MAGMA-CTR-CMAC | + | + | 32 |
| 2 | MAGMA-NULL-CMAC | + | − | 32 |
| 3 | MAGMA-CTR-CMAC8 | + | + | 64 |
| 4 | MAGMA-NULL-CMAC8 | + | − | 64 |

All of them use the block cipher "Magma" [1] $\mathsf{E} : V^k \times V^n \to V^n$ with a key length of $k = 256$ bits and a block length of $n = 64$ bits.

According to GOST R 34.13-2015 [2], the counter mode CTR is used for encryption and CMAC ensures the integrity of the messages. The nonce for

the counter mode is the 32 least significant bits of the sequence number

$$IV = \mathsf{DerIv}(SN) = \mathsf{lsb}_{n/2}(SN),\ SN \in V^{48}.$$

The key derivation function $\mathsf{KDF} : V^k \times V^{\leq L} \times \mathbb{N} \to (V^n)^{\leq d}$ is based on several different calls of $\mathsf{CMAC}$

$$\begin{aligned}\Gamma = \mathsf{KDF}(K, X, d) =& \mathsf{CMAC[E]}(K, \mathsf{byte}(1,1)||X||\mathsf{byte}(n\cdot d, 2))||\\ & \mathsf{CMAC[E]}(K, \mathsf{byte}(2,1)||X||\mathsf{byte}(n\cdot d, 2))||\\ & ...\\ & \mathsf{CMAC[E]}(K, \mathsf{byte}(d,1)||X||\mathsf{byte}(n\cdot d, 2)),\end{aligned}$$

$\mathsf{byte}(x, j)$ is the representation of an integer $x$ as a byte string of length $j$. The derived keys are computed as

$$K_{MAC}||K_{ENC} = \Gamma,\ d = \frac{2\cdot k}{n} = 8, \text{ with } CS \in \{1, 3\},$$
$$K_{MAC} = \Gamma,\ d = \frac{k}{n} = 4, \quad \text{with } CS \in \{2, 4\}.$$

The input data $X$ for $\mathsf{KDF}$ contains, among other things:
– the number $CS$ of the cipher suite;
– the source identifier $S_{ID}$;
– 35 most significant bits of the sequence number $SN \in V^{48}$ $\mathsf{DerIvKDF}(SN) = \mathsf{msb}_{35}(SN)$.

In $\mathsf{KDF}$ the input length of $\mathsf{CMAC}$ does not exceed 50 bytes (seven $n$-bit blocks, $l_{KDF} = 7$). Note that due to the dependency of $\mathsf{KDF}$ from 35 bits of $SN$, no more than $2^{48}/2^{35} = 2^{13}$ packets are processed with the same derived key (or key pair).

### 5.1 Known bounds for the cipher modes

We list the known bounds in relevant threat models for the ciphers modes used in cipher suites 1-4. Recall that $q$ is the number of protected messages (queries to the oracle); $l$ is the maximum length of a single message in $n$-bit blocks.

The security proof of $\mathsf{CTR[E]}$ in the $IND\text{-}CPNA$ model (see the definition in Appendix A) is essentially a consequence of the PRP-PRF Switching Lemma [16] due to which [9]:

$$\mathrm{Adv}^{IND\text{-}CPNA}_{\mathsf{CTR[E]}}(t, q, l) \leq \mathrm{Adv}^{PRP}_{\mathsf{E}}(t', q\cdot l) + \frac{(q\cdot l)^2}{2^{n+1}}\ t' = t + O(ql).$$

For CMAC a number of estimates in the $PRF$ model are known [17, 18, 19, 20], we quote the last of them.

**Theorem** ([20, Theorem 3.1]). *The advantage of the adversary in the $PRF$ model attacking the cryptoalgorithm* CMAC *is bounded by*[3]

$$\mathrm{Adv}^{PRF}_{\mathsf{CMAC[E]}}(t,q,l) \le \mathrm{Adv}^{PRP}_{\mathsf{E}}(t', q\cdot l+1) + \frac{16\cdot q^2 + q\cdot l^2 + 4\cdot q\cdot l}{2^n} + \epsilon(q,l),$$

*where* $t' = t + O(q\cdot l)$, $q\cdot(l+1) \le 2^{n-1}$.

$PRF$-security of CMAC is sufficient for $VO$-$PRF$-security of CMAC-based $\mathsf{KDF}(K, X, d)$. In other words, KDF is indistinguishable from random function when the input and the output lengths are not constant. Let the adversary queries to KDF be $(X_1, d_1)$,...,$(X_q, d_q)$, and $(X_i, d_i) \ne (X_j, d_j)$, $1 \le i < j \le q$. It is easy to see that under such conditions all inputs to the underlying CMAC[E] are different, hence

$$\mathrm{Adv}^{VO\text{-}PRF}_{\mathsf{KDF[CMAC[E]]}}(t,q) \le \mathrm{Adv}^{PRF}_{\mathsf{CMAC[E]}}(t', q\cdot d, l_{KDF} = 7).$$

## 5.2 AEAD-security of composition

It is well known that the algorithms of cipher suites 1-4 produce secure authenticated encryption modes.

**Lemma 1**. *The advantage of the adversary in the $NAE$ model attacking the cryptoalgorithm*

$$\mathsf{CTR\text{-}CMAC} : \mathbf{K}\times\mathbf{A}\times\mathbf{P}\to\mathbf{C}\times\mathbf{T},$$
$$\mathsf{CTR\text{-}CMAC} : (V^k\times V^k)\times V^{\le l\cdot n}\times V^{\le l\cdot n}\to V^{\le l\cdot n}\times V^{\tau},\ \textit{is bounded by}$$

$$\mathrm{Adv}^{NAE}_{\mathsf{CTR\text{-}CMAC}}(t,q,\nu) \le \mathrm{Adv}^{PRF}_{\mathsf{CMAC[E]}}(t', q+\nu, l) + \mathrm{Adv}^{IND-CPNA}_{\mathsf{CTR[E]}}(t', q, l) + \frac{\nu}{2^{\tau}},$$

$t' = t + O((q+\nu)\cdot l)$. *The query from the adversary to the left oracle is $(A, P)$ and $A = H$.*

**Lemma 2**. *The advantage of the adversary in the $NAE$ model attacking the cryptoalgorithm*

$$\mathsf{NULL\text{-}CMAC} : \mathbf{K}\times\mathbf{A}\times\mathbf{P}\to\mathbf{C}\times\mathbf{T},$$
$$\mathsf{NULL\text{-}CMAC} : V^k\times V^{\le l\cdot n}\times\emptyset\to\emptyset\times V^{\tau},\ \textit{is bounded by}$$

$$\mathrm{Adv}^{NAE}_{\mathsf{NULL\text{-}CMAC}}(t,q,\nu) \le \mathrm{Adv}^{PRF}_{\mathsf{CMAC[E]}}(t', q+\nu, l) + \frac{\nu}{2^{\tau}},\ t' = t + O((q+\nu)\cdot l).$$

*The query from the adversary to the left oracle is $(A, \emptyset)$, $A = H||P$.*

[3]The value of $\epsilon(q,l)$ has a bulky form and from a practical point of view is approximately zero. So, for compactness, we omit it.

Thus, both AEAD-modes satisfy the requirements of theorem 1.

The proof of the first lemma is presented in Appendix B. The proof of the second is a direct consequence of the first for the case $\mathbf{P} = \mathbf{C} = \emptyset$.

### 5.3 Estimates of key capacity

Further, by "key capacity" we mean the permissible amount of data processed under a single key until it should be rotated. Here we discuss approaches to its calculation.

In [3], the concepts of "the maximum allowable probability of a single forgery" ($\pi_{\text{mac}}$) and "the maximum allowable probability of successful application of cryptanalysis" ($\pi_{\text{enc}}$) are defined. The $NAE$ model includes both integrity attacks (forgeries) and privacy attacks (for example, "reading without key"), hence, for any used $\mathsf{Alg}$ the inequality must hold true

$$\text{Adv}^{NAE}_{\mathsf{Alg}}(t, q, \nu) < \pi = \min(\pi_{\text{enc}}, \pi_{\text{mac}}).$$

For illustrative purposes, we choose $\min(\pi_{\text{enc}}, \pi_{\text{mac}}) = 2^{-10}$.

For adversary, we assume, though greatly exaggerating one's real capabilities, that his computational resources are equal to $t \approx t' \approx 2^{128}$ operations.

Recall that $\kappa$ is the number of derived keys, $q$ (resp. $q' = 2^{13}$) is the number of packets protected with one master (resp. derived) key. For simplicity, we also assume that due to some technical protection, the corresponding number of forgery attempts $\nu$ (resp. $\nu'$) is much less than $q$ (resp. $q'$). The maximum packet length is $l \leq \frac{2048 \cdot 8}{n} = 2^8$ blocks. $\mathsf{KDF}$ uses $d \in \{4, 8\}$ calls of $\mathsf{CMAC}$ per one derived key.

The security of all cipher suites reduces to $PRP$-security of the "Magma". Even in the light of existing attacks [29, 30, 31], with the declared $t$ the distinguishing advantage can be considered equal to *zero* for most purposes $\text{Adv}^{PRP}_{\mathsf{Magma}} \approx 0$ (see details in Appendix C).

Thus, summing up the above and simplifying the estimates to the most significant terms, we get:

$$\begin{aligned}
\varepsilon_{KDF} &\leq \text{Adv}^{PRF}_{\mathsf{CMAC}}(t', \kappa \cdot d = 2^{21} \cdot 8, l_{KDF} = 7) &&\lessapprox \frac{16 \cdot (\kappa \cdot d)^2}{2^n} = 2^{-12}, \\
\varepsilon_{CTR} &= \text{Adv}^{IND-CPNA}_{\mathsf{CTR}}(t', q' + \nu' \approx 2^{13}, l = 2^8) &&\lessapprox \frac{(q' \cdot l)^2}{2^{n+1}} = 2^{-23}, \\
\varepsilon_{CMAC} &= \text{Adv}^{PRF}_{\mathsf{CMAC}}(t', q' + \nu' \approx 2^{13}, l = 2^8) &&\lessapprox \frac{16 \cdot (q')^2}{2^n} = 2^{-34}.
\end{aligned}$$

For the first and the third $CS$, $\varepsilon_{CS} \approx \varepsilon_{CTR} + \varepsilon_{CMAC} \approx \varepsilon_{CTR}$. For the other two ($CS \in \{2, 4\}$), $\varepsilon_{CS} \approx \varepsilon_{CMAC}$.

It is not difficult to see that for each cipher suite $\varepsilon_{CS} < \pi$, the same is true for $\varepsilon_{KDF}$ when the number of derived keys $\kappa \leq 2^{21}$. In other words, if, as usual, we consider each derived key *separately*, then "the protocol is secure" with the above restriction on $\kappa$.

On the other hand, if we consider the whole protocol and all the keys, then the restriction is now imposed on the sum

$$\mathrm{Adv}^{NAE}_{\mathsf{CRISP}}(t,q,\nu) \leq \mathrm{Adv}^{VO-PRF}_{\mathsf{KDF}}(t',\kappa)+\kappa\cdot\mathrm{Adv}^{NAE}_{\mathsf{CS}}(t',q',\nu') = \varepsilon_{KDF}+\kappa\cdot\varepsilon_{CS},$$

where the second term acquires the greatest importance. So, for the first cipher suite, from $(\varepsilon_{KDF} + \kappa \cdot \varepsilon_{CS}) < \pi$ follows $\kappa < 2^{13}$. The latter is a very strict limitation for practice. In addition, if we replace the derived keys with truly random ones, then $(\kappa \cdot \varepsilon_{CS}) < \pi$ and the estimate does *not* change. In other words, KDF and derived keys do not make CRISP worse.

We emphasize that the above is not an "artifact of provable security". A similar result can be obtained from a constructive point of view, in the sense of: "the probability of a successful attack on *any one* from $\kappa$ cryptosystems is approximately $\kappa$ times greater than the similar probability for one pre-chosen cryptosystem". The choice of the first (each key separately) or second (all keys in the entire system) approaches should be made based on the requirements for a specific information system.

It should be noted that there are many ways to increase the key capacity.

Perhaps the most effective is the use of a cipher with a relatively large block size, namely "Kuznyechik" [1], with $n = 128$, the value of $\kappa$ is greater than the "unreachable" $2^{54}$.

The greatest contribution to the final estimate is made by $\varepsilon_{CTR}$, which degrades quadratically with the growth of the number of blocks. Consequently, some improvements can be achieved by using: the internal re-keying as realized in CTR-ACPKM [5]; truncating of the block cipher output to $s < n$ bits (as provided by the standard [2]); double application of CTR.

# 6 Conclusion

Using the provable security approach [8, 9] to the analysis of the cryptoalgorithms, we formally proved that the CRISP protocol [4] provides confidentiality, integrity and protection against replays.

CRISP was considered as the algorithm of the authenticated encryption with associated data (AEAD) in the relevant threat model.

We presented the list of sufficient requirements for the cipher suites used in CRISP. The main ones are:

1) the cipher suites used with the same master key must have the same $PRF$-secure key derivation function;

2) the encryption algorithm and the message authentication code algorithm, applied consequently, must form a secure deterministic AEAD-scheme.

The existing cipher suites [4] satisfy all the specified requirements.

The obtained estimates allowed us to form motivated recommendations on the key capacity.

# 7 Acknowledgements

The author thanks Alexey Urivskiy, Olga Shemyakina, Andrey Rybkin and Mikhail Borodin for useful discussions. The author is grateful to Andrey Shcherbachenko for many valuable suggestions and corrections. Detailed comments and a list of inaccuracies from anonymous reviewers considerably improved the article – special thanks!

# References

[1] *GOST R 34.12-2015 – National standard of the Russian Federation – Information technology – Cryptographic data security – Block ciphers*, 2015.

[2] *GOST R 34.13-2015 – National standard of the Russian Federation – Information technology – Cryptographic data security – Block ciphers operation modes*, 2015.

[3] *R 1323565.1.005-2017 – Information technology – Cryptographic data security – Acceptable amount of data to be processed without key change for particular block cipher modes of operation GOST R 34.13-2015*, 2017.

[4] *R 1323565.1.029-2019 – Information technology – Cryptographic data security – Secure exchange protocol for industrial systems*, 2020.

[5] *R 1323565.1.017-2018 – Information technology – Cryptographic data security – Cryptographic algorithms accompanying the use of block encryption algorithms*, 2018.

[6] Wegman M., Carter L., "New hash functions and their use in authentication and set equality", *Journal of Computer and System Sciences*, **22** (1981), 265–279.

[7] Black J., Halevi S., Krawczyk H., Krovetz T., Rogaway P., "UMAC: Fast and Secure Message Authentication", CRYPTO '99, Lect. Notes Comput. Sci., **1666**, 1999, 216–233..

[8] Bellare M., Rogaway P., *Introduction to Modern Cryptography*, 2005.

[9] Rogaway P., *Evaluation of Some Blockcipher Modes of Operation*, CRYPTREC, 2011.

[10] McGrew D. A., Viega J., "The Security and Performance of the Galois/Counter Mode (GCM) of Operation", INDOCRYPT 2004, Lect. Notes Comput. Sci., **3348**, 2004, 343–355.

[11] Armando A. et al., "The AVISPA Tool for the Automated Validation of Internet Security Protocols and Applications.", CAV 2005, Lect. Notes Comput. Sci., **3576**, 2005, 281–285.

[12] Canetti R., Krawczyk H., "Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels", EUROCRYPT 2001, Lect. Notes Comput. Sci., **2045**, 2001, 453–474.

[13] LaMacchia, B., Lauter, K., Mityagin, A., "Stronger Security of Authenticated Key Exchange", ProvSec 2007, Lect. Notes Comput. Sci., **4784**, 2007, 1–16.

[14] Krawczyk H., "The Order of Encryption and Authentication for Protecting Communications (Or: How Secure is SSL?)", CRYPTO 2001, Lect. Notes Comput. Sci., **2139**, 2001, 310–331.

[15] Canvel B., Hiltgen A., Vaudenay S., Vuagnoux M., "Password interception in a SSL/TLS channel", CRYPTO 2003, Lect. Notes Comput. Sci., **2729**, 2003, 583–599.

[16] Chang D., Nandi M., "A Short Proof of the PRP/PRF Switching Lemma", *Cryptology ePrint Archive, Report 2008/078*, 2008.

[17] Iwata T., Kurosawa K., "OMAC: One-Key CBC MAC", FSE 2003, Lect. Notes Comput. Sci., **2887**, 2003, 129–153.

[18] Iwata T., Kurosawa K., "Stronger Security Bounds for OMAC, TMAC and XCBC", INDOCRYPT 2003, Lect. Notes Comput. Sci., **2904**, 2003, 402–415.

[19] Nandi M., "Improved security analysis for OMAC as a pseudorandom function", *Journal of Mathematical Cryptology*, **3**:2 (2009), 133–148.

[20] Chattopadhyay S., Jha A., Nandi M., "Towards Tight Security Bounds for OMAC, XCBC and TMAC", ASIACRYPT 2022, 2022.

[21] Bellare M., Namprempre C., "Authenticated Encryption: Relations among notions and analysis of the generic composition paradigm", ASIACRYPT 2000, Lect. Notes Comput. Sci., **1976**, 2000, 531–545.

[22] Shrimpron T., "A Characterization of Authenticated-Encryption as a Form of Chosen-Ciphertext Security", *Cryptology ePrint Archive, Report 2004/272*, 2004.

[23] Kohno T., Palacio A., Black J., "Building Secure Cryptographic Transforms, or How to Encrypt and MAC", *Cryptology ePrint Archive, Report 2003/177*, 2003.

[24] Bellare M., Kohno T., Namprempre C., "Breaking and Provably Repairing the SSH Authenticated Encryption Scheme: A Case Study of the Encode-then-Encrypt-and-MAC Paradigm", *ACM Transactions on Information and Systems Security*, **7**:2 (2004), 206–241.

[25] Boyd C., Hale B., Mjølsnes S. F., Stebila D., "From Stateless to Stateful: Generic Authentication and Authenticated Encryption Constructions with Application to TLS", Cryptographers Track at the RSA Conference 2016, 2016, 55–71.

[26] Rogaway P., Zhang Y., "Simplifying Game-Based Definitions Indistinguishability up to Correctness and Its Application to Stateful AE", CRYPTO 2018, Lect. Notes Comput. Sci., **10992**, 2018, 3–32.

[27] Biham, E., Shamir, A., "Differential cryptanalysis of DES-like cryptosystems", *J. Cryptology*, 1991, 3–72.

[28] Matsui M., "Linear cryptanalysis method for DES cipher", EUROCRYPT'93, Lect. Notes Comput. Sci., **765**, 1994, 386–397.

[29] Isobe T., "A Single-Key Attack on the Full GOST Block Cipher", FSE 2011, Lect. Notes Comput. Sci., **6733**, 2011, 290–305.

[30] Dinur I., Dunkelman O., Shamir A., "Improved Attacks on Full GOST", FSE 2012, Lect. Notes Comput. Sci., **7549**, 2012, 9–28.

[31] Kara O., Karakoc F., "Fixed Points of Special Type and Cryptanalysis of Full GOST", CANS 2012, Lect. Notes Comput. Sci., **7712**, 2012, 86–97.

[32] A. A. Dmukh, D. M. Dygin, G. B. Marshalko, "A lightweight-friendly modification of GOST block cipher", *Mat. Vopr. Kriptogr.*, **5**:2 (2014), 47–55.

# A Definitions of the formal models

**Definition.** The advantage of $\mathcal{A}$ in the model $PRP$ ($PRP$-$CPA$ – indistinguishability from a random permutation under chosen plaintext attack) for the keyed cryptoalgorithm $\mathsf{E} : \mathbf{K} \times \mathbf{X} \to \mathbf{X}$ is

$$\mathrm{Adv}^{PRP}_{\mathsf{E}}(\mathcal{A}) = \Pr\left(K \xleftarrow{\mathrm{R}} \mathbf{K}; \mathcal{A}^{\mathsf{E}_K(\cdot)} \Rightarrow 1\right) - \Pr\left(\Pi \xleftarrow{\mathrm{R}} \mathrm{Perm}(\mathbf{X}); \mathcal{A}^{\Pi(\cdot)} \Rightarrow 1\right),$$

where $\mathbf{K}$, $\mathbf{X}$ are spaces of the keys and blocks respectively.

**Definition.** The advantage of $\mathcal{A}$ in the model $PRF$ ($PRF$-$CMA$ – indistinguishability from a random function under chosen message attack) for the keyed cryptoalgorithm $\mathsf{F} : \mathbf{K} \times \mathbf{X} \to \mathbf{Y}$ is

$$\mathrm{Adv}^{PRF}_{\mathsf{F}}(\mathcal{A}) = \Pr\left(K \xleftarrow{\mathrm{R}} \mathbf{K}; \mathcal{A}^{\mathsf{F}_K(\cdot)} \Rightarrow 1\right) - \Pr\left(\mathsf{R} \xleftarrow{\mathrm{R}} \mathrm{Func}(\mathbf{X}, \mathbf{Y}); \mathcal{A}^{\mathsf{R}(\cdot)} \Rightarrow 1\right),$$

where $\mathbf{K}$, $\mathbf{X}$, $\mathbf{Y}$ are spaces of the keys, messages, and outputs respectively.

**Definition.** The advantage of $\mathcal{A}$ in the model $VO$-$PRF$ (variable output – indistinguishability from a random function with variable output length) for the keyed cryptoalgorithm $\mathsf{F} : \mathbf{K} \times V^* \times \mathbb{N} \to V^*$ is

$$\begin{aligned}\mathrm{Adv}^{VO\text{-}PRF}_{\mathsf{F}}(\mathcal{A}) = &\Pr\left(K \xleftarrow{\mathrm{R}} \mathbf{K} : \mathcal{A}^{\mathsf{F}_K(\cdot,\cdot)} \Rightarrow 1\right) - \\ &- \Pr\left(\mathsf{R} \xleftarrow{\mathrm{R}} \mathrm{Func}(V^* \times \mathbb{N}, V^*) : \mathcal{A}^{\mathsf{R}(\cdot,\cdot)} \Rightarrow 1\right).\end{aligned}$$

The query from $\mathcal{A}$ to the oracle is $(X, L) \in V^* \times \mathbb{N}$, where $X$ is data and $L$ is the output length in bits.

**Definition**. The advantage of $\mathcal{A}$ in the distinguishing two cryptosystems $\mathsf{S}$ and $\widetilde{\mathsf{S}}$ (with the same interfaces) is

$$\mathrm{Adv}^{IND}_{\mathsf{S},\widetilde{\mathsf{S}}}(\mathcal{A}) = \Pr\left(\mathcal{A}^{\mathsf{S}(\cdot)} \Rightarrow 1\right) - \Pr\left(\mathcal{A}^{\widetilde{\mathsf{S}}(\cdot)} \Rightarrow 1\right).$$

**Definition.** The advantage of $\mathcal{A}$ in the model $IND$-$CPNA$ (indistinguishability under chosen plaintext and nonce attack, also denoted as priv) for the encryption mode $\mathsf{EncMode} : V^k \times V^s \times V^{\leq L} \to V^{\leq L}$ is

$$\mathrm{Adv}^{IND\text{-}CPNA}_{\mathsf{EncMode}}(\mathcal{A}) = \Pr\left(K \xleftarrow{\mathrm{R}} V^k : \mathcal{A}^{\mathsf{EncMode}(K,\cdot,\cdot)} \Rightarrow 1\right) - \Pr\left(\mathcal{A}^{\$(\cdot,\cdot)} \Rightarrow 1\right).$$

$\mathsf{EncMode}(K, \cdot, \cdot)$ is the encryption oracle, that receives the query $(N, P) \in V^s \times V^{\leq L}$ and returns the ciphertext $C \in V^{\leq L}$, where $|C| = |P| + ext(P)$. The oracle $\$(\cdot, \cdot)$ receives the query $(N, P) \in V^s \times V^{\leq L}$ and returns a random binary string of length $|P| + ext(P)$ bits. The adversary $\mathcal{A}$ cannot repeat the value $N$, each value of $N \in V^s$ is unique. The adversary $\mathcal{A}$ makes $q$ queries of no more than $l$ $n$-bit blocks each, $l \cdot n \leq L$. The extension function $ext(P)$ computes the length of the required padding.

# B Proofs

**Theorem 1.** *The advantage of the adversary in the $NAE$ model attacking the* CRISP *that uses the cipher suites from the set* $\mathbf{CS} = \{\mathsf{CS}_1, ..., \mathsf{CS}_c\}$,

$$\mathsf{CS}_i = (\mathsf{KDF}, \mathsf{AE}_i, \mathsf{DerIvKDF}, \mathsf{DerIv}_i),\ i = 1, ..., c, \quad \textit{is bounded by}$$

$$\mathrm{Adv}^{NAE}_{\mathsf{CRISP}}(t, q, \nu) \le \mathrm{Adv}^{VO-PRF}_{\mathsf{KDF}}(t', \kappa) + \sum_{j=1}^{\kappa} \mathrm{Adv}^{NAE}_{\mathsf{AE}^{(j)}}(t', q^{(j)}, \nu^{(j)}),$$

$$\textit{where } \kappa \le q + \nu, \quad \sum_{j=1}^{\kappa} q^{(j)} = q, \quad \sum_{j=1}^{\kappa} \nu^{(j)} = \nu, \quad \mathsf{AE}^{(j)} \in \{\mathsf{AE}_1, ..., \mathsf{AE}_c\}.$$

*Provided that:*

*1) the input of* KDF *contains $S_{ID}$, $CS$,* DerIvKDF$(SN)$*;*

*2) for any $SN \neq SN'$:* DerIvKDF$(SN) \neq$ DerIvKDF$(SN')$ *or/and* DerIv$_i(SN) \neq$ DerIv$_i(SN')$, $i = 1, ..., c$.

Proof.

Recall that here we consider the protocol with a single pre-shared master key $K$. Many senders (each with a unique identifier $S_{ID}$) use $K$.

The adversary, according to the $NAE$ model, has access to the pair of oracles. The left "encryption" oracle emulates all senders, the right "verification" oracle corresponds to one receiver. The adversary chooses a specific sender by manipulating associated data $A$, including unprotected "imaginary" external data $A_{\text{ext}}$, and fields `KeyId` and `ExternalKeyIdFlag` in the header $H$. According to the assumptions described earlier, an arbitrary change in external data or/and these fields entails a change in $(K, S_{ID})$. Due to the uniqueness of $K$, this means that the $S_{ID}$ must be changed.

According to the requirements of the theorem, all cipher suites use the same KDF. Consider the CRISP-I protocol, in which a random function $\mathsf{R} \in \mathrm{Func}(V^* \times \mathbb{N}, V^*)$ is used instead of KDF. Let's construct algorithm $\mathcal{B}$ so that the inequality holds

$$\mathrm{Adv}^{IND}_{\mathsf{CRISP},\mathsf{CRISP\text{-}I}}(\mathcal{A}) \le \mathrm{Adv}^{VO\text{-}PRF}_{\mathsf{KDF}}(\mathcal{B}).$$

Each query from $\mathcal{A}$ to either of its two oracles may require the computation of derived keys. Algorithm $\mathcal{B}$ emulates one of two protocols (CRISP or CRISP-I) for $\mathcal{A}$. Hence, $\mathcal{B}$ makes up to $\kappa \le (q + \nu)$ queries to the oracle (KDF or R). One response from the oracle is the derived key $K^{(j)}$ for the cipher mode $\mathsf{AE}^{(j)}$, $j = 1, ..., \kappa$. So, $\mathcal{B}$ has all derived keys and, therefore, can perfectly simulate the protocol. The result of $\mathcal{B}$ is equal to the result of $\mathcal{A}$.

In fact, CRISP-I contains $\kappa$ subsystems (i.e. some $\mathsf{AE}^{(j)}$ with the key $K^{(j)}$) *independent* of each other. By virtue of condition 1, the following subsystems have independent keys:

– with different cipher suites ($CS$ is the part of KDF input);

– with the same cipher suites, but with the different senders ($S_{ID}$ is also the part of KDF input);

– with the same cipher suites and the same $S_{ID}$, but with different $\mathsf{DerIvKDF}(SN)$.

Recall that the pair $(S_{ID}, SN)$ is considered as nonce in CRISP and CRISP-I – the sender does not repeat its own sequence numbers. Along with this, different $SN$ can correspond to the same $IV = \mathsf{DerIv}_i(SN)$ for some $i = 1, ..., c$. Due to condition 2, within any of the $\kappa$ subsystems, the $IV$ values are also not repeated.

The adversary chooses one subsystem for interaction by specifying associated data $A$ (this includes $S_{ID}$, $CS$, $SN$) in the query.

In the $j$-th system, the adversary can make $q^{(j)}$ (resp. $\nu^{(j)}$) queries to the "encryption" (resp. "verification") oracle. The maximum of $q^{(j)}$ depends on the number of bits in $SN$ that do not affect $\mathsf{DerIvKDF}(SN)$. All forgery attempts can be carried out using a single $(S_{ID}, CS, SN)$, hence, $\nu^{(j)} \leq \nu$. The total numbers of queries are $\sum_{j=1}^{\kappa} q^{(j)} = q$ and $\sum_{j=1}^{\kappa} \nu^{(j)} = \nu$.

Thanks to the independence of the keys in CRISP-I, we can use the so-called "hybrid argument". Let we have the sequence of the protocols[4]

$$\mathsf{CRISP\text{-}I}^{(0)}, ..., \mathsf{CRISP\text{-}I}^{(\kappa)},$$

where $\mathsf{CRISP\text{-}I}^{(0)} = \mathsf{CRISP\text{-}I}$ and $\mathsf{CRISP\text{-}I}^{(\kappa)}$ is the "ideal" (all $\kappa$ pairs of oracles are $(\$, \perp)$). In $\mathsf{CRISP\text{-}I}^{(j)}$, $0 < j < \kappa$, all pairs of oracles with indexes $1, ..., j$, are replaced by the "ideal" $(\$, \perp)$ ones, all other pairs are "real" $(j+1, ..., \kappa)$.

If $\mathcal{A}$ can effectively distinguish $\mathsf{CRISP\text{-}I}^{(j-1)}$ and $\mathsf{CRISP\text{-}I}^{(j)}$, then there is $\mathcal{B}^{(j)}$ that can effectively attack $\mathsf{AE}^{(j)}$ in the $NAE$ model. Before starting interactions, $\mathcal{B}^{(j)}$ generates keys $K^{(j')}$ for subsystems with indexes $j' > j$. After that, for any query from $\mathcal{A}$, $\mathcal{B}^{(j)}$ determines the index $j'$ of the oracle pairs by the associated data in the query. If $j' < j$, then $\mathcal{B}^{(j)}$ simulates "ideal" oracle ($\$ or $\perp$). If $j' = j$, then $\mathcal{B}^{(j)}$ makes the corresponding query to its own oracle and returns the response to $\mathcal{A}$. In other cases ($j' > j$), $\mathcal{B}^{(j)}$ simulates "real" oracle by using a self-generated key $K^{(j')}$. If $\mathcal{B}^{(j)}$ interacts with the "real" (resp. "ideal") oracles, then $\mathsf{CRISP\text{-}I}^{(j-1)}$ (resp. $\mathsf{CRISP\text{-}I}^{(j)}$) is perfectly simulated for $\mathcal{A}$. The result of $\mathcal{B}^{(j)}$ is equal to the result of $\mathcal{A}$.

[4]Speaking more formally, we can enumerate all possible $\kappa_{\max}$ triples $(S_{ID}, CS, \mathsf{DerIvKDF}(SN))$ and consider all corresponding subsystems. In general, $\kappa_{\max} \geq \kappa$, but the adversary does not make any queries to $(\kappa_{\max} - \kappa)$ systems, and hence, using $\kappa_{\max}$ instead of $\kappa$ does not affect the result.

The advantage of $\mathcal{A}$ is bounded by

$$\mathrm{Adv}^{IND}_{\mathsf{CRISP\text{-}I}^{(j-1)},\mathsf{CRISP\text{-}I}^{(j)}}(\mathcal{A}) \le \mathrm{Adv}^{NAE}_{\mathsf{AE}^{(j)}}(\mathcal{B}_j),$$

$\mathcal{B}_j$ makes $q^{(j)}$ and $\nu^{(j)}$ queries.

By the triangle inequality we obtain

$$\mathrm{Adv}^{IND}_{\mathsf{CRISP\text{-}I}^{(0)},\mathsf{CRISP\text{-}I}^{(\kappa)}}(\mathcal{A}) \le \sum_{j=1}^{\kappa} \mathrm{Adv}^{NAE}_{\mathsf{AE}^{(j)}}(\mathcal{B}_j),$$

and due to the arbitrariness of the algorithm $\mathcal{A}$ the original statement of the theorem is true.

□

**Lemma 1**. *The advantage of the adversary in the NAE model attacking the cryptoalgorithm*

$\mathsf{CTR\text{-}CMAC} : \mathbf{K} \times \mathbf{A} \times \mathbf{P} \to \mathbf{C} \times \mathbf{T}$,

$\mathsf{CTR\text{-}CMAC} : (V^k \times V^k) \times V^{\le l\cdot n} \times V^{\le l\cdot n} \to V^{\le l\cdot n} \times V^{\tau}$, *is bounded by*

$$\mathrm{Adv}^{NAE}_{\mathsf{CTR\text{-}CMAC}}(t,q,\nu) \le \mathrm{Adv}^{PRF}_{\mathsf{CMAC[E]}}(t',q+\nu,l) + \mathrm{Adv}^{IND-CPNA}_{\mathsf{CTR[E]}}(t',q,l) + \frac{\nu}{2^{\tau}},$$

$t' = t + O((q+\nu)\cdot l)$. *The query from the adversary to the left oracle is* $(A,P)$ *and* $A = H$.

Proof.

$\mathsf{CTR\text{-}CMAC}$ is a pair of the algorithms denoted here by

$$(\mathsf{AE}[\mathsf{CTR},\mathsf{CMAC}], \mathsf{AE}^{-1}[\mathsf{CTR},\mathsf{CMAC}]).$$

Recall that the nonce $IV$ is determined by the associated data $A$ and is equal to the 32 least significant bits of the sequence number $SN$.

By definition, the advantage of the adversary $\mathcal{A}$ is

$$\begin{aligned}\mathrm{Adv}^{NAE}_{\mathsf{CTR\text{-}CMAC}}(\mathcal{A}) = \Pr((K_{ENC},K_{MAC}) \xleftarrow{\mathrm{R}} V^k \times V^k;\\ \mathcal{A}^{\mathsf{AE}((K_{ENC},K_{MAC}),\cdot,\cdot),\mathsf{AE}^{-1}((K_{ENC},K_{MAC}),\cdot,\cdot,\cdot)} \Rightarrow 1) -\\ - \Pr(\mathcal{A}^{\$(\cdot,\cdot),\perp(\cdot,\cdot,\cdot)} \Rightarrow 1),\end{aligned}$$

where the oracle \$ returns a random binary string of length $|P| + \tau$ in the response to the query $(A,P)$, and the oracle $\perp$ always returns symbol "$\perp$"

Firstly, $\mathsf{CMAC}$ is replaced by a random function $\mathsf{R} : V^{\le l\cdot n} \to V^{\tau}$. Let $\mathcal{A}$ be able to effectively distinguish the original cryptoalgorithm from the modified one, then there is $\mathcal{B}_1$ that can distinguish $\mathsf{CMAC}$ from a random

function. The algorithm $\mathcal{B}_1$ generates $K_{ENC} \xleftarrow{\mathrm{R}} V^k$ and stores it. The query $(A, P)$ from $\mathcal{A}$ to the left oracle is processed by $\mathcal{B}_1$ as follows: read $IV$ from associated data $A$; compute $C = \mathsf{CTR}(K_{ENC}, IV, P)$; receive the tag by the query to the oracle $T = \mathcal{O}(A||C)$, $\mathcal{O} \in \{\mathsf{CMAC}, \mathsf{R}\}$; return $(C, T)$ to $\mathcal{A}$. The query $(A, C, T)$ from $\mathcal{A}$ to the oracle $\mathsf{AE}^{-1}$ is processed by $\mathcal{B}_1$ as follows: receive the tag $T' = \mathcal{O}(A||C)$; compare $T' = T$; if equality is not satisfied, return character "$\perp$"; otherwise, read $IV$ from $A$ and return $P = \mathsf{CTR}(K_{ENC}, IV, C)$.

$\mathcal{B}_1$ perfectly simulates for $\mathcal{A}$ the cryptoalgorithm $\mathsf{AE}[\mathsf{CTR}, \mathsf{CMAC}]$ or $\mathsf{AE}[\mathsf{CTR}, \mathsf{R}]$ (and, of course, the corresponding $\mathsf{AE}^{-1}$). The result of $\mathcal{B}_1$ is equal to the result of $\mathcal{A}$, hence

$$\begin{aligned}
&\Pr((K_{ENC}, K_{MAC}) \xleftarrow{\mathrm{R}} V^k \times V^k; \\
&\quad \mathcal{A}^{\mathsf{AE}[\mathsf{CTR},\mathsf{CMAC}]((K_{ENC},K_{MAC}),\cdot,\cdot),\mathsf{AE}^{-1}[\mathsf{CTR},\mathsf{CMAC}]((K_{ENC},K_{MAC}),\cdot,\cdot,\cdot)} \Rightarrow 1) - \\
&- \Pr(K_{ENC} \xleftarrow{\mathrm{R}} V^k; \mathsf{R} \xleftarrow{\mathrm{R}} \mathrm{Func}(V^{\leq l \cdot n}, V^\tau); \\
&\quad \mathcal{A}^{\mathsf{AE}[\mathsf{CTR},\mathsf{R}](K_{ENC},\cdot,\cdot),\mathsf{AE}^{-1}[\mathsf{CTR},\mathsf{R}](K_{ENC},\cdot,\cdot,\cdot)} \Rightarrow 1) \leq \mathrm{Adv}^{PRF}_{\mathsf{CMAC}[\mathsf{E}]}(\mathcal{B}_1).
\end{aligned}$$

The number of queries from $\mathcal{B}_1$ is equal to the number of queries from $\mathcal{A}$ and is $q + \nu$. The number of queries from $\mathcal{B}_1$ is equal to $q + \nu$.

Secondly, the oracle $\mathsf{AE}^{-1}$ is replaced by $\perp$. The advantage of the adversary in this case is bounded by the probability of making a forgery in $\nu$ attempts. Let $\mathsf{AE}^{-1}$ receive the query $(A, C, T) \notin \{(A_1, C_1, T_1), ..., (A_q, C_q, T_q)\}$. If $A||C \notin \{A_1||C_1, ..., A_q||C_q\}$, then the guessing probability is

$$\Pr(\mathsf{R}(A||C) = T) = 2^{-\tau}.$$

If $A||C = A_i||C_i$, $i = 1, ..., q$ , then $T \neq T_i$, and hence $\Pr(\mathsf{R}(A||C) = T) = 0$. Therefore, the probability of at least one correct guess in $\nu$ attempts is at most

$$\begin{aligned}
&\Pr(\mathcal{A}^{\mathsf{AE}[\mathsf{CTR},\mathsf{R}](K_{ENC},\cdot,\cdot),\mathsf{AE}^{-1}[\mathsf{CTR},\mathsf{R}](K_{ENC},\cdot,\cdot,\cdot)} \Rightarrow 1) - \\
&- \Pr(\mathcal{A}^{\mathsf{AE}[\mathsf{CTR},\mathsf{R}](K_{ENC},\cdot,\cdot),\perp(\cdot,\cdot,\cdot)} \Rightarrow 1) \leq \frac{\nu}{2^\tau}.
\end{aligned}$$

Thirdly, the oracle $\mathsf{AE}[\mathsf{CTR}, \mathsf{R}]$ is replaced by \$. If replacement can be effectively detected, then there is the algorithm $\mathcal{B}_2$ effectively attacking $CTR$ in the $IND\text{-}CPNA$ model. The query $(A, P)$ from $\mathcal{A}$ to the left oracle is processed by $\mathcal{B}_2$ as follows: read $IV$ from associated data $A$; get $C = \mathcal{O}(IV, P)$ from the oracle $\mathcal{O} \in \{\mathsf{CTR}, \$\}$; generate $T \xleftarrow{\mathrm{R}} V^\tau$; return the response $(C, T)$. Due to the fact that all queries $(A, P)$ are different, the random generation

$T \xleftarrow{\mathrm{R}} V^\tau$ corresponds perfectly to the behavior of a random function $\mathsf{R}$. The response to the query from $\mathcal{A}$ to $\perp$ is trivially simulated. The result of $\mathcal{B}_2$ is equal to the result of $\mathcal{A}$, and therefore we obtain,

$$\Pr(\mathcal{A}^{\mathsf{AE}[\mathsf{CTR},\mathsf{R}](K_{ENC},\cdot,\cdot),\perp(\cdot,\cdot,\cdot)} \Rightarrow 1) - \Pr(\mathcal{A}^{\$(\cdot,\cdot),\perp(\cdot,\cdot,\cdot)} \Rightarrow 1) \leq \mathrm{Adv}^{IND-CPNA}_{\mathsf{CTR}[\mathsf{E}]}(\mathcal{B}_2).$$

Algorithm $\mathcal{B}_2$ makes no more than $q$ queries. We get the stated inequality as the sum of the advantages using the triangle inequality. □

# C Heuristic estimates of basic problem complexity

The security of the cipher suites used in the CRISP protocol is reduced to the single basic problem, namely, the indistinguishability of "Magma" from a random permutation.

Although it is practically impossible to give the exact upper bound for the advantage of the adversary in the PRP model for "Magma", the need to provide practical recommendations motivates us to estimate the value of $\mathrm{Adv}^{PRP}_{\mathsf{Magma}}(\mathcal{A})$ heuristically. The plausible approach is to narrow down the set of all possible algorithms $\mathcal{A}$ with resources $(t, q)$ to the set of currently known methods of constructive cryptanalysis. Methods that require more than $2^k$ operations for precomputations (for filling the initial memory of the algorithm $\mathcal{A}$) are excluded from the consideration.

The block cipher "Magma" is structurally identical to the GOST 28147-89 algorithm and, unlike its predecessor, has a fixed set of s-boxes, said to provide resistance against the differential [27] and linear [28] methods of the cryptanalysis (see also the design rationale of the 2-GOST cipher [32]).

In [31] a simple distinguisher using "symmetric fixed points" was proposed. For a random permutation $\Pi \in \mathrm{Perm}(V^n)$ the probability of the equality $\Pi(x||x) = x||x$ to hold true for arbitrary $x \in V^{n/2}$ is about $2^{-n}$, and for "Magma" it is twice as much. Hence, by checking $q \leq 2^{n/2}$ "symmetric points" the distinguishing advantage is about $\approx q \cdot (2 \cdot 2^{-n} - 2^{-n}) \approx q \cdot 2^{-n}$.

The distinguisher can be built using a key recovery algorithm. If the correct key was found, then the distinguisher's response is "1" (interaction with the cipher), otherwise, the result is "0" (interaction with a random permutation $\Pi$). For $q > \frac{k}{n} = \frac{256}{64} = 4$, the probability of a false answer after interacting with $\Pi$ is almost zero. So, in this case we can consider the distinguishing advantage and the probability of key recovery to be equal.

The success probability of the simple key guessing is about $t \cdot 2^{-k}$.

Two special methods of key recovery are described in [29, 30]. Both methods are arranged in a similar way. Consistently and independently of each other, $q$ known plaintext–ciphertext pairs are considered. For each pair with a probability of $2^{-p}$, a "rare event" will occur. The probability of at least one event among $q$ pairs is upper bounded by $q \cdot 2^{-p}$. For each pair, assuming that the event has occured, $2^c$ operations are performed and the same number of possible keys are constructed, each of which is checked on other pairs. If the event really happened, then the true key necessarily belongs to the set of tested ones.

The total number of keys constructed is $q \cdot 2^c$. The adversary can perform $t$ computational operations (we assume that one operation is enough to encrypt a block), the proportion of tested keys does not exceed $\frac{t}{q \cdot 2^c}$. The probability of recovering the true key can then be estimated as

$$\frac{q}{2^p} \cdot \frac{t}{q \cdot 2^c} = \frac{t}{2^{p+c}}.$$

However, the probability of success cannot be greater than the probability of a rare event ($q \cdot 2^{-p}$). Therefore, we obtain the upper bound as

$$\min\left(\frac{q}{2^p}, \frac{t}{2^{p+c}}\right).$$

Isobe [29] uses the so-called "reflection property" to mount an attack. The probability of "rare event" is $2^{-p} = 2^{-\frac{n}{2}} = 2^{-32}$. For each pair plaintext-ciphertext, $2^c = 2^{192}$ keys are constructed.

In the attack [30] proposed by Dinur, Dunkelman, Shamir, the "fixed point" is used, $2^{-p} = 2^{-n} = 2^{-64}$, $2^c = 2^{128}$.

Thus, the general from of the heuristic estimation is

$$\mathrm{Adv}^{PRP}_{\mathsf{Magma}}(t,q) \lessapprox \max_{t_1+t_2+t_3=t}\left(\frac{t_1}{2^{256}}, \min\left(\frac{q}{2^{32}}, \frac{t_2}{2^{224}}\right), \min\left(\frac{q}{2^{64}}, \frac{t_3}{2^{192}}\right)\right) + \min\left(2^{-32}, \frac{q}{2^{64}}\right).$$

Simplify for $t \ll 2^{192}$ and arbitrary $q < 2^{32}$

$$\mathrm{Adv}^{PRP}_{\mathsf{Magma}}(t,q) \lessapprox \frac{t}{2^{192}} + \frac{q}{2^{64}}.$$

Therefore, the distinguishing advantage can be considered equal to *zero* for most purposes.

Similarly, other methods of cryptanalysis of the "Magma" cipher can be taken into account in the heuristic estimates.

# On the security of one RFID authentication protocol

Anastasiia Chichaeva, Stepan Davydov, Ekaterina Griboedova, and Kirill Tsaregorodtsev

JSC "NPK Kryptonite", Moscow, Russia
{a.chichaeva, s.davydov, e.griboedova, k.tsaregorodtsev}@kryptonite.ru

**Abstract**

The main result of the paper is the formalization of the authentication property as an exact model in the provable security framework and security proof for the RFID authentication protocol developed in TC 26.

**Keywords:** authentication, provable security, RFID

## 1 Introduction

Radio Frequency Identification (RFID) systems have become popular for their powerful capability as wireless identification. Security and privacy threats also evolve with the progress of RFID technology ([1, 2, 3]). At the same time, the standardization process is far from complete: specific authentication protocols that differ from those standardized worldwide (e.g., in ISO [4, 5]) arise in practice; most of RFID tags simply lack any of the cryptographic mechanisms. Hence, many RFID systems do not satisfy all the modern security requirements, such as Tag and/or Reader provably secure (see [6, 7]) authentication; privacy of the parties; confidentiality and integrity of additional transmitted data.

The scientific research in the field of RFID authentication security is inadequately represented in the Russian-language specialized literature, and there are no standardized solutions based on Russian cryptographic algorithms; hence, the development of such protocols is an extremely urgent task. At the moment, the working group TC 26 is developing standards of RFID authentication protocol. The balance between technical characteristics and security is always critical in such low-resource systems as RFID (see [8]); thus, it seems appropriate to develop several cryptographic solutions for different price segments and applications.

In the modern literature the authentication property is usually defined in the context of pre-shared key derivation, i.e. as the property of AKE-protocol (authenticated key exchange). Adversarial models for AKE-protocols are well studied and take into account various practical attacks (see e.g. [9, 10, 11]). The protocol considered in this article does not imply key agreement (only authentication one or both parties). The result of the AKE-protocol run is a fresh session key (in general, without any guarantee that the other party has generated the same key), so the models developed for such protocols are inapplicable in case of studying authentication. The closest to the definition of authentication property considered in this article is the definition from [12]. The difference lies in the exact formalization in terms of oracle indistinguishability. The approach adopted in this paper is convenient for combining with other security properties (formulated in terms of indistinguishability, e.g. for confidentiality or privacy) and also has a more general interpretation of partnered (paired) sessions.

In this paper we identify security requirements and technical features of RFID tags (Section 3), describe the "light" authentication protocol (that provides a minimal set of security properties), intended for cheap low-resource tags (Section 4), propose the adversarial model for authentication protocols with optional additional data transfer (Section 5) and give the sketch of the analysis of protocol security in the proposed model (Section 6).

## 2 Notation

Encryption (decryption) under the key $k$ is denoted as $E_k$ ($E_k^{-1}$ resp.); $CBC_k^{IV}(x)$ ($\widehat{CBC}_k^{IV}(x)$) is the encryption of the plaintext $x$ using CBC mode [13] with the initialization vector $IV$ and the key $k$; the $IV$ is added to the beginning of the resulting ciphertext (in $\widehat{CBC}$ each application of a block cipher $E_k(\cdot)$ is replaced by the inverse permutation $E_k^{-1}(\cdot)$); $MAC_k(x)$ is a message authentication code for $x$ under the key $k$. By $r \xleftarrow{\$} M$ we denote a uniformly random element $r$ sampled from a finite set $M$. $Consts$ is a set of predefined 4-bit constants, which specify authentication and data protection modes, as well as the role of the participant (Tag, Reader). String concatenation is denoted as $\|$; $x \leftarrow y$ means "assing value $y$ to $x$".

# 3 Cryptographic and technical requirements for RFID system

## 3.1 General description

The RFID system consists of one dedicated Reader and several other parties (Tags). We assume that the system works in the presence of an active adversary, which can intercept and modify all messages passing between honest parties. In this paper we consider a class of protocols based on symmetric cryptography (see also Section 3.2), in which parties authenticate using the pre-shared secret key(s) (see the standards [4, 5]). The mechanism of its generation and distribution to the Tag's protected memory is beyond the scope of this work and should be studied separately. From now on we assume that the triple of uniformly random independent keys $(k, k^e, k^m)$ is distributed between each Tag and the Reader. Keys $k^e$ and $k^m$ are used to protect additional data (in practice it is usually convenient to transfer securely some additional data (payments, identifiers) in conjunction with the authentication procedure (as it is implemented in [4])); key $k$ is the pre-shared key intended for authentication purpose.

## 3.2 Technical requirements

Features of RFID systems impose some restrictions on the implementation of cryptographic mechanisms. The following technical key requirements were listed in [8].

1. It is advisable to implement cryptographic protection mechanisms on RFID tags that operate in the near field; this condition imposes physical restrictions on the reading range and makes the possibility of relay-attacks less critical.

2. It should be possible to implement simplest RFID protocols on passive tags without autonomous power sources (more complex mechanisms can be implemented on active tags).

3. Tags shall have a protected WORM (write once, read many) memory to store shared secret keys.

4. The implementation should have a relatively small gate area; in particular, symmetric-cryptography based protocols are preferable.

**Remark 1.** *As a consequence of the last item, we propose protocols that do not use $E_k^{-1}$ operation on the Tag side.*

### 3.3 Cryptographic requirements

The following security requirements are addressed in the paper.

1. **Authentication:** upon successful completion of the protocol, the verifier must be sure that the interaction took place with a legitimate prover (a participant with a shared secret key). Also, the verifier must know exactly which party was authenticated (e.g., unique ID of the party).
2. **Confidentiality:** in a data processing mode that ensures confidentiality an adversary does not receive any information about the protected data (except for its length).
3. **Integrity:** in a data processing mode that ensures integrity an adversary cannot correctly modify (replace, forge) the data.

Note that there exist a number of additional relevant security properties (see [8]), which are not addressed in this paper due to the implementation constraints.

1. **User privacy:** in the protocol under consideration the authentication requires the negotiation of the shared secret key to be used, which is achieved by the tag broadcasting its $ID$ into the channel in an insecure manner. It allows adversary to trace Tags (construction and analysis of protocols that ensure user privacy are described in e.g. [14, 15, 16]).
2. **Relay attack:** a variation of the "man-in-the-middle" attack: an adversary intercepts all the responses of one of the participants and sends them unchanged on its own behalf; the protocol under consideration does not allow to prevent this class of attacks (see survey [17]).
3. **System availability violation (DoS):** this problem cannot be solved by cryptographic means only (see [3] for an overview of possible solutions).

## 4 Protocol description

In this section we describe the RFID authentication protocol (based on 64-bit block cipher, e.g., "Magma" [18]) that provides Tag Authentication (TAM-mode) with optional Reader Authentication (MAM-mode) and optional secure additional data transmission (specified by $ProtMode$ parameter). Due to the space constraints, the description of IAM-mode (Reader-only authentication) is omitted (see [19] for more details). The parameter

$ProtMode$ (see Table 1) specifies the data protection mode (integrity protection and optional encryption of the additional data).

Table 1: Supported data protection modes

| $Prot\ Mode$ | $Protect(Resp, Data)$ | $\widehat{Protect}(Resp, Data)$ |
|---|---|---|
| 00 | $Resp$ | |
| 10 | $Resp \parallel Data \parallel MAC_{k^m}(Resp \parallel Data)$ | |
| 11 | $IV \xleftarrow{\$} \{0,1\}^{64}$<br>$Resp \parallel CBC^{IV}_{k^e}(Data) \parallel$<br>$MAC_{k^m}(Resp \parallel CBC^{IV}_{k^e}(Data))$ | $IV \xleftarrow{\$} \{0,1\}^{64}$<br>$Resp \parallel \widehat{CBC}^{IV}_{k^e}(Data) \parallel$<br>$MAC_{k^m}(Resp \parallel \widehat{CBC}^{IV}_{k^e}(Data))$ |

Table 2: Constant specification for TAM and MAM modes

| $ProtMode$ | TAM | MAM, Tag | MAM, Reader |
|---|---|---|---|
| $ProtMode = 00$<br>no add. data | 0x0 | 0x6 | 0x9 |
| $ProtMode = 10$<br>data integrity | 0x1 | 0x7 | 0xa |
| $ProtMode = 11$<br>auth. encryption | 0x2 | 0x8 | 0xb |

We assume that each Tag and the Reader have a triple of independent uniformly random shared secret keys $(k, k^e, k^m)$. In general, more than one tuple of keys can be shared between the Tag and the Reader, but without loss of generality we can consider only the case of one pre-shared key tuple on each Tag. The protocol runs as follows (see Figure 1).

1. The Tag broadcasts its identifier $ID$.
2. The Reader initiates the authentication procedure. It sends an authentication request to the Tag, which contains a $Rlen$-bit random number $R$ (challenge) and authentication parameters $params$, which includes authentication type (TAM, IAM, MAM), data protection mode $ProtMode$, the size of the additional data (if any) and other technical parameters (see [19]).

3. The Tag chooses an appropriate constant $C_1 \in Consts$ (see Table 2), concatenates it with the challenge $R$, encrypts $C_1 \parallel R$ under the key $k$ obtaining the result $TResp$. In case of MAM-mode the Tag also generates a second random challenge $r$ of a bit length $Rlen$ and sets $Resp \leftarrow TResp \parallel r$, otherwise $Resp \leftarrow TResp$. Tag sends $Resp$ to the Reader (with an optional additional data in the format specified in Table 1).

4. The Reader first checks the MAC tag (if any) under the key $k^m$ specific for the Tag with identifier $ID$. Then it parses $Resp$ and decrypts $TResp$ under the key $k$, verifies $C_1$ and $R$. If all values are correct, then the Tag authentication is successful, and the additional data (if any) can be accepted (and decrypted under key $k^e$, if needed). *Note. The protocol ends at this stage if there is no Reader authentication step. The following steps of the protocol are performed only if the Reader has to be authenticated (i.e., optional).*

5. The Reader chooses an appropriate constant $C_2 \in Consts$ (see Table 2), concatenates it with the challenge $r$, encrypts $C_2 \parallel r$ under the key $k$ obtaining the result $IResp$ and sends it to the Tag (with an optional additional data in the format specified in Table 1).

6. The Tag first checks the MAC value (if any), then it calculates the value $IResp' = E_k(C_2 \parallel r)$ and compares it with $IResp$ (note that it is not needed to implement decryption procedure at the Tag side, see Remark 1). If all values are correct, then the Reader authentication is successful, and the additional data (if any) can be accepted (and decrypted in $\widehat{CBC}$-mode, if needed).

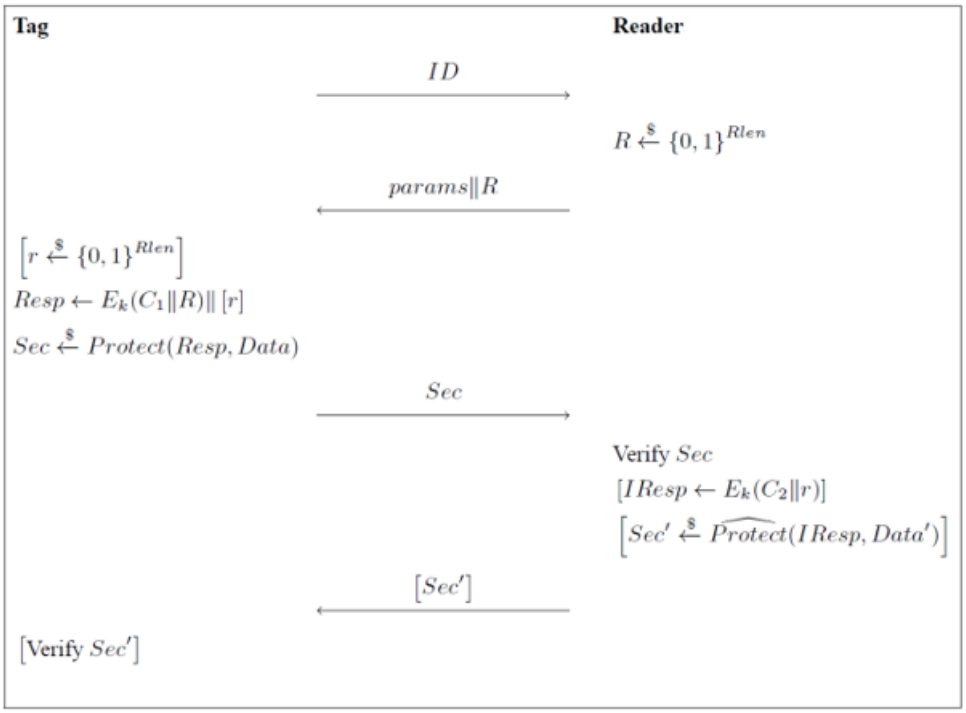



Figure 1: Authentication protocol, optional steps marked as [...]

Constant value used in protocol (i.e., $C_1$ and $C_2$) are unique for authentication types (TAM, MAM), data protection modes $ProtMode$ (plain, integrity, AE-mode) and holders (Tag, Reader). The length of random challenges $Rlen = 60$ bits for 64-bit block cipher (i.e. concatenating randomness with the 4-bit constant gives 64 bits in total).

# 5 Adversarial model

## 5.1 Formal definition of authentication protocol

**Definition 1.** *An authentication protocol with optional data transfer is a triple of probabilistic algorithms* $\Pi = (InitReader, InitTag, Auth)$ *with the following properties:*

1. $InitReader()$ *is the Reader initialization algorithm (unique participant); it takes no input and returns Reader initial state.*

2. $InitTag(ID, Reader)$ *is a Tag initialization algorithm; it takes as input a unique Tag* $ID$ *and current state of the Reader and returns initial state of the Tag* $state_{ID}$ *(which contains secret keys* $k_{ID}$*) and updated state of the Reader.*

3. $Auth(state_A, m)$ *is an authentication algorithm; it takes as input participant's* $state_A$ *and a message* $m$ *to be processed and returns an updated*

*state* $state'_A$ *and response* $m'$.

For the protocol in question we have the following: $InitReader()$ generates empty database for the Reader, $InitTag$ generates triple $k_{ID} = (k, k^e, k^m)$ (pre-shared keys between the Tag $ID$ and the Reader). Keys are not changed during the protocol and recorded in the Tag state and Reader database.

We assume that the type of authentication (unilateral or mutual) and the additional data to be transmitted is stored in the state of the participant.

### 5.2 Adversarial model for authentication protocol

In this section we describe an Experiment which formalizes the notion of secure RFID-system in the "provable security" framework [6, 7, 9, 10], i.e. we have to specify concrete oracles (interfaces) and determine the success measure (advantage) of an adversary (some probabilistic algorithm). Then we give an explicit estimate of advantage in terms of adversarial running time and number of queries to different oracles.

The Experiment maintains a set of consistent dictionaries (tables):

1. The table of Tags $Tags[ID]$ consisting of records with the following fields:
   - Tag state $Tags[ID].state$;
   - current session number $Tags[ID].current\_session$.

   This table fills in and changes whenever the adversary makes queries to the $CreateTag$, $Send$ or $StartTagSession$ oracles.
2. The table of sessions $Sessions[\pi]$ consisting of records with the following fields:
   - session holder $Sessions[\pi].holder$ (i.e. either some Tag $ID$ or the Reader);
   - expected partner in the session $Sessions[\pi].partner$;
   - result of the session run $Sessions[\pi].result$ (i.e. in progress, accept, error code);
   - session identifier $Sessions[\pi].sid$, which is used to formalize the notion of partnered sessions (see Definition 3 below).

The Experiment also maintains *Reader* data structure in accordance with the protocol (i.e., updates the state $Reader.state[\pi]$ in all sessions $\pi$ and Reader database of keys $Reader.database$).

**Definition 2.** *The advantage of the adversary $\mathcal{A}$ in the* $\mathrm{AUTH}^+$*-model for authentication protocol* $\Pi = (InitReader, InitTag, Auth)$ *is defined as:*

$$\mathsf{Adv}_{\Pi}^{\mathrm{auth}^+}(\mathcal{A}) = \mathbb{P}\left[\mathrm{Exp}_{\Pi}^{\mathrm{AUTH}^+-1}(\mathcal{A}) \to 1\right] - \mathbb{P}\left[\mathrm{Exp}_{\Pi}^{\mathrm{AUTH}^+-0}(\mathcal{A}) \to 1\right],$$

*where* $\mathrm{Exp}^{\mathrm{AUTH}^+-b}$, $b \in \{0,1\}$ *is of the following form:*

$\mathrm{Exp}_{\Pi}^{\mathrm{AUTH}^+-b}(\mathcal{A})$

$\pi \leftarrow 0$
$Tags \leftarrow [\,]$
$Sessions \leftarrow [\,]$
$Reader \leftarrow InitReader()$
$b' \xleftarrow{\$} \mathcal{A}^{\mathcal{O},SetMessage^b,Test^b}()$
**return** $b'$

*By* $\mathcal{O}$ *we denote a set of oracles CreateTag, StartReaderSession, StartTagSession, Send, Result. All of the oracles are defined in the following pseudocode.*

$CreateTag(ID)$

$Tags[ID].state \xleftarrow{\$} \Pi.InitTag(ID, Reader.params)$
$Tags[ID].current\_session = 0$
$Reader.database[ID] = Tags[ID].state$
**return** *success*

$StartReaderSession(mode)$

$\pi++$
$Sessions[\pi].holder = Reader$
$Sessions[\pi].partner = \perp$
$Sessions[\pi].result = in\text{-}progress$
$Sessions[\pi].sid = \perp$
$Reader.state[\pi].mode = mode$
$Reader.state[\pi].message = \varepsilon$
$Reader.state[\pi].params = Reader.params$
**return** $\pi$

$StartTagSession(ID)$

$\pi++$
$Sessions[\pi].holder = ID$
$Sessions[\pi].partner = Reader$
$Sessions[\pi].result = in\text{-}progress$
$Sessions[\pi].sid = \perp$
$Tags[ID].current\_session = \pi$
$Tags[ID].state.mode = \varepsilon$
$Tags[ID].state.message = \varepsilon$
$Tags[ID].state.params = \varepsilon$
**return** $\pi$

$Result(\pi)$

**return** $Sessions[\pi].result$

$SetMessage^b(\pi, M_0, M_1)$

$holder = Sessions[\pi].holder$
**if** $(holder = Reader)$
  $state = Reader.state[\pi]$
**else**
  $state = Tags[holder].state$
**fi**
**if** $(M_0 \neq M_1)$AND$(state.mode \neq AE)$
  **return** $\perp$
**fi**
$state.message = M_b$
**return** *success*

$Send(\pi, m)$

$holder = Sessions[\pi].holder$
**if** $holder = Reader$
  $state \leftarrow (Reader.state[\pi], Reader.database)$
**else**
  **if** $\pi \neq Tags[holder].current\_session$
    **return** $\perp$
  **fi**
  $state \leftarrow Tags[holder].state$
**fi**
$(state, m') \xleftarrow{\$} \Pi.Auth(state, m)$
$UpdateSession(Sessions[\pi], m, state, m')$
**return** $m'$

$Test^b(\pi)$

**if** $(b = 0)$ **then**
  **return** 0
**else**
  $t_1 \leftarrow Correctness(\pi)$
  $t_2 \leftarrow$ NOT$(Match(\pi, Sessions))$
  **return** $(t_1 \& t_2)$
**fi**

**Definition 3.** *The session identifier sid consists of the following fields:*

- *authentication type* $AUTH\_TYPE \in \{\mathrm{TAM}, \mathrm{MAM}\}$;
- *data protection type* $ProtMode \in \{PLAIN, MAC, AE\}$;

– *random challenges* $R$ *or* $r \parallel R$ *(depending on the authentication type:* $R$ *for* TAM *and* $r \parallel R$ *for* MAM*);*

– $Protect(Resp, Data)$ *and* $\widehat{Protect}(IResp, Data)$ *(in case of* $ProtMode \in \{MAC, AE\}$*);*

**Definition 4.** *The predicate* $Match(\pi, Sessions)$ *is true if and only if there exists* $\pi'$ *such that following set of rules is fullfiled:*

– $Sessions[\pi].holder = Sessions[\pi'].partner$*,* $Sessions[\pi].partner = Sessions[\pi'].holder$*;*

– *if* $Sessions[\pi].sid.AUTH\_TYPE \neq$ MAM*, then* $Sessions[\pi].sid = Sessions[\pi'].sid$*;*

– *if* $Sessions[\pi].sid.AUTH\_TYPE =$ MAM*,* $Sessions[\pi].sid.ProtMode \neq PLAIN$ *and* $Sessions[\pi].holder = ID$*, then* $Sessions[\pi].sid = Sessions[\pi'].sid$*;*

– *if* $Sessions[\pi].sid.AUTH\_TYPE =$ MAM*,* $Sessions[\pi].sid.ProtMode \neq PLAIN$ *and* $Sessions[\pi].holder = Reader$*, then only the equality of the fields* $AUTH\_TYPE$*,* $ProtMode$*,* $R$*,* $r$ *and* $Protect(Resp, Data)$ *in* $Sessions[\pi].sid$ *and* $Sessions[\pi'].sid$ *is checked;*

– *if* $Sessions[\pi].sid.AUTH\_TYPE =$ MAM*,* $Sessions[\pi].sid.ProtMode = PLAIN$ *and* $Sessions[\pi].holder = ID$*, then only the equality of the fields* $AUTH\_TYPE$*,* $ProtMode$*,* $r$ *in* $Sessions[\pi].sid$ *and* $Sessions[\pi'].sid$ *is checked;*

– *if* $Sessions[\pi].sid.AUTH\_TYPE =$ MAM*,* $Sessions[\pi].sid.ProtMode = PLAIN$ *and* $Sessions[\pi].holder = Reader$*, then only the equality of the fields* $AUTH\_TYPE$*,* $ProtMode$*,* $R$ *in* $Sessions[\pi].sid$ *and* $Sessions[\pi'].sid$ *is checked;*

**Remark 2.** *Match predicate binds two sessions (from the Tag and the Reader "points of view") in one object. It formalizes the following logic: if the authentication finished successfully, then the legitimate partner was "alive", i.e. responded properly to the* ***holder's challenge*** *(or the collision of challenges happened). Moreover, in case of additional data transfer, Tag's MAC value binds* $r$*,* $R$ *and* $Data$*, Reader's MAC value binds* $r$ *and* $Data'$*, hence,* $Data$ *and* $Data'$ *are implicitly binded, i.e. if the Reader authentication is correct on the Tag's side, then it is guaranteed that* $Data'$ *is an answer not only to the Tag's challenge* $r$*, but also to the* $Data$ *message.*

**Remark 3.** *The type* $AUTH\_TYPE = \text{MAM}$ *is a special case due to the following facts.*

- *The adversary is always possible to abort the last message delivery; in this case the session on the Reader side is already finished, whereas on the Tag side it is incomplete; we do not consider this situation as security violation.*
- *The value* $r$ *is loosely connected to the first messages of* MAM*-type session (in case of no MAC calculation, i.e.* $ProtMode = PLAIN$*); for instance, one can bind it to the particular session via more elaborate calculation of* $TResp$ *(e.g. with the help of OMAC/HMAC functions); but this countermeasure complicated the logic and computations of the protocol giving not so much extra security (in any case the session with the wrong randomness will be rejected by the holder). Hence, in case of no additional data we also do not consider replacement of Tag's randomness* $r$ *from the Reader "point of view", as well as Reader's randomness* $R$ *from the Tag "point of view" as a security violation (i.e., MAM-type session without additional data is equivalent to two independent authentication sessions: TAM and IAM).*

**Definition 5.** *The predicate* $Correctness(\pi)$ *filters out trivial attacks in the model and is defined as follows:* $Sessions[\pi].result = accept$ *and if* $Sessions[\pi].sid.AUTH\_TYPE = \text{TAM}$*, then* $Sessions[\pi].holder = Reader$*.*

As we can see from Definitions 3 and 4, the following situations will be considered in the AUTH$^+$-model as a security violation:

- undetectable replacement of authentication type $AUTH\_TYPE$;
- undetectable replacement of data protection mode $ProtMode$;
- replacement of the random challenge $r$ or $R$ undetectable for the challenger (i.e. $r$ for Tag, $R$ for Reader, see also Remark 2);
- undetectable modification of transmitted data $Data$, if any (see also Remark 2).

The function $UpdateSession$ acts as follows:

- updates $sid$ according to the received protocol messages;
- updates $result$ in case of error or successful completion of the protocol;

– updates the expected partner in the session;

**Definition 6.** *By MAC-session (AE-session) we denote a session $\pi$ with a $ProtMode = MAC$ ($ProtMode = AE$ respectively).*

**Definition 7.** *Let* $\mathtt{Adv}_{\Pi}^{\mathrm{AUTH}^+}(t, d, \mathcal{P}, \mathcal{Q}, \mathcal{R}, \Theta, \mathcal{M}, \mathcal{N}, \Phi, \Psi, \widehat{\mathcal{Q}}, \widehat{\mathcal{M}}, \widehat{\Phi})$ *be the maximal advantage* $\mathsf{Adv}_{\Pi}^{\mathrm{auth}^+}(\mathcal{A})$, *over all adversaries $\mathcal{A}$ whose running time does not exceed $t$ and with the following restriction on oracle queries:*

– *number of $CreateTag$ queries does not exceed $d$;*

– *number of sessions $\pi$ for which holder or partner is $ID_i$ does not exceed $p_i = \mathcal{P}[i]$;*

– *number of AE-sessions with holder $ID_i$ (with holder Reader and partner $ID_i$) does not exceed $q_i = \mathcal{Q}[i]$ ($\widehat{q_i} = \widehat{\mathcal{Q}}[i]$ resp.);*

– *number of MAC-sessions between $ID_i$ and Reader (i.e. where $ID_i$ is a holder or a partner) does not exceed $r_i = \mathcal{R}[i]$;*

– *number of queries $Test^b(\pi)$, where $ID_i$ is either a holder or a partner does not exceed $\theta_i = \Theta[i]$;*

– *maximal block length of the data in AE-sessions with holder $ID_i$ (with holder Reader and partner $ID_i$) does not exceed $\mu_i = \mathcal{M}[i]$ ($\widehat{\mu}_i = \widehat{\mathcal{M}}[i]$ resp.);*

– *maximal block length of the data in MAC-sessions with holder or partner $ID_i$ does not exceed $\nu_i = \mathcal{N}[i]$;*

– *total block length of the data in AE-sessions with holder $ID_i$ (with holder Reader and partner $ID_i$) does not exceed $\phi_i = \Phi[i]$ ($\widehat{\phi}_i = \widehat{\Phi}[i]$ resp.);*

– *total block length of the data in MAC-sessions with holder or partner $ID_i$ does not exceed $\psi_i = \Psi[i]$.*

### 5.3 The relevance of the model

The adversarial model precisely describes the adversary capabilities to interact with the system and strictly defines what a successful attack is. In the model under consideration the adversary has the following opportunities:

– create legitimate tags using $CreateTag$ queries;

– start sessions of chosen type using $StartReaderSession$ or $StartTagSession$ queries;

– set additional data to be authenticated and/or encrypted using $SetMessage^b$ query; the bit $b$ controls which of the data messages $(M_0, M_1)$ will be processed; in case of AE-sessions it is possible that $M_0 \neq M_1$ (this oracle formalizes the inability of adversary to break the confidentiality of the transmitted data);

– check the result of the session using *Result* query;

– send messages to protocol participants using *Send* query; messages are transmitted within some fixed session $\pi$ to the session holder (the Reader is able to participate in parallel sessions, but for Tags all sessions are strictly sequential, and the adversary is able to send message only in the current session for the Tag);

– test sessions using $Test^b$ query.

Last oracle formalizes three possible security issues: the property of secure participant authentication, data integrity within the session, integrity at the session level. If the adversary is able to authenticate without the help of Tag (or Reader), or to forge MAC-value, then it is possible to construct a session $\pi$ for which there would be no matched session $\pi'$. When such session $\pi$ is constructed, it can be tested using $Test^b(\pi)$ query. The answer will help the adversary to guess the value $b$ correctly.

**Remark 4.** *There are some differences from classical RFID security models (i.e., [14, 16]): a number of oracles were not included in the discussed model.*

1. *Oracles $Draw^b$, $Free$: RFID protocol in question does not give any guarantee on the Tag privacy (the $ID$ is transmitted in the cleartext).*

2. *Oracle $CorruptLTK$: forward secrecy and forward privacy is not guaranteed (the adversary can trivially break the confidentiality of old messages using corrupted long-term key).*

3. *Oracle $CorruptState$: it gives the opportunity to break the confidentiality of the protocol — the adversary simply writes message $M_b$ to the memory, and then check the inner state of the Tag.*

We emphasize a few technical points implicitly assumed in the model:

– it is expected that $ID$ queries for the $CreateTag$ oracle are unique; if the same $ID$ is used twice, then $CreateTag$ returns an error $\perp$;

– it is assumed that *Send* oracle accepts only previously started sessions $\pi$, otherwise it throws an error; also the session $\pi$ must be unfinished (otherwise the error is thrown, or empty message is returned);

– it is assumed that $Test^b$ accepts only previously started sessions;

These requirements do not narrow down the set of adversaries under consideration. Any "useless" query can be eliminated without reducing the probability of successful attack.

## 6 Estimating the adversarial advantage

In this section we give a sketch of a proof for the main theorem on the security of the proposed RFID protocol $\Pi$ in the AUTH$^+$-model. The full proof of the theorem is contained in Appendix B and C.

**Theorem 1.** *The following inequality holds:*

$$\mathtt{Adv}_{\Pi}^{\mathrm{AUTH}^+}(t, d, \mathcal{P}, \mathcal{Q}, \mathcal{R}, \Theta, \mathcal{M}, \mathcal{N}, \Phi, \Psi, \widehat{\mathcal{Q}}, \widehat{\mathcal{M}}, \widehat{\Phi}) \leq$$

$$\leq \sum_{i=1}^{d} \mathtt{Adv}^{LOR2}\Big(t + T, q_i, \mu_i, \phi_i, \widehat{q}_i, \widehat{\mu}_i, \widehat{\phi}_i\Big) +$$

$$+2 \cdot \sum_{i=1}^{d} \mathtt{Adv}^{\mathrm{EUF-CMA}}\Big(t + T, r_i + q_i + \widehat{q}_i, \max\left(\mu_i + 2, \widehat{\mu}_i + 2, \nu_i + 1\right),$$

$$\phi_i + 2 \cdot q_i + \widehat{\phi}_i + 2 \cdot \widehat{q}_i + \psi_i + r_i, \theta_i\Big) +$$

$$+2 \cdot \sum_{i=1}^{d} \Big(\mathtt{Adv}^{\mathrm{PRP}}(t + T, p_i + \theta_i) + \frac{\theta_i \cdot (p_i + \theta_i)}{2^{Rlen}} + \frac{\theta_i |Consts|}{|\mathrm{Dom}| - p_i - \theta_i + 1}\Big).$$

Main steps (reductions) of the proof are schematically represented in Fig. 2. The reduction from $LOR2$ to $PRP$ holds for $(CBC, \widehat{CBC})$ pair of encryption modes with a randomly chosen $IV$, the reduction from $EUF\text{-}CMA$ to $PRP$ holds for CMAC.

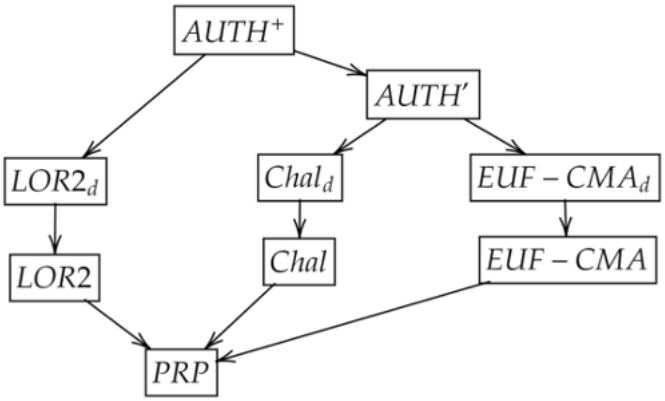



Figure 2: Schematic representation of the proof

The first step of the proof is to break the AUTH$^+$-model into two disjoint models. It is shown that the insecurity of $\Pi$ in the AUTH$^+$ can be bounded from above by two terms:

– the insecurity of $\Pi$ in the $LOR2_d$ (see section B.3.2) — the model for estimating the hardness of breaking confidentiality for $d$ participants;

– the insecurity of $\Pi$ in the AUTH$'$ (see section B.6) — the model for estimating the hardness of breaking authenticity and/or integrity for $d$ participants.

The insecurity in AUTH$'$-model can be estimated in terms of insecurities in $Chal_d$-model (see Section B.5.3, the model for authentication without additional data transmission) and $\mathrm{EUF}-\mathrm{CMA}_d$-model (see Section B.4.2, the model for data integrity for $d$ independent keys).

Each model for $d$ independent participants can be reduced to the case of a single participant (see Sections B.3.1, B.3.2, B.4.1, B.4.2). One-participant models are reduced to the insecurity of the block cipher in use in the PRP-model (Section B.1) and probabilistic estimates.

**Corollary 1.** *Let the following assumptions be true:*

– *The best known estimate for "Magma" block cipher insecurity in* sPRP*-model is* $\frac{q\cdot t}{2^{256}}$ *(see [20, 21]);*

– *the pair of modes* $(CBC, \widehat{CBC})$ *with random IV is used (see [22]);*

– *constant length is 4 bits, challenge length is 60 bits.*

*Let* $p = \max_i p_i$, $\phi = \left(\max_i \phi_i + \max_i \widehat{\phi}_i + \max_i \psi_i\right)$, $\mu = \max_i (\mu_i, \widehat{\mu}_i, \nu_i)$, *then it holds that:*

$$\mathtt{Adv}_{\Pi}^{\mathrm{AUTH}^+}(t, d, \mathcal{P}, \mathcal{Q}, \mathcal{R}, \Theta, \mathcal{M}, \mathcal{N}, \Phi, \Psi, \widehat{\mathcal{Q}}, \widehat{\mathcal{M}}, \widehat{\Phi}) \leq$$

$$\leq \frac{6d\phi^2}{2^{64}-\phi} + dp \cdot \mathtt{Adv}^{\mathrm{EUF-CMA}}(3p, \mu+2, 3\phi+5p) + \\ + \left(\frac{dp^2}{2^{58}} + \frac{9dp}{2^{63}-p}\right) + \frac{d\cdot(t+T+2\phi)\cdot(p+\phi)}{2^{255}}.$$

Using the results from [23, 24, 25, 26], this estimate can be further simplified.

# References


[1] I. Damgård, M. Ø. Pedersen, "RFID security: Tradeoffs between security and efficiency", Cryptographers Track at the RSA Conference, 2008, 318–332.

[2] T. van Deursen, "50 ways to break RFID privacy", IFIP PrimeLife International Summer School on Privacy and Identity Management for Life, 2010, 192–205.

[3] A. Mitrokotsa, M. Beye, P. Peris-Lopez, "Classification of RFID Threats based on Security Principles", 2010.

[4] ISO, *ISO/IEC 29167-10 Information technology — Automatic identification and data capture techniques — Part 10: Crypto suite AES-128 security services for air interface communications*, 2017.

[5] ISO, *ISO/IEC 29167-21 Information technology — Automatic identification and data capture techniques — Part 21: Crypto suite SIMON security services for air interface communications*, 2018.

[6] F. Guo, W. Susilo, Y. Mu, *Introduction to security reduction*, Springer, Cham, Switzerland, 2018, 253 pp.

[7] J. Katz, Y. Lindell, *Introduction to modern cryptography*, CRC press, Boca Raton, Florida, 2020, 626 pp.

[8] V. Belsky, E. Griboedova, K. Tsaregorodtsev, A. Chichaeva, "Security of RFID systems", *International Journal of Open Information Technologies*, 2021, In Russian.

[9] C. Boyd, A. Mathuria, D. Stebila, *Protocols for authentication and key establishment*, Springer, Berlin, Heidelberg, 2020.

[10] K.-K. R. Choo, *Secure key establishment*, **41**, Springer Science & Business Media, 2008, 216 pp.

[11] B. Dowling, M. Fischlin, F. Günther, D. Stebila, "A cryptographic analysis of the TLS 1.3 handshake protocol", *Journal of Cryptology*, **34**:4 (2021), 1–69

[12] M. Bellare, P. Rogaway, "Entity authentication and key distribution", Annual International Cryptology Conference, 1993, 232–249

[13] *GOST 34.13-2018. Information technology. Cryptographic data security. Modes of operation for block ciphers*, 2018, In Russian.

[14] S. Vaudenay, "On privacy models for RFID", International conference on the theory and application of cryptology and information security, 2007, 68–87.

[15] R. H. Deng, Y. Li, M. Yung, Y. Zhao, "A zero-knowledge based framework for RFID privacy", *Journal of Computer Security*, **19**:6 (2011), 1109–1146.

[16] J. Hermans, A. Pashalidis, F. Vercauteren, B. Preneel, "A new RFID privacy model", European symposium on research in computer security, 2011, 568–587.

[17] G. Avoine, M. Bingöl, I. Boureanu, S. Čapkun, G. Hancke, S. Kardaş, C. H. Kim, C. Lauradoux, B. Martin, J. Munilla, A. Peinando, K. B. Rasmussen, D. Singelee, A. Tchamkerten, R. Trujillo-Rasua, S. Vaudenay, "Security of distance-bounding: A survey", *ACM Computing Surveys (CSUR)*, **51**:5 (2018), 1–33.

[18] *GOST 34.12-2018. Information technology. Cryptographic data security. Block ciphers*, 2018, In Russian.

[19] *Information technology. Cryptographic data security. Security services for RFID air interfaces. Part 2. The use of cryptographic mechanisms to ensure the security of RFID air interfaces (project)*, 2022, In Russian.

[20] T. Isobe, "A single-key attack on the full GOST block cipher", International Workshop on Fast Software Encryption, 2011, 290–305.

[21] I. Dinur, O. Dunkelman, A. Shamir, "Improved attacks on full GOST", International Workshop on Fast Software Encryption, 2012, 9–28.

[22] K. Tsaregorodtsev, "An analysis of modes of encryption for RFID devices", *Prikladnaya Diskretnaya Matematika*, **13** (2020), In Russian.

[23] T. Iwata, K. Kurosawa, "Stronger security bounds for OMAC, TMAC, and XCBC", International Conference on Cryptology in India, 2003, 402–415.

[24] M. Bellare, O. Goldreich, A. Mityagin, *The Power of Verification Queries in Message Authentication and Authenticated Encryption*, Cryptology ePrint Archive, Report 2004/309, 2004 `eprint.iacr.org/2004/309.pdf`.

[25] M. Nandi, "Improved security analysis for OMAC as a pseudorandom function", *Journal of Mathematical Cryptology*, **3**:2 (2009), 133–148.

[26] S. Chattopadhyay, A. Jha, M. Nandi, "Towards tight security bounds for OMAC, XCBC and TMAC", Advances in Cryptology–ASIACRYPT 2022: 28th International Conference on the Theory and Application of Cryptology and Information Security, 2023, 348–378.

[27] P. Rogaway, "Evaluation of Some Blockcipher Modes of Operation", *Cryptography Research and Evaluation Committees (CRYPTREC)*, 2011 `www.cs.ucdavis.edu/ rogaway/papers/modes.pdf`.

[28] S. Chatterjee, A. Menezes, P. Sarkar, "Another look at tightness", Selected Areas in Cryptography: 18th International Workshop, SAC 2011, 2012, 293–319.

# A Associated ISO standards

The following standards were analyzed before designing the protocol in question [4, 5], and the next vulnerabilities were noted.

## A.1 Not-so-small probability of successful authentication without knowing the secret key

For the correct authentication the adversary has to answer to the challenge $R \in \{0,1\}^{Rlen-rlen}$ by $E_{k_{id}}(C \parallel r \parallel R)$ for some $r \in \{0,1\}^{rlen}$ and $C \in Consts$. This change shortens the length of $R$, which helps adversary to simplify the brute-force search.

## A.2 Big control over $IV$ generation

In modes where the additional data encryption is used, the $IV$ generation is combined with authentication, namely:

$$IV = E_{k_{id}}(const \parallel r \parallel R),$$

where $r$ is chosen by the party encrypting the data; $R$ is chosen by the second participant. Since the adversary is able to repeat the same $R$ for different

authentication attempts, it gets a lot of control over the generation of $IV$, which is very undesirable property for the most of modes of encryption. For instance, for the algorithm using 64-bit block cipher, the party which encrypts information adds only 20 bits of randomness $r \in \{0,1\}^{20}$. If the adversary will always repeat the same $R$, the collision of $r$ is expected after $\approx 2^{10}$ queries. Thus, with a high probability after $2^{10}$ attempts, the adversary can obtain a $IV$ collision, the consequences of this collision may be different for different encryption modes, in general this event is undesirable. For example, CBC mode does not provide security for repeating $IV$ (see for example [27]). CTR mode is completely insecure with repeating IV [27]. Consequently, the confidentiality property of additional data is violated. Moreover, the adversary gets the opportunity to respond to the same $R$ request with two different ciphertexts. In particular, it is able to replace one ciphertext with another without disrupting the protocol run. That is, the integrity property (at the session level) is violated. Note that in the 128-bit version, this problem is also present: a random number $r \in \{0,1\}^{32}$, which allows us to expect a collision after $\approx 2^{16}$ requests.

### A.3 Authentication as a decryption oracle

In protocols [4, 5], the authentication key is the same as the encryption key (for both operations the shared key $k_{id}$ is used). This allows the adversary to use authentication as an oracle to decrypt ciphertext blocks. In particular, authentication in the $IAM$ mode has the form:

$$E_{k_{id}}^{-1}(C \parallel r \parallel R),$$

where $R$ is chosen by the adversary, $C$ is a protection mode constant. Thus, if the adversary has a ciphertext block starting with the corresponding 4 bits of $C$, then it can always feed a part of the ciphertext block $R$ and wait until the $r$ value (which is chosen by the second participant) accidentally matches with the second part of the ciphertext block. For an algorithm using a 64-bit block cipher, after $q$ requests the block will be decrypted by the adversary with a probability $\frac{q}{2^{20}}$ (provided that the block has the correct structure, that is, it starts with the correct 4-bit constant). For a 128-bit block cipher, this probability is $\frac{q}{2^{32}}$ ( provided that the block has the required structure, that is, it starts with the correct 16-bit constant). This attack violates the confidentiality property.

# B Auxiliary definitions and models

In this section we introduce some additional adversarial models required to prove the security of an authentication protocol with optional data transfer in the AUTH$^+$ model.

Denote by $T$ the time required to simulate the entire experiment AUTH$^+$, which consists of the following steps.

1. Initializing the Reader record *Reader*.
2. Generating $3d$ independent keys $(k, k^e, k^m)$ (i.e., Tags initialization $CreateTag$).
3. Maintaining tables consistently throughout the experiment ($Tags$, $Sessions$ and $Reader.database$).
4. Processing messages according to the protocol (encryption, MAC calculation, response generation).
5. Updating the structures throughout the experiment (in particular, generating *sid*, tracking the number of the last running session, updating the states of the Tag and the Reader in different sessions).
6. Simulating verifications ($Test$ oracle): MAC verification, responses verification.

We will use the value $T$ in all further estimates of the adversary's success probabilities in intermediate models.

**Remark 5.** *The time $T$ can be estimated as follows:*

$$T \le 3d + 5 \cdot \sum_i p_i + 2 \cdot \sum_i \Big(\psi_i + r_i + 2\phi_i + 2q_i + 2\widehat{\phi}_i + 2\widehat{q}_i\Big),$$

*the terms correspond to the execution time of the following steps.*

- *Generating at most $3d$ different keys.*
- *Simulating no more than 5 message transmissions for each protocol run: $5 \cdot \sum_i p_i$.*
- *Processing (encryption, decryption) no more than 2 requests for each protocol run: $4 \cdot \sum_i p_i$.*
- *Generating at most $\sum_i (q_i + \widehat{q}_i)$ different IV during encryption.*

– *Processing (encryption, decryption) no more than* $2 \cdot \sum_i \left(\phi_i + \widehat{\phi}_i + q_i + \widehat{q}_i\right)$ *message blocks in total.*

– *Processing (calculation, MAC checking) no more than* $2 \cdot \sum_i \left(\psi_i + r_i + \phi_i + q_i + \widehat{\phi}_i + \widehat{q}_i\right)$ *message blocks in total.*

### B.1 PRP-security of block cipher

Let us consider a family of permutations $\pi_k$ on the set Dom indexed by some key $k$ from the set of keys Keys. An example of such a family is a block cipher with a key length of 256 bits and a block length of 64 bits:

$$\pi_k = E(k, \cdot),\ \text{Keys} = \{0,1\}^{256},\ \text{Dom} = \{0,1\}^{64}.$$

**Definition 8.** *A family of permutations*

$$\pi : \text{Keys} \times \text{Dom} \to \text{Dom}$$

*is called pseudorandom if it is computationally indistinguishable (for a uniformly random choice of a key from the set* Keys*) from a random permutation on the set* Dom*.*

There are two following models for the security of a block cipher:

– In the first one only the encryption oracle is available to the adversary. The adversary can choose the plaintext $m$ and obtain the corresponding ciphertext $c = \mathcal{O}(m)$.

– In the second model the adversary also has access to the decryption oracle. The adversary can feed ciphertext $c$ to the decryption oracle and obtain $\mathcal{O}^{-1}(c)$.

Next, we give a formal definition of PRP and sPRP models.

**Definition 9.** *Let us define an advantage of the adversary* $\mathcal{A}$ *in the* PRP*-model to be:*

$$\mathsf{Adv}_E^{\text{prp}}(\mathcal{A}) = \mathbb{P}\left[\text{Exp}^{\text{Left}}(\mathcal{A}) \to 1\right] - \mathbb{P}\left[\text{Exp}^{\text{Right}}(\mathcal{A}) \to 1\right]$$

*where* $\text{Exp}^{\text{Left}}$ *and* $\text{Exp}^{\text{Right}}$ *are defined as follows:*

| $\underline{\text{Exp}^{\text{Left}}(\mathcal{A})}$ | $\underline{\text{Exp}^{\text{Right}}(\mathcal{A})}$ |
|---|---|
| $k \xleftarrow{\$} \mathsf{KGen}$ | $\pi \xleftarrow{\$} S_{\text{Dom}}$ |
| $b' \xleftarrow{\$} \mathcal{A}^{\mathcal{O}}$ | $b' \xleftarrow{\$} \mathcal{A}^{\mathcal{O}}$ |
| **return** $b'$ | **return** $b'$ |
| $\underline{\mathcal{O}(m)}$ | $\underline{\mathcal{O}(m)}$ |
| **return** $E_k(m)$ | **return** $\pi(m)$ |

**Definition 10.** *Let* $\mathtt{Adv}^{\mathrm{PRP}}(t,p)$ *be the maximal value among* $\mathsf{Adv}_E^{\mathrm{prp}}(\mathcal{A})$*, where* $\mathcal{A}$*'s attack time does not exceed* $t$ *and* $\mathcal{A}$ *makes no more than* $p$ *queries to the oracle* $\mathcal{O}$*.*

**Definition 11.** *Let us define an advantage of adversary* $\mathcal{A}$ *in the* sPRP*-model:*

$$\mathsf{Adv}_E^{\mathrm{sprp}}(\mathcal{A}) = \mathbb{P}\left[\mathrm{Exp}^{\mathrm{Left}}(\mathcal{A}) \to 1\right] - \mathbb{P}\left[\mathrm{Exp}^{\mathrm{Right}}(\mathcal{A}) \to 1\right]$$

*where* $\mathrm{Exp}^{\mathrm{Left}}$ *and* $\mathrm{Exp}^{\mathrm{Right}}$ *are defined as follows:*

| $\underline{\mathrm{Exp}^{\mathrm{Left}}(\mathcal{A})}$ | $\underline{\mathrm{Exp}^{\mathrm{Right}}(\mathcal{A})}$ |
|---|---|
| $k \xleftarrow{\$} \mathsf{KGen}$ | $\pi \xleftarrow{\$} S_{\mathrm{Dom}}$ |
| $b' \xleftarrow{\$} \mathcal{A}^{\mathcal{O},\mathcal{O}^{-1}}$ | $b' \xleftarrow{\$} \mathcal{A}^{\mathcal{O},\mathcal{O}^{-1}}$ |
| **return** $b'$ | **return** $b'$ |
| $\underline{\mathcal{O}(m)}$ | $\underline{\mathcal{O}(m)}$ |
| **return** $E_k(m)$ | **return** $\pi(m)$ |
| $\underline{\mathcal{O}^{-1}(m)}$ | $\underline{\mathcal{O}^{-1}(m)}$ |
| **return** $E_k^{-1}(m)$ | **return** $\pi^{-1}(m)$ |

**Definition 12.** *Let* $\mathtt{Adv}^{\mathrm{sPRP}}(t,q,\widehat{q})$ *be the maximal value among* $\mathsf{Adv}_E^{\mathrm{sprp}}(\mathcal{A})$*, where* $\mathcal{A}$*'s attack time does not exceed* $t$ *and* $\mathcal{A}$ *makes no more than* $q$ *queries to the oracle* $\mathcal{O}$ *and no more than* $\widehat{q}$ *requests to the oracle* $\mathcal{O}^{-1}$*.*

## B.2 Existing estimates of the PRP and sPRP security of block ciphers

The standard assumption for block ciphers is that the most successful distinguishing attack is a brute force attack, in both the PRP-model and the sPRP-model. Then the following approximations hold:

$$\mathtt{Adv}^{\mathrm{PRP}}(t,p) = \mathtt{Adv}^{\mathrm{sPRP}}(t,q,\widehat{q}) \approx \frac{t}{2^{klen}},$$

where *klen* denotes the length of the key; $p = q + \widehat{q}$; testing a key takes 1 cycle of computation.

For the «Magma» algorithm, it is reasonable to take the following approximation (based on attacks from articles [20, 21]):

$$\mathtt{Adv}^{\mathrm{PRP}}(t,p) \approx \frac{q \cdot t}{2^{256}}.$$

## B.3 Adversarial model for confidentiality of a pair of encryption modes

### B.3.1 Single key confidentiality model

Let us describe the $LOR2$ model of the indistinguishability for a pair of encryption modes.

**Definition 13.** *An encryption scheme $\mathcal{SE}$ with a pair of associated encryption modes $(\mathsf{Enc}, \widehat{\mathrm{Enc}})$ is a set of five (probabilistic) algorithms:*

- *Algorithm for generating the key of encryption modes* $\mathsf{KGen}$.
- *A pair of (probabilistic) encryption algorithms* $(\mathsf{Enc}, \widehat{\mathrm{Enc}})$.
- *A pair of (deterministic) decryption algorithms* $(\mathsf{Dec}, \widehat{\mathrm{Dec}})$.

*Encryption and decryption algorithms must satisfy the standard decryption correctness requirement:*

$$\mathsf{Dec}(k, \mathsf{Enc}(k, m)) = m, \quad \widehat{\mathrm{Dec}}(k, \widehat{\mathrm{Enc}}(k, m)) = m$$

The main difference from the «ordinary» encryption scheme is that two algorithms are «interwined» by using one shared secret key, which (theoretically) gives the adversary more opportunities to attack. Next, we present a formalization of CPA-security for a pair of associated modes $(\mathsf{Enc}, \widehat{\mathrm{Enc}})$.

During the experiment, the adversary interacts with two oracles: $\mathrm{LOR}^b$ and $\widehat{\mathrm{LOR}}^b$. The adversary can feed to the oracles two messages of the same length:

$$(M_0, M_1), \quad |M_0| = |M_1|.$$

The oracle selects the message $M_b$, encrypts it according to the encryption mode $\mathsf{Enc}$ (for the $\mathrm{LOR}^b$ oracle) or $\widehat{\mathrm{Enc}}$ (for the $\widehat{\mathrm{LOR}}^b$ oracle) and returns the result to the adversary. The goal of the adversary is to determine the value of bit $b$ using the received ciphertexts. If the adversary is able to distinguish between these experiments with a high probability, then it means that it is able to recover partial information about the plaintext from the ciphertexts.

Note that in our description of the model the adversary has access to two oracles at once (which corresponds to the adversary's ability to send a message for encryption to the Tag (processes messages in the $\mathsf{Enc}$ mode) and to the Reader (processes messages in the $\widehat{\mathrm{Enc}}$ mode), the Tag and the Reader have shared encryption key). Even in this case the adversary should not be able to recover partial information about messages.

**Definition 14.** *Let us define an advantage of adversary $\mathcal{A}$ in the $LOR2$ model to be:*

$$\mathsf{Adv}^{\mathrm{lor2}}_{\mathcal{SE}}(\mathcal{A}) = \mathbb{P}\left[\mathrm{Exp}^{LOR2-1}(\mathcal{A}) \to 1\right] - \mathbb{P}\left[\mathrm{Exp}^{LOR2-0}(\mathcal{A}) \to 1\right],$$

*where* $\mathrm{Exp}^{LOR2-b}$, $b \in \{0,1\}$ *is defined as follows:*

| $\mathrm{Exp}^{LOR2-b}_{\mathcal{SE}}(\mathcal{A})$ | $\mathrm{LOR}^b(M_0, M_1)$ |
|---|---|
| $k \xleftarrow{\$} \mathsf{KGen}$ | **return** $\mathsf{Enc}(k, M_b)$ |
| $b' \xleftarrow{\$} \mathcal{A}^{\mathrm{LOR}^b, \widehat{\mathrm{LOR}}^b}$ | $\widehat{\mathrm{LOR}}^b(M_0, M_1)$ |
| **return** $b'$ | **return** $\widehat{\mathrm{Enc}}(k, M_b)$ |

**Definition 15.** *Let* $\mathsf{Adv}^{LOR2}(t, q, \mu, \phi, \widehat{q}, \widehat{\mu}, \widehat{\phi})$ *be the maximal value among adversarial advantages in the $LOR2$ model, where the maximum is taken over all adversaries $\mathcal{A}$ with the following restrictions:*

- *$\mathcal{A}$'s attack time does not exceed $t$;*
- *the number of queries $(M_0, M_1)$ to the oracle* LOR *(*$\widehat{\mathrm{LOR}}$*) does not exceed $q$ (*$\widehat{q}$ *resp.);*
- *maximal length $|M_b|$ of query $(M_0, M_1)$ to the oracle* LOR *(*$\widehat{\mathrm{LOR}}$*) does not exceed $\mu$ (*$\widehat{\mu}$ *resp.);*
- *total length of queries of the type $(M_0, M_1)$ to oracle* LOR *(*$\widehat{\mathrm{LOR}}$*) does not exceed $\phi$ (*$\widehat{\phi}$ *resp.);*

### B.3.2 Multiple keys confidentiality model

We also introduce an additional model $LOR2_d$, in which the adversary interacts with $d$ pairs of oracles $\mathrm{LOR}^b$ and $\widehat{\mathrm{LOR}}^b$ on different keys $k_1, \ldots, k_d$, but with the same fixed bit $b$. This model formalizes the situation in which the Reader interacts with $d$ independent Tags.

**Definition 16.** *Let us define an advantage of the adversary $\mathcal{A}$ in the model $LOR2_d$ to be*

$$\mathsf{Adv}^{\mathrm{lor2_d}}_{\mathcal{SE}}(\mathcal{A}) = \mathbb{P}\left[\mathrm{Exp}^{LOR2_d-1}(\mathcal{A}) \to 1\right] - \mathbb{P}\left[\mathrm{Exp}^{LOR2_d-0}(\mathcal{A}) \to 1\right],$$

*where* $\mathrm{Exp}^{LOR2_d-b}$, $b \in \{0,1\}$, *is defined as follows:*

$\underline{\mathrm{Exp}_{\mathcal{SE}}^{LOR2_d-b}(\mathcal{A})}$

**for** $i = 1..d$ **do**

  $k[i] \xleftarrow{\$} \mathsf{KGen}$

**endfor**

$b' \xleftarrow{\$} \mathcal{A}^{LOR_d^b, \widehat{\mathrm{LOR}}^b}$

**return** $b'$

$\underline{LOR_d^b(i, M_0, M_1)}$

**return** $\mathsf{Enc}(k[i], M_b)$

$\underline{\widehat{\mathrm{LOR}}^b(i, M_0, M_1)}$

**return** $\widehat{\mathrm{Enc}}(k[i], M_b)$

**Definition 17.** *Let* $\mathtt{Adv}^{LOR2_d}(t, \mathcal{Q}, \mathcal{M}, \Phi, \widehat{\mathcal{Q}}, \widehat{\mathcal{M}}, \widehat{\Phi})$ *be the maximal value among adversarial advantages in the* $LOR2_d$ *model, where the maximum is taken over all adversaries* $\mathcal{A}$ *with the following restrictions (i ranges from 1 to d):*

- $\mathcal{A}$*'s attack time does not exceed* $t$*;*
- *the number of queries of the type* $(i, M_0, M_1)$ *to oracle* $LOR_d$ *(*$\widehat{LOR_d}$*) does not exceed* $q_i = \mathcal{Q}[i]$ *(*$\widehat{q_i} = \widehat{\mathcal{Q}}[i]$ *resp.);*
- *maximal length* $|M|$ *of query of the type* $(i, M_0, M_1)$ *to oracle* $LOR_d$ *(*$\widehat{LOR_d}$*) does not exceed* $\mu_i = \mathcal{M}[i]$ *(*$\widehat{\mu}_i = \widehat{\mathcal{M}}[i]$ *resp.);*
- *total length of queries of the type* $(i, M_0, M_1)$ *to oracle* $LOR_d$ *(*$\widehat{LOR_d}$*) does not exceed* $\phi_i = \Phi[i]$ *(*$\widehat{\phi}_i = \widehat{\Phi}[i]$ *resp.);*

Since the keys in the $LOR2_d$-model are independent, then we can expect that the $LOR2_d$-model is reduced to $LOR2$ submodels for each of the keys, since the information obtained when encrypting messages on some key $k$ does not help in distinguishing encrypted messages on the key $k'$, which is chosen independently of $k$. The following theorem, in which $\mathtt{Adv}^{LOR2_d}$ is bounded from above using $\mathtt{Adv}^{LOR2}$, confirms this fact.

**Theorem 2.** *The following inequality holds:*

$$\mathtt{Adv}^{LOR2_d}(t, \mathcal{Q}, \mathcal{M}, \Phi, \widehat{\mathcal{Q}}, \widehat{\mathcal{M}}, \widehat{\Phi}) \le \sum_i \mathtt{Adv}^{LOR2}(t + T, q_i, \mu_i, \phi_i, \widehat{q}_i, \widehat{\mu}_i, \widehat{\phi}_i).$$

*Proof.* Let us apply the standard hybrid argument technique. We describe a series of experiments $\mathrm{Exp}_i$, where each experiment differs from the previous one only in how exactly queries of the type $(i, M_0, M_1)$ are processed within some fixed $i$. In the experiment $\mathrm{Exp}_0$ all oracles process the message $M_1$ for all queries of the type $(i, M_0, M_1)$ for any $i$. In the experiment $\mathrm{Exp}_1$ oracles process message $M_0$ for queries of the type $(1, M_0, M_1)$, for all other queries with $i \neq 1$ — message $M_1$. In general, in the experiment $\mathrm{Exp}_i$ oracles process the message $M_0$ for queries of the type $(j, M_0, M_1)$, $j \le i$, and message $M_1$ for all queries of the type $(j, M_0, M_1)$, $j > i$.

| $(1, M_0, M_1)\ (2, M_0, M_1)\ \ldots$ $\ldots (i-1, M_0, M_1)$ $\vdots$ Simulation, processing $M_0$ | $(i, M_0, M_1)$ $\vdots$ Query to Oracle | $(i+1, M_0, M_1)\ (i+2, M_0, M_1)\ \ldots$ $\ldots (d, M_0, M_1)$ $\vdots$ Simulation, processing $M_1$ |
|---|---|---|

Let $\mathcal{A}$ be the adversary for the original experiment $LOR2_d$ with computational constraints $(t, \mathcal{Q}, \mathcal{M}, \Phi, \widehat{\mathcal{Q}}, \widehat{\mathcal{M}}, \widehat{\Phi})$. Denote by $\mathcal{A}^{(b_1,b_2,\ldots,b_d)}$ the adversary $\mathcal{A}$ interacting with the oracles $LOR_d$ and $\widehat{LOR_d}$ which process message $M_{b_i}$ for a query of the type $(i, M_0, M_1)$. We also introduce the notation $b^k = \underbrace{(b, b, \ldots, b)}_{k\text{times}}$, $b \in \{0, 1\}$. Then the following equality holds:

$$\mathsf{Adv}^{\mathrm{lor2_d}}_{\mathcal{SE}}(\mathcal{A}) = \mathbb{P}\left[\mathcal{A}^{(1,1,\ldots,1)} \to 1\right] - \mathbb{P}\left[adv^{(0,0,\ldots,0)} \to 1\right] =$$

$$= \mathbb{P}\left[\mathcal{A}^{(1^d)} \to 1\right] - \mathbb{P}\left[adv^{(0,1^{d-1})} \to 1\right] +$$

$$+ \mathbb{P}\left[\mathcal{A}^{(0,1^{d-1})} \to 1\right] - \mathbb{P}\left[adv^{(0^2,1^{d-2})} \to 1\right] + \ldots$$

$$\ldots + \mathbb{P}\left[\mathcal{A}^{(0^{d-1},1)} \to 1\right] - \mathbb{P}\left[adv^{(0^d)} \to 1\right].$$

We show that the $i$-th term in the sum can be estimated as $\mathsf{Adv}^{\mathrm{lor2}}_{\mathcal{SE}}(\mathcal{B})$, where $\mathcal{B}$ has computational constraints $(t + T, q_i, \mu_i, \phi_i, \widehat{q}_i, \widehat{\mu}_i, \widehat{\phi}_i)$.

The adversary $\mathcal{B}$ acts within the framework of the experiment $LOR2$, that is, it can interact with two «interwined» oracles $\mathrm{LOR}^b$ and $\widehat{\mathrm{LOR}}^b$ with a fixed unknown key $k$ and bit $b$. The goal of $\mathcal{B}$ is to determine the bit $b$. Adversary $\mathcal{B}$ uses adversary $\mathcal{A}$ as a subroutine. The adversary $\mathcal{B}$ will do the following:

– Generates $d-1$ keys $k_1, \ldots, k_{i-1}, k_{i+1}, \ldots, k_d$,

– For queries of the type $(j, M_0, M_1)$, $j < i$ from adversary $\mathcal{A}$ to oracle $LOR_d$ adversary $\mathcal{B}$ takes *left* message $M_0$, process it in $\mathsf{Enc}$ mode on key $k_j$ and return it to adversary $\mathcal{A}$. Similar actions (with $\mathsf{Enc}$ mode replaced by $\widehat{\mathrm{Enc}}$ mode) are to be done when $\mathcal{A}$ queries oracle $\widehat{LOR_d}$.

– For queries of the type $(j, M_0, M_1)$, $j > i$ from adversary $\mathcal{A}$ to oracle $LOR_d$ adversary $\mathcal{B}$ takes *right* message $M_1$, process it in $\mathsf{Enc}$ mode on key $k_j$ and return it to adversary $\mathcal{A}$. Similar actions (with $\mathsf{Enc}$ mode replaced by $\widehat{\mathrm{Enc}}$ mode) are to be done when $\mathcal{A}$ queries oracle $\widehat{LOR_d}$.

– When query of the type $(i, M_0, M_1)$ is processed, adversary $\mathcal{B}$ redirects the query to its own oracle LOR or $\widehat{\text{LOR}}$ respectively (depending on which oracle the original adversary's $\mathcal{A}$ query was sent to).

The running time of the $\mathcal{B}$ is equal to the running time of the adversary $\mathcal{A}$ plus additional time for processing queries like $(j, M_0, M_1)$, $j \neq i$, which can be bounded from above by $T$. The resources of the adversary $\mathcal{B}$ with respect to LOR and $\widehat{\text{LOR}}$ oracles are equal to $(q_i, \mu_i, \phi_i)$ and $(\widehat{q}_i, \widehat{\mu}_i, \widehat{\phi}_i)$, respectively.

The adversary $\mathcal{B}$ returns the same bit as the adversary $\mathcal{A}$. Hence, the advantage of $\mathcal{B}$ equals:

$$\mathsf{Adv}^{\text{lor2}}_{\mathcal{SE}}(\mathcal{B}) = \mathbb{P}\left[\mathcal{B}^1 \to 1\right] - \mathbb{P}\left[\mathcal{B}^0 \to 1\right] =$$

$$\mathbb{P}\left[\mathcal{A}^{(0^{i-1},1,1^{d-i})} \to 1\right] - \mathbb{P}\left[\mathcal{A}^{(0^{i-1},0,1^{d-i})} \to 1\right].$$

Thus, the $i$-th difference in the sum can be bounded from above as:

$$\mathbb{P}\left[\mathcal{A}^{(0^{i-1},1,1^{d-i})} \to 1\right] - \mathbb{P}\left[\mathcal{A}^{(0^{i-1},0,1^{d-i})} \to 1\right] \leq$$

$$\leq \mathtt{Adv}^{LOR2}(t + T, q_i, \mu_i, \phi_i, \widehat{q}_i, \widehat{\mu}_i, \widehat{\phi}_i),$$

and the whole sum is bounded from above as:

$$\mathsf{Adv}^{\text{lor2}_\text{d}}_{\mathcal{SE}}(\mathcal{A}) \leq \sum_i \mathtt{Adv}^{LOR2}(t + T, q_i, \mu_i, \phi_i, \widehat{q}_i, \widehat{\mu}_i, \widehat{\phi}_i).$$

Taking the maximum over all adversaries $\mathcal{A}$ on the left side of the inequality (with a constraint on the resources used) we obtain the statement of the theorem. □

Note that the theorem proved above implies a simpler, but less precise estimate.

**Corollary 2.** *Let us introduce the following notation:*

$$q = \max_i q_i, \mu = \max_i \mu_i,$$

$$\widehat{q} = \max_i \widehat{q}_i, \widehat{\mu} = \max_i \widehat{\mu}_i.$$

*Then the following inequality holds:*

$$\mathtt{Adv}^{LOR2_d}(t, \mathcal{Q}, \mathcal{M}, \Phi, \widehat{\mathcal{Q}}, \widehat{\mathcal{M}}, \widehat{\Phi}) \leq d \cdot \mathtt{Adv}^{LOR2}(t + T, q, \mu, q \cdot \mu, \widehat{q}, \widehat{\mu}, \widehat{q} \cdot \widehat{\mu}).$$

### B.3.3 Existing estimates for pair of encryption modes

For a pair of modes $(CBC, \widehat{CBC})$ for the «Magma» algorithm, the following estimate was obtained (see article [22]):

$$\mathtt{Adv}^{LOR2}(t, q, \mu, \phi, \widehat{q}, \widehat{\mu}, \widehat{\phi}) \leq$$
$$\leq \frac{\widehat{\phi}^2}{2^{64} - \widehat{\phi}} + \frac{\phi^2}{2^{64} - \phi} + \frac{(\phi + \widehat{\phi})^2}{2^{64}} + \frac{(\phi + \widehat{\phi}) \cdot (t + \phi + \widehat{\phi})}{2^{255}}.$$

## B.4 Adversarial models for integrity

### B.4.1 Single key integrity model

To investigate integrity of transmitted data we will use the standard $\mathrm{EUF} - \mathrm{CMA}$ model of forging MAC tag for the message in **deterministic MAC function** setting (see [7, 24]).

An adversary is given access to the MAC calculation oracle $\mathcal{O}$ and MAC verification oracle $Verify$. It is able to adaptively choose messages $m$ and obtain MAC-tags $\tau$ for them (using $\mathcal{O}$ queries) under a fixed (unknown to the adversary) key $k$.

The ultimate goal is to forge a tag $\tau$ for a message $m$ that was not queried before, i.e. to obtain a pair $(m, \tau)$ such that $\tau$ is a valid tag for $m$ (under a key $k$), and $m$ does not belong to the set of input queries of $\mathcal{O}$.

Let $MAC_k(m)$ be a function that computes MAC tag under a key $k$ for a message $m$.

**Definition 18.** *Define an advantage of adversary $\mathcal{A}$ in the* $\mathrm{EUF} - \mathrm{CMA}$ *model to be:*

$$\mathsf{Adv}^{\mathrm{euf-cma}}_{MAC}(\mathcal{A}) = \mathbb{P}\left[\mathrm{Exp}^{\mathrm{EUF-CMA}}(\mathcal{A}) \to 1\right],$$

*where* $\mathrm{Exp}^{\mathrm{EUF-CMA}}$ *is defined as follows:*

| $\mathrm{Exp}^{\mathrm{EUF-CMA}}(\mathcal{A})$ | $\mathcal{O}(m)$ | $Verify(m, \tau)$ |
|---|---|---|
| $k \xleftarrow{\$} \mathsf{KGen}$ | $sent \leftarrow sent \cup \{m\}$ | $res \leftarrow (\tau = MAC_k(m))$ |
| $sent = \emptyset$ | **return** $MAC(k, m)$ | **if** $(m \notin sent)$ & $(res = true)$ |
| $win \leftarrow false$ | | $win \leftarrow true$ |
| $\mathcal{A}^{\mathcal{O}, Verify}$ | | **fi** |
| **return** $win$ | | **return** $res$ |

**Definition 19.** *Let* $\mathtt{Adv}^{\mathrm{EUF-CMA}}(t, r, \nu, \psi, \theta)$ *be the maximal value among adversarial advantages in the* $\mathrm{EUF} - \mathrm{CMA}$ *model, where the maximum is taken over all adversaries $\mathcal{A}$ with the following restrictions:*

– *$\mathcal{A}$'s attack time does not exceed $t$;*

– *the number of $\mathcal{O}$ queries does not exceed $r$;*

– *maximal length of $\mathcal{O}$ query $\max_m |m|$ does not exceed $\nu$;*

– *total length of $\mathcal{O}$ queries $\sum_m |m|$ does not exceed $\psi$;*

– *the number of $Verify$ queries does not exceed $\theta$.*

#### B.4.2 Multiple keys integrity model

As it was done in section B.3.2, we can expand the basic single key $\mathrm{EUF}-\mathrm{CMA}$ model (see prev. section B.4.1) to the case of multiple keys $k_i$, $i = 1, \ldots, d$.

**Definition 20.** *Let us define an advantage of adversary $\mathcal{A}$ in the $\mathrm{EUF}-\mathrm{CMA}_d$ model to be:*

$$\mathsf{Adv}^{\mathrm{euf-cma_d}}_{MAC}(\mathcal{A}) = \mathbb{P}\left[\mathrm{Exp}^{\mathrm{EUF-CMA}_d}(\mathcal{A}) \to 1\right],$$

*where $\mathrm{Exp}^{\mathrm{EUF-CMA}_d}$ is defined as follows:*

$\mathrm{Exp}^{\mathrm{EUF-CMA}_d}(\mathcal{A})$

**for** $i = 1..d$ **do**
  $k[i] \xleftarrow{\$} \mathsf{KGen}$
  $sent[i] = \emptyset$
**endfor**
$win \leftarrow false$
$\mathcal{A}^{\mathcal{O},Verify}$
**return** $win$

$\mathcal{O}(i, m)$

$sent[i] \leftarrow sent[i] \cup \{m\}$
**return** $MAC(k[i], m)$

$Verify(i, m, \tau)$

$res \leftarrow (\tau = MAC_{k[i]}(m))$
**if** $(m \notin sent[i])$ & $(res = true)$
  $win \leftarrow true$
**fi**
**return** $res$

**Definition 21.** *Let $\mathtt{Adv}^{\mathrm{EUF-CMA}_d}(t, \mathcal{R}, \mathcal{N}, \Psi, \Theta)$ be the maximal value among adversarial advantages in the $\mathrm{EUF}-\mathrm{CMA}_d$-model, where the maximum is taken over all adversaries $\mathcal{A}$ with the following restrictions:*

– *$\mathcal{A}$'s attack time does not exceed $t$;*

– *the number of $\mathcal{O}$ queries of the type $(i, m)$ does not exceed $r_i = \mathcal{R}[i]$;*

– *maximal length of $\mathcal{O}$ query of the type $(i, m)$ $\max_{(i,m)} |m|$ does not exceed $\nu_i = \mathcal{N}[i]$;*

– *total length of $\mathcal{O}$ queries of the type $(i, m)$ $\sum_{(i,m)} |m|$ does not exceed $\psi_i = \Psi[i]$;*

– *the number of* $Verify$ *queries of the type* $(i, m)$ *does not exceed* $\theta_i = \Theta[i]$.

Next theorem follows immediately from the hybrid argument (see also [28]).

**Theorem 3.** *The following inequality holds:*

$$\mathtt{Adv}^{\mathrm{EUF-CMA}_d}(t, \mathcal{R}, \mathcal{N}, \Psi, \Theta) \leq \sum_i \mathtt{Adv}^{\mathrm{EUF-CMA}}(t + T, r_i, \nu_i, \psi_i, \theta_i),$$

*Proof.* Let $\mathcal{A}$ be the adversary in $\mathrm{EUF} - \mathrm{CMA}_d$ model. Let $\mathsf{F}$ be an event that $\mathcal{A}$ successfully forges a tag for some of the keys $k[j]$, $j = 1, \ldots, d$, $\mathsf{F_i}$ be an event that $\mathcal{A}$ successfully forges a tag for a particular key $k[i]$.

Then we have $\mathsf{F} = \cup_i \mathsf{F_i}$,

$$\mathsf{Adv}^{\mathrm{euf-cma_d}}_{MAC}(\mathcal{A}) = \mathbb{P}[\mathsf{F}] \leq \sum_i \mathbb{P}[\mathsf{F_i}]\,.$$

$\mathcal{A}$ makes no more than $r_i$ queries to the $\mathcal{O}$ of the type $(i, m)$, maximal length of query of the type $(i, m)$ does not exceed $\nu_i$, total length queries of the type $(i, m)$ does not exceed $\psi_i$, the number of $Verify$ queries of the type $(i, m)$ does not exceed $\theta_i$.

All queries of the type $(j, m)$, $j \neq i$, can be simulated (as it was done in section B.3.2). The time needed for simulation can be bounded from above by $T$. Thus, the probability $\mathbb{P}[\mathsf{F_i}]$ can be estimated as follows:

$$\mathbb{P}[\mathsf{F_i}] \leq \mathtt{Adv}^{\mathrm{EUF-CMA}}(t + T, r_i, \nu_i, \psi_i, \theta_i),$$

hence the theorem. □

#### B.4.3 Existing results for $\mathrm{EUF} - \mathrm{CMA}$ model

The (standard) problem of forging MAC tag (based on CMAC) was investigated earlier in [23, 25, 26, 27]. Three (incomparable) estimates are presented in the literature [23, 25, 26]. For concrete parameters $(r, \nu, \psi, \theta)$ the minimum among the three can be used.

### B.5 Adversarial model for authentication value forgery

#### B.5.1 Single key model

In this section we present a model for studying the authentication property.

In TAM, MAM modes without additional data transmission in order to successfully authenticate an adversary is forced to make a forgery of the form $E_k(C \parallel R)$ for a randomly chosen $R$ of length $Rlen$ and constant $C$, chosen accordingly to the protocol specification.

In the model an adversary is given access to the oracles $\mathcal{E}$, $Verify$.

– $\mathcal{E}$ encrypts a given block $(C, x)$, where the length of $x$ (in bits) is $Rlen$, $C$ is one of the constants for the protection mode;

– $Verify$ checks the correctness of a forgery on a randomly chosen $R$ (see below); it can be called only once in the experiment (which corresponds to the real-world situation: one can fail authentication only once, after that a new challenge will be generated);

**Definition 22.** *Let us define an advantage of adversary $\mathcal{A}$ in the model* Chal *to be:*

$$\mathsf{Adv}^{\mathrm{chal}}(\mathcal{A}) = \mathbb{P}\left[\mathrm{Exp}^{\mathrm{Chal}}(\mathcal{A}) \to 1\right]$$

*where* $\mathrm{Exp}^{\mathrm{Chal}}$ *is defined as follows:*

$\mathrm{Exp}^{Chal}(\mathcal{A})$

$win \leftarrow \mathsf{false}$
$k \xleftarrow{\$} \mathsf{KGen}$
$R \leftarrow \varnothing$
$(Const, state) \xleftarrow{\$} \mathcal{A}^{\mathcal{E}}$
$check \leftarrow \mathsf{true}$
$R \xleftarrow{\$} \{0,1\}^{Rlen}$
$\mathcal{A}^{\mathcal{E},Verify}(R, state)$
**return** $win$

$\mathcal{E}(C, x)$

**if** $(C = Const)\,\&\,(x = R)$
  $check \leftarrow \mathsf{false}$
**fi**
**return** $E_k(C \parallel x)$

$Verify(y)$

$win \leftarrow win \vee (check\,\&\,(E_k(Const \parallel R) = y))$
**return** $win$

**Definition 23.** *Let* $\mathtt{Adv}^{\mathrm{Chal}}(t, p)$ *be the maximal value among adversarial advantages in the* Chal *model, where the maximum is taken over all adversaries $\mathcal{A}$ with the following restrictions:*

– *$\mathcal{A}$'s attack time does not exceed $t$;*

– *the number of $\mathcal{E}$ queries does not exceed $p$;*

In the model defined above the adversary interacts with the oracles in two steps:

– At the first step an adversary $\mathcal{A}$ is able to interact with the oracle $\mathcal{E}$. At the end of the step it outputs a pair ($Const$, $state$), where $Const$ is a

constant for which it will try to finish authentication successfully, *state* is some «inner» information needed at the second step.

– At the second step the Experimenter chooses random $R$ of length $Rlen$ and gives it to the adversary with the variable *state*. The goal of the adversary is to forge $E_k(Const \parallel R)$. It is unable to feed a pair $(Const, R)$ to the $\mathcal{E}$ input. The adversary can check the validity of a forgery using $Verify$ oracle.

#### B.5.2 Estimating the advantage

The security of the protocol in the Chal model is based on PRP-security of block cipher $E_K(\cdot)$ (see section B.1) and a probability of a forgery for random permutation.

**Theorem 4.** *The following inequality holds:*

$$\mathtt{Adv}^{\mathrm{Chal}}(t,p) \le \mathtt{Adv}^{\mathrm{PRP}}(t,p+1) + \frac{p}{2^{Rlen}} + \frac{|Consts|}{|\mathrm{Dom}| - p},$$

*where $Consts$ is a set of constants used in the protocol.*

*Proof.* The proof consists of two steps. The first step is to replace each occurence of $E_k(m)$ with a random permutation $\pi(m)$. The second step is to obtain an estimate for a forgery probability in case of a random permutation $\pi$ instead of $E_k$ is used.

**Step 1: replacing $E_k$ with a random permutation.** In the Chal experiment we replace the choice of a random key $k$ with the choice of a random permutation on the set Dom, and $\mathcal{E}$ with the oracle returning $\pi(C \parallel x)$. Denote the resulting experiment as Chal′

Let $\mathcal{A}$ be an adversary for the $\mathrm{Exp}^{\mathrm{Chal}}$. Then we can write:

$$\mathsf{Adv}^{\mathrm{chal}}(\mathcal{A}) = \mathbb{P}\left[\mathrm{Exp}^{\mathrm{Chal}}(\mathcal{A}) \to 1\right] =$$

$$= \left(\mathbb{P}\left[\mathrm{Exp}^{\mathrm{Chal}}(\mathcal{A}) \to 1\right] - \mathbb{P}\left[\mathrm{Exp}^{\mathrm{Chal}'}(\mathcal{A}) \to 1\right]\right) + \mathbb{P}\left[\mathrm{Exp}^{\mathrm{Chal}'}(\mathcal{A}) \to 1\right].$$

Let us estimate the first summand. Let $\mathcal{A}$ be an adversary,

$$\left(\mathbb{P}\left[\mathrm{Exp}^{\mathrm{Chal}}(\mathcal{A}) \to 1\right] - \mathbb{P}\left[\mathrm{Exp}^{\mathrm{Chal}'}(\mathcal{A}) \to 1\right]\right) = \varepsilon.$$

Let us use $\mathcal{A}$ to build an adversary $\mathcal{B}$ in the $\mathrm{Exp}^{\mathrm{PRP}}$, who makes the same number of requests to its oracle (plus one additional request) and also has an advantage $\varepsilon$.

The adversary $\mathcal{B}$ is given an oracle access to $\mathcal{O}$, which implements either a pseudorandom permutation with a randomly chosen key $k$ or a random permutation. The adversary $\mathcal{B}$ uses the adversary $\mathcal{A}$ as a subroutine. When $\mathcal{A}$ requests encryption query of the form $(C, x)$, the adversary $\mathcal{B}$ forms a request $(C \parallel x)$ to the oracle $\mathcal{O}$.

In the second stage, $\mathcal{B}$ generates a random $R$ and gives it to the $\mathcal{A}$, and also simulates a request to the oracle $Verify$, namely, independently checks the correctness of the forgery (using the variable *check* and an additional request to the oracle $\mathcal{O}$) and returns the result of forgery verification.

The advantage of $\mathcal{B}$ is:

$$\mathsf{Adv}_E^{\mathrm{prp}}(\mathcal{B}) = \mathbb{P}\left[\mathrm{Exp}^{\mathrm{Left}}(\mathcal{B}) \to 1\right] - \mathbb{P}\left[\mathrm{Exp}^{\mathrm{Right}}(\mathcal{B}) \to 1\right].$$

Hence, in the Left experiment the adversary $\mathcal{B}$ ideally simulates the environment of the experiment Chal for the adversary $\mathcal{A}$. Thus,

$$\mathbb{P}\left[\mathrm{Exp}^{\mathrm{Left}}(\mathcal{B}) \to 1\right] = \mathbb{P}\left[\mathrm{Exp}^{\mathrm{Chal}}(\mathcal{A}) \to 1\right].$$

Analogously, in the Right experiment the adversary $\mathcal{B}$ ideally simulates the environment of the experiment Chal$'$ for the adversary $\mathcal{A}$:

$$\mathbb{P}\left[\mathrm{Exp}^{\mathrm{Right}}(\mathcal{B}) \to 1\right] = \mathbb{P}\left[\mathrm{Exp}^{\mathrm{Chal}'}(\mathcal{A}) \to 1\right].$$

Consequently,

$$\begin{gathered}\mathbb{P}\left[\mathrm{Exp}^{\mathrm{Chal}}(\mathcal{A}) \to 1\right] - \mathbb{P}\left[\mathrm{Exp}^{\mathrm{Chal}'}(\mathcal{A}) \to 1\right] = \\ = \mathbb{P}\left[\mathrm{Exp}^{\mathrm{Left}}(\mathcal{B}) \to 1\right] - \mathbb{P}\left[\mathrm{Exp}^{\mathrm{Right}}(\mathcal{B}) \to 1\right] = \\ = \mathsf{Adv}_E^{\mathrm{prp}}(\mathcal{B}) \leq \mathtt{Adv}^{\mathrm{PRP}}(t, p+1).\end{gathered}$$

**Step 2: estimating the probability of a forgery for a random permutation.** Let us now compute probability of correct response on the challenge $(C, R)$, assuming that $p[c]$ queries of the type $(C, x)$ were done.

Assume that the adversary chooses particular $C$ for the second part of the experiment, and a random challenge $R$ was chosen by the experiment.

1. If the adversary has previously asked $(C, R)$ for a given $C$, then it is able to finish experiment successfully with probability 1: $res = \mathcal{O}(C, R)$.
2. If the adversary has not previously asked $(C, R)$ for a given $C$, then $\pi(C \parallel R)$ could be any of the «remaining» values in the range of $\pi$, hence the success probability is:

$$\frac{1}{|\mathrm{Dom}| - p},$$

where $p = \sum_c p[c]$.

As a result, the probability of a successful forgery $\mathcal{O}(C, R)$ for a fixed $C$ can be bounded from above as:

$$\frac{p[c]}{2^{Rlen}} + \left(1 - \frac{p[c]}{2^{Rlen}}\right) \cdot \frac{1}{|\mathrm{Dom}| - p}.$$

The probability of choosing some fixed $C$ by the adversary does not exceed 1. Thus, we obtain the final estimate:

$$\mathtt{Adv}^{\mathrm{Chal}}(t, p) \le \mathtt{Adv}^{\mathrm{PRP}}(t, p+1) + \sum_c \frac{p[c]}{2^{Rlen}} + \left(1 - \frac{p[c]}{2^{Rlen}}\right) \cdot \frac{1}{|\mathrm{Dom}| - p} \le$$

$$\le \mathtt{Adv}^{\mathrm{PRP}}(t, p+1) + \frac{p}{2^{Rlen}} + \frac{|Consts|}{|\mathrm{Dom}| - p}$$

□

### B.5.3 The case of multiple verifications

We can consider the following generalization, in which the adversary can query $\theta$ verifications on randomly selected queries (the Chal-$\theta$ model, oracle $\mathcal{E}$ works similarly):

$\mathrm{Exp}^{\text{Chal-}\theta}(\mathcal{A})$

$k \xleftarrow{\$} \mathsf{KGen}$
$state \leftarrow \varnothing$
$win \leftarrow \mathsf{false}$
**for** $i = 1..\theta$ **do**
  $R \leftarrow \varnothing$
  $(Const, state) \xleftarrow{\$} \mathcal{A}^{\mathcal{E}}(state)$
  $check \leftarrow \mathsf{true}$
  $R \xleftarrow{\$} \{0,1\}^{Rlen}$
  $state \xleftarrow{\$} \mathcal{A}^{\mathcal{E}, Verify}(R, state)$
**endfor**
**return** $win$

As in the previous model, at each step of the experiment the adversary can query $Verify$ oracle only once. The probability of forgery can be estimated from similar considerations (see section B.5.1). It is necessary to take into account that the adversary receives additional small information when failing an authentication attempt (namely, that $E_k(C \parallel R) \neq y$ for a

given $C$ and $R$). To simplify the evaluation, we will assume that in fact the adversary receives information about the correct answer of $y$ every time it queries $Verify$ oracle.

At the first step we replace every occurence of the block cipher $E_k(\cdot)$ by random permutation $\pi(\cdot)$ (similar to the step described in B.5.1). With such a transition, the term $\mathtt{Adv}^{\mathrm{PRP}}(t, p+\theta)$ arises.

At the second step we estimate the success probability in the following way.

Let $\mathsf{F_i}$ be an event that the adversary successfully forges at $i$-th step, $\mathsf{F} = \cup_i \mathsf{F_i}$. Then:

$$\mathbb{P}[\mathsf{F}] = \mathbb{P}\left[\cup_{i=1}^{\theta} \mathsf{F_i}\right] =$$

$$= \mathbb{P}[\mathsf{F_1}] + \mathbb{P}\left[\mathsf{F_2} \cap \overline{\mathsf{F_1}}\right] + \ldots + \mathbb{P}\left[\mathsf{F}_\theta \cap \overline{\cup_{\mathsf{i}=1}^{\theta-1} \mathsf{F_i}}\right] \leq$$

$$\leq \mathbb{P}[\mathsf{F_1}] + \mathbb{P}\left[\mathsf{F_2} \mid \overline{\mathsf{F_1}}\right] + \ldots + \mathbb{P}\left[\mathsf{F}_\theta \mid \overline{\cup_{\mathsf{i}=1}^{\theta-1} \mathsf{F_i}}\right].$$

Each of the summands of the type $\mathbb{P}\left[\mathsf{F_n} \mid \overline{\cup_{\mathsf{i}=1}^{\mathsf{n}-1} \mathsf{F_i}}\right]$ can be estimated as follows: $\frac{p+\theta}{2^{Rlen}} + \frac{|Consts|}{|\mathrm{Dom}|-p-\theta+1}$, where $p$ is the total number of $\mathcal{E}$ queries at all steps. The first term is responsible for the probability of collision with one of the previous queries, the second term evaluates the probability of randomly guessing the answer under a fresh challenge.

Hence, the following estimate holds for the Chal-$\theta$ model:

$$\mathsf{Adv}^{\text{chal-}\theta}(\mathcal{A}) = \mathbb{P}\left[\mathrm{Exp}^{\text{Chal-}\theta}(\mathcal{A}) \to 1\right] \leq$$

$$\leq \mathtt{Adv}^{\mathrm{PRP}}(t, p+\theta) + \frac{\theta \cdot (p+\theta)}{2^{Rlen}} + \frac{\theta |Consts|}{|\mathrm{Dom}| - p - \theta + 1}.$$

### B.5.4 The case of multiple keys and multiple verifications

A further generalization takes into account a set of $d$ independent keys (similar to the $LOR2_d$ model, see section B.3.2). Let us denote the resulting model by $\mathrm{Chal}_d$. We can (similarly to the previous section B.5.3) split the event $\mathsf{F}$ (the adversary forges for at least one of the $d$ keys) into sub-events $\mathsf{F_i}$ (the adversary forges for the key $k_i$, where $i = 1, \ldots, d$).

We can estimate the probability of $\mathsf{F}$ using the estimates for $\mathsf{F_i}$:

$$\mathtt{Adv}^{\mathrm{Chal}_d}(t, \mathcal{P}, \Theta) = \mathbb{P}[\mathsf{F}] \leq \sum_{i=1}^{d} \mathbb{P}[\mathsf{F_i}] \leq$$

$$\leq \sum_{i=1}^{d} \left( \mathtt{Adv}^{\mathrm{PRP}}(t + T, p_i + \theta_i) + \frac{\theta_i \cdot (p_i + \theta_i)}{2^{Rlen}} + \frac{\theta_i |Consts|}{|\mathrm{Dom}| - p_i - \theta_i + 1} \right),$$

where $\theta_i = \Theta[i]$ is the number of of attempts to successfully authenticate on the $i$-th key, $p_i = \mathcal{P}[i]$ is the total number of $\mathcal{E}$ queries on $i$-th key. The term $T$ is added to the total time of an attack due to the need to simulate $d-1$ different keys (similar to the proof for $LOR2_d$).

## B.6 Intermediate model for authentication with additional data transmission

Let us introduce the intermediate model AUTH′ useful for the proof of security in the full model AUTH$^+$. The model AUTH′ differs from AUTH$^+$ model by $SetMessage$ oracle: in AUTH′ it does not depend on the bit $b$.

The pseudocode of the $SetMessage$ oracle is given below.

$SetMessage(\pi, M)$

$holder = Sessions[\pi].holder$
**if** $(holder = Reader)$
  $state = (Reader.state[\pi], Reader.database)$
**else**
  $state = Tags[holder].state$
**fi**
$state.message = M$
**return** *success*

**Definition 24.** *Let*

$$\mathtt{Adv}_{\Pi}^{\mathrm{AUTH}'}(t, d, \mathcal{P}, \mathcal{Q}, \mathcal{R}, \Theta, \mathcal{M}, \mathcal{N}, \Phi, \Psi, \widehat{\mathcal{Q}}, \widehat{\mathcal{M}}, \widehat{\Phi})$$

*be a maximal advantage* $\mathsf{Adv}_{\Pi}^{\mathrm{auth}'}(\mathcal{A})$, *where the maximum is taken over all adversaries* $\mathcal{A}$ *with the attack time not exceeding* $t$ *and the resourses restrictions as specified in* AUTH$^+$ *model.*

Now we will show that in order to succeed in the AUTH′ model, the adversary must either forge a response $Resp$, or to forge at least one MAC $\tau$.

**Theorem 5.** *The following inequality holds:*

$$\mathtt{Adv}_{\Pi}^{\mathrm{AUTH}'}(t,d,\mathcal{P},\mathcal{Q},\mathcal{R},\Theta,\mathcal{M},\mathcal{N},\Phi,\Psi,\widehat{\mathcal{Q}},\widehat{\mathcal{M}},\widehat{\Phi}) \leq$$

$$\leq 2\cdot\sum_{i=1}^{d}\Big(\mathtt{Adv}^{\mathrm{PRP}}(t+T,p_i+\theta_i)+\frac{\theta_i\cdot(p+\theta_i)}{2^{Rlen}}+\frac{\theta_i|Consts|}{|\mathrm{Dom}|-p_i-\theta_i+1}\Big)+$$

$$+2\cdot\sum_{i=1}^{d}\mathtt{Adv}^{\mathrm{EUF-CMA}}\Big(t+T,r_i+q_i+\widehat{q}_i,\max(\mu_i+2,\widehat{\mu}_i+2,\nu_i+1),$$

$$\phi_i+2\cdot q_i+\widehat{\phi}_i+2\cdot\widehat{q}_i+\psi_i+r_i,\theta_i\Big)$$

*Proof.* Let $\mathcal{A}$ be an adversary in AUTH′ model for authentication protocol $\Pi$. Let $\mathsf{Bad}$ be an event such that $Test^b$ in AUTH′ model answers 1 at least once.

Let us write an advantage of $\mathcal{A}$ in AUTH′ model in the following alternative form:

$$\mathsf{Adv}_{\Pi}^{\mathrm{auth}'}(\mathcal{A})=\mathbb{P}\left[\mathcal{A}^1\to 1\right]-\mathbb{P}\left[\mathcal{A}^0\to 1\right]=$$

$$=2\cdot\mathbb{P}[\mathcal{A}\to b'\colon b'=b]-1=$$

$$2\cdot\mathbb{P}[\mathcal{A}\to b'\colon b'=b\mid\mathsf{Bad}]\cdot\mathbb{P}[\mathsf{Bad}]+2\cdot\mathbb{P}\left[\mathcal{A}\to b'\colon b'=b\mid\overline{\mathsf{Bad}}\right]\cdot\mathbb{P}\left[\overline{\mathsf{Bad}}\right]-1.$$

If the event $\overline{\mathsf{Bad}}$ happens, then $Test^b$ answers 0 on all queries. In that case the resulting bit of the adversary $b'$ is statistically independent of bit $b$, hence the equality $b=b'$ is true with the following probability:

$$\mathbb{P}\left[\mathcal{A}\to b'\colon b'=b\mid\overline{\mathsf{Bad}}\right]=\frac{1}{2}.$$

In that case the advantage can be estimated as follows:

$$\mathsf{Adv}_{\Pi}^{\mathrm{auth}'}(\mathcal{A})\leq 2\cdot\mathbb{P}[\mathsf{Bad}]\,.$$

Now let us estimate the probability of the event $\mathbb{P}[\mathsf{Bad}]$. We can divide the event $\mathsf{Bad}$ into sub-events $\mathsf{Bad}_1$ and $\mathsf{Bad}_2$, where the event $\mathsf{Bad}_1$ is that $Test^1(\pi)$ answers 1 for some session $\pi$ without additional data, the event $\mathsf{Bad}_2$ is that $Test^1(\pi)$ answers 1 for some session $\pi$ with the additional data transfer. In that case we have:

$$\mathbb{P}[\mathsf{Bad}]\leq\mathbb{P}[\mathsf{Bad}_1]+\mathbb{P}[\mathsf{Bad}_2]\,.$$

The oracle $Test^1(\pi)$ answers 1 if and only if $\pi$ is terminated correctly and does not have matching session.

**Sessions without additional data transmission.** We show that in case of a session mode without additional data transmission (i.e. $ProtMode = PLAIN$) for the correct session termination it is required that the adversary constructs a correct $E_k(C \parallel rand)$ for a randomly generated $rand$ at one of the authentication steps (where $rand$ is either $r$ or $R$). In that case the realization of the event $\mathsf{Bad}_1$ in the interaction with some adversary $\mathcal{A}$ leads to the success of the adversary $\mathcal{A}$ in the experiment $\mathrm{Chal}_d$. In each of the cases ($AUTH\_TYPE = \mathrm{TAM}$, the holder is $Reader$; $AUTH\_TYPE = \mathrm{MAM}$, the holder is $Reader$; $AUTH\_TYPE = \mathrm{MAM}$, the holder is Tag $ID$) the adversary must obtain $Resp = E_k(C \parallel rand)$; if it tries to obtain the value in some parallel session, then either $AUTH\_TYPE$, or $ProtMode$, or session holder, or $rand$ differs from the respective values in the session under attack; if they are the same, then the session is automatically becomes partnered (because only those values are used in $Match$ predicate, see Definition 4). The cases above correspond to some queries to the Oracle $\mathcal{O}$ in the model $\mathrm{Chal}_d$, i.e. these queries are valid in the $\mathrm{Chal}_d$ model. If $\mathcal{A}$ successfully forges, then it forges in the $\mathrm{Chal}_d$ model.

Let us also notice that in the situation under consideration we give the adversary more opportunities than it actually has (in real-world scenarios): for example, the adversary cannot start a parallel session with a different $ProtMode$ from the Tag side without finishing the current one.

We see that in all cases discussed above if the adversary succeeds in the $\mathrm{AUTH}'$ model, then it manages to solve the problem in the $\mathrm{Chal}_d$ model. Thus, the probability of the occurrence of the event $\mathsf{Bad}_1$ can be estimated from above by:

$$\mathbb{P}[\mathsf{Bad}_1] \le \mathtt{Adv}^{\mathrm{Chal}_d}(t, \mathcal{P}, \Theta) \le$$
$$\le \sum_{i=1}^{d} \Big( \mathtt{Adv}^{\mathrm{PRP}}(t + T, p_i + \theta_i) + \frac{\theta_i \cdot (p + \theta_i)}{2^{Rlen}} \Big) +$$
$$+ \sum_{i=1}^{d} \Big( \frac{\theta_i |Consts|}{|\mathrm{Dom}| - p_i - \theta_i + 1} \Big).$$

**Sessions with additional data transmission.** In case of a mode with an additional data transmission for the correct session termination it is required that the adversary presents a correctly formed $E_k(C \parallel rand)$ and also a correct MAC value for the entire message. We show that if $\mathsf{Bad}_2$ happens when interacting with some adversary $\mathcal{A}$, then $\mathcal{A}$ is able to succeed in the experiment $\mathrm{EUF} - \mathrm{CMA}_d$.

During the session a message of the form $Sec = Resp \parallel Data \parallel MAC(k^m, Resp \parallel Data)$ is transmitted, where $Data$ is some additional data (**in the cleartext or encrypted one**). Two conditions must be satisfied in order to successfully terminate the session:

– $Resp$ is correctly formed;

– the MAC value $\tau = MAC(k_m, Resp \parallel Data)$ is valid.

Let us consider three separate cases. **Case 1:** TAM with additional data transmission, and holder is Reader.

If the adversary tries to obtain $\tau$ in some parallel session, then we have either $C' \neq C$ (wrong session type $AUTH\_TYPE$, or $ProtMode$, or $holder$), or $R' \neq R$, or $Data \neq Data'$, or $ID' \neq ID$. In all these case either $Resp' \neq Resp$, or $Data' \neq Data$, or $ID' \neq ID$, hence the adversary must forge a tag on a key $k^m_{ID}$ for a fresh message $Resp \parallel Data$.

If all data fields are the same, i.e. $C' = C$, $R' = R$, $Data' = Data$, $ID' = ID$, then due to the fact that MAC is deterministic, we have a session that is partnered to the attacked one (i.e., $Protect$ part is the same for both sessions). Hence, the only nontrivial way in this case to attack a session is to forge a MAC tag for some fresh message.

**Case 2:** MAM with additional data transmission, and holder is Reader. This case differs from the previous one only by an addition of $r$-value in $Resp$.

**Case 3:** MAM with additional data transmission, and holder is some Tag $ID$. This case is special due to the *implicit* binding of $Data$ and $Data'$; hence, it must be treated separately. This type of session ends successfully if and only if $\widehat{Protect}(IResp, Data')$ is valid. The value $IResp$ depends on $r$ generated by the Tag, on the mode constant $C$ and on the Tag's key $k_{ID}$.

If the adversary tries to obtain $\tau$ in some parallel session, and one of the conditions hold (either $C' \neq C$ (wrong session type $AUTH\_TYPE$, or $ProtMode$, or $holder$), or $r' \neq r$, or $Data'' \neq Data'$, or $ID' \neq ID$), then either $IResp' \neq IResp$, or $Data'' \neq Data'$, or $ID' \neq ID$, hence the adversary must forge a tag on a key $k^m_{ID}$ for a fresh message $IResp \parallel Data'$.

Now let us consider the situation when $C' = C$, $r' = r$, $Data'' = Data'$, $ID' = ID$. Thus, the adversary in the parallel session was able to correctly pass the Reader's check of the message $Protect(Resp, Data)$, where $Resp = TResp \parallel r$ (otherwise the session is terminated). Hence, it is true that there is a session $\pi'$ that ends successfully at the Reader side, i.e., this is the previous case (case 2, see above). We have already shown that this situation holds true if and only if the sessions from the Reader's "point of view" and the Tag's

"point of view" are matched, i.e. $Data$ and $R$ are also the same (otherwise the adversary was able to forge MAC-value at the previous step).

But taking into account that MAC is deterministic, we have a session that is partnered to the attacked one (i.e., $Protect$ part is the same for both sessions). Hence, the only nontrivial way in this case to attack a session is to forge a MAC tag for some fresh message.

We see that in all cases discussed above if the adversary succeeds in the AUTH′ model, then it manages to solve the problem in the $\mathrm{EUF}-\mathrm{CMA}_d$ model. Thus, the probability of the occurrence of the event $\mathsf{Bad}_2$ can be estimated from above by:

$$\mathbb{P}[\mathsf{Bad}_2] \le \mathtt{Adv}^{\mathrm{EUF-CMA}_d}(t, \widetilde{\mathcal{R}}, \widetilde{\mathcal{N}}, \widetilde{\Psi}, \Theta) \le$$
$$\le \sum_i \mathtt{Adv}^{\mathrm{EUF-CMA}}(t+T, \widetilde{r_i}, \widetilde{\nu_i}, \widetilde{\psi_i}, \theta_i),$$

where the following notation is used:

– The total number of AE-sessions and MAC-sessions where $ID_i$ is holder or expected partner does not exceed

$$\widetilde{r_i} = r_i + q_i + \widehat{q_i}.$$

– The maximal length of additional data $|Data|$ (in blocks, taking into account $IV$ and $Resp$) in AE-sessions and MAC-sessions where $ID_i$ is holder or expected partner does not exceed

$$\widetilde{\nu_i} = \max(\mu_i + 2, \widehat{\mu}_i + 2, \nu_i + 1).$$

– The total length of additional data $\sum |Data|$ (in blocks, taking into account $IV$ and $Resp$) in AE-sessions and MAC-sessions where $ID_i$ is holder or expected partner does not exceed

$$\widetilde{\psi_i} = \phi_i + 2 \cdot q_i + \widehat{\phi}_i + 2 \cdot \widehat{q}_i + \psi_i + r_i.$$

Now the theorem follows from two estimates on the probability of event $\mathsf{Bad}$. □

# C Security proofs for TAM, IAM, MAM in $\mathrm{AUTH}^+$ model

**Theorem 6.** *The following inequality holds:*

$$\mathtt{Adv}_{\Pi}^{\mathrm{AUTH}^+}(t, d, \mathcal{P}, \mathcal{Q}, \mathcal{R}, \Theta, \mathcal{M}, \mathcal{N}, \Phi, \Psi, \widehat{\mathcal{Q}}, \widehat{\mathcal{M}}, \widehat{\Phi}) \le$$

$$\leq \sum_{i=1}^{d} \mathtt{Adv}^{LOR2}\Big(t+T, q_i, \mu_i, \phi_i, \widehat{q}_i, \widehat{\mu}_i, \widehat{\phi}_i\Big) +$$

$$+2 \cdot \sum_{i=1}^{d} \mathtt{Adv}^{\mathrm{EUF-CMA}}\Big(t+T, r_i+q_i+\widehat{q}_i, \max\left(\mu_i+2, \widehat{\mu}_i+2, \nu_i+1\right),$$

$$\phi_i + 2\cdot q_i + \widehat{\phi}_i + 2\cdot \widehat{q}_i + \psi_i + r_i, \theta_i\Big) +$$

$$+2 \cdot \sum_{i=1}^{d} \Big(\mathtt{Adv}^{\mathrm{PRP}}(t+T, p_i+\theta_i) + \frac{\theta_i \cdot (p_i+\theta_i)}{2^{Rlen}} + \frac{\theta_i |Consts|}{|\mathrm{Dom}| - p_i - \theta_i + 1}\Big).$$

*Proof.* Let $\mathcal{A}^{b_1 b_2}$ be an adversary interacting with a following oracles:

– $CreateTag, StartReaderSession, StartTagSession, Send, Result$;

– $SetMessage^{b_1}, Test^{b_2}$.

Using this notation we can write:

$$\mathsf{Adv}_{\Pi}^{\mathrm{auth}^+}(\mathcal{A}) = \mathbb{P}\left[\mathcal{A}^{11} \to 1\right] - \mathbb{P}\left[\mathcal{A}^{00} \to 1\right] =$$
$$\left(\mathbb{P}\left[\mathcal{A}^{11} \to 1\right] - \mathbb{P}\left[\mathcal{A}^{10} \to 1\right]\right) + \left(\mathbb{P}\left[\mathcal{A}^{10} \to 1\right] - \mathbb{P}\left[\mathcal{A}^{00} \to 1\right]\right).$$

Let us estimate the second term.

Since the bit $b_2$ is fixed and equals 0, the oracle $Test^0$ becomes trivial (gives 0 to any request of the adversary), which means it does not give any additional information to the adversary $\mathcal{A}$. If there is an adversary $\mathcal{A}$, for which the term equals $\varepsilon$, then it is possible to construct another adversary $\mathcal{B}$, who uses $\mathcal{A}$ as a subroutine and achieves the same advantage $\varepsilon$ in the $LOR2_d$ model.

In the $LOR2_d$ model (see section B.3.2 for more details) the adversary $\mathcal{B}$ is given access to two oracles $\mathrm{LOR}_d^b$ and $\widehat{\mathrm{LOR}_d}^b$, and $\mathcal{B}$ must guess the bit $b$. The adversary $\mathcal{B}$ simulates all conditions of the $\mathrm{AUTH}^+$ experiment for $\mathcal{A}$, except for the $SetMessage^b$ oracle. Queries to the $SetMessage^b$ oracle are redirected by $\mathcal{B}$ to $\mathrm{LOR}_d^b$ and $\widehat{\mathrm{LOR}_d}^b$ oracles.

Let us note that $\mathcal{B}$ is able to simulate the experiment for $\mathcal{A}$, even though he does not posess secret keys $k_i$, that are fixed inside $\mathrm{LOR}_d^b$ and $\widehat{\mathrm{LOR}_d}^b$ oracles. At the beginning of the experiment $\mathcal{B}$ initialize arrays $Sessions$, $Tags$, and $Reader$ data structure.

– When $\mathcal{A}$ queries $CreateUser(ID)$ oracle, $\mathcal{B}$ generates two keys $k$ and $k^m$ for the tag $ID$ and stores them as well as initial inner state of the tag. The third key $k^e$ of $ID$ is already fixed inside $\mathrm{LOR}_d^b$ and $\widehat{\mathrm{LOR}_d}^b$ oracles. All keys are independent.

– The adversary $\mathcal{B}$ can generate values $TResp$, $IResp$ according to the TAM, IAM, MAM protocols and check values from $\mathcal{A}$, since for generating these values (and checking them) only the key $k$ is needed.

– The adversary $\mathcal{B}$ can generate tag $MAC(k^m, M)$ for any chosen $M$ and verify tags, since for these actions only the knowledge of $k^m$ is needed.

– When $\mathcal{B}$ needs to simulate data transmission from $SetMessage^b$ query (when the protection mode is encrypt), it sends the pair $(M_0, M_1)$ to the appropriate oracle ($\mathrm{LOR}_d^b(i, M_0, M_1)$ if $SetMessage$ was queried for the session, whose holder is $ID_i$; $\widehat{\mathrm{LOR}_d}^b(i, M_0, M_1)$, if $SetMessage$ was queried for the session, whose partner is $ID_i$).

– When $\mathcal{A}$ queries $Test^0$ oracle, $\mathcal{B}$ returns 0.

Attack time for $\mathcal{B}$ is equal to the $\mathcal{A}$-time plus the time for simulating the protocol messages (no more than $T$). The adversary $\mathcal{B}$ returns the same bit as $\mathcal{A}$.

From the considerations above one can see that the following inequality holds:

$$\mathsf{Adv}_{\mathcal{SE}}^{\mathrm{lor2d}}(\mathcal{B}) = \mathbb{P}\left[\mathcal{B}^1 \to 1\right] - \mathbb{P}\left[\mathcal{B}^0 \to 1\right] = \\ = \mathbb{P}\left[\mathcal{A}^{10} \to 1\right] - \mathbb{P}\left[\mathcal{A}^{00} \to 1\right].$$

Let us now estimate the first term:

$$\mathbb{P}\left[\mathcal{A}^{11} \to 1\right] - \mathbb{P}\left[\mathcal{A}^{10} \to 1\right].$$

In this case we simulate the $SetMessage^1$ oracle. Since the bit $b$ in the oracle $setMessage^b$ is fixed, it does not give information to the adversary $\mathcal{A}$. Let us construct $\mathcal{B}$ adversary in the AUTH′ model (see section B.6), who uses $\mathcal{A}$ as a subroutine. Given the query $(M_0, M_1)$ from $\mathcal{A}$ to $SetMessage^1$ oracle, the adversary $\mathcal{B}$ «chooses» message $M_1$ and feeds it as an input to his own $SetMessage$ oracle. All other queries from $\mathcal{A}$ the adversary $\mathcal{B}$ retranslates without any change to his own oracles.

The advantage of $\mathcal{B}$ in the AUTH′ model equals:

$$\mathsf{Adv}^{\mathrm{auth}'}(\mathcal{B}) = \mathbb{P}\left[\mathcal{B}^1 \to 1\right] - \mathbb{P}\left[\mathcal{B}^0 \to 1\right] = \\ \mathbb{P}\left[\mathcal{A}^{11} \to 1\right] - \mathbb{P}\left[\mathcal{A}^{10} \to 1\right].$$

Hence, the following estimate holds:

$$\mathsf{Adv}_{\Pi}^{\mathrm{auth}^+}(\mathcal{A}) \leq$$

$$\mathtt{Adv}_{\mathcal{SE}}^{LOR2_d}(t+T,\mathcal{Q},\mathcal{M},\Phi,\widehat{\mathcal{Q}},\widehat{\mathcal{M}},\widehat{\Phi})+$$
$$+\mathtt{Adv}_{\Pi}^{\mathrm{AUTH}'}(t,d,\mathcal{P},\mathcal{Q},\mathcal{R},\Theta,\mathcal{M},\mathcal{N},\Phi,\Psi,\widehat{\mathcal{Q}},\widehat{\mathcal{M}},\widehat{\Phi}).$$

Using this and previously obtained estimate (for $\mathtt{Adv}^{LOR2_d}$ and $\mathtt{Adv}^{\mathrm{AUTH}'}$) we obtain the final result. □

# Symmetric Cryptography
# Linear transformations

# Circulant matrices over $\mathbb{F}_2$ and their use for construction efficient linear transformations with high branch number

Stepan Davydov and Yuri Shkuratov

JSRPC «Kryptonite», Moscow, Russia
s.davydov@kryptonite.ru, y.shkuratov@kryptonite.ru

**Abstract**

Linear transformations over $\mathbb{F}_2$ with high branch number are studied in this work. New class of transformations is proposed – transformations, which correspond multiplication in the ring $\mathbb{F}_2[x]/f(x)$. An important subclass of this class is studied in detail – transformations defined by circulant matrices over $\mathbb{F}_2$. It is proposed to decompose any matrix as sum of products diagonal matrices and circulant matrices over $\mathbb{F}_2$, which allows to offer new software implementations of linear transformations. AES and Whirlpool matrices have been decomposed using this method.

**Keywords:** Linear transformation, branch number, MDS matrix, circulant matrix, matrix decomposition, software implementation of linear transformation.

## 1 Introduction

As Claude Shennon declared in "Communication Theory of Secrecy Systems" [1], transformations used in ciphers and hash functions should provide confusion and diffusion of the input data. Linear transformations are mostly used to provide diffusion properties. High branch numbers of the linear transformation matrix and its transpose are needed to protect against the differential [2] and the linear [3], [4] methods of cryptanalisys.

To date some theoretical methods to construct maximum distance separable (further MDS) matrices over $\mathbb{F}_{2^s}$ are known (see [5]). Namely, Cauchy matrices (used in Streebog hash function [6]), Vandermonde matrices, recursive (also named serial [7]) matrices (used in PHOTON hash function [7], Kuznyechik block cipher [8]), Hadamard matrices etc.

Another way to construct MDS matrices is to seek among matrices of a certain class. Since circulant matrices have some same submatrices, seeking methods are more efficient among them [5]. MDS circulant matrices are used in AES cipher [9], SM4 cipher [10], Whirlpool hash function [11] etc.

Hardware and software implementations efficiency is also an importatnt property of the linear transformations. Software implementation is usually reduced to the set (extended set) of the processor instructions usage [12]. Special processor instructions sets usage (for example, AES NI [13]) is a good variant for the implementation efficiency. Precalculated tables (see LUT-tables, [14]) is another good way but some amount of fast memory is needed. If we don't have the required amount of memory, the question of the efficient implementation is actual.

Linear transformations, which correspond multiplication in the ring $\mathbb{F}_2[x]/f(x)$, are studied in this work. This class generalize the class of circulant matrices over $\mathbb{F}_2$. Software implementation of such transformations requires small amount of memory, much less then LUT-tables (see Statement 2). It is not claimed strictly, but authors consider that such transformations may have fast software implementation, since CLMUL instruction set [15] may be used to implement them (see Statement 2). Verification of this fact is not included in this article, it may be in further studies.

Also, it is proposed to decompose any matrix as sum of products diagonal matrices and circulant matrices over $\mathbb{F}_2$, which allows to offer new software implementations of linear transformations. It is prooved that if decomposed matrix is circulant over $\mathbb{F}_{2^s}$, its decomposition has not more then $s$ summands. AES and Whirlpool matrices have been decomposed using this method.

## 2 Definitions and preliminaries

The vector coordinates are numbered from right to left, the matrix rows are numbered from bottom to top.

**Definition 1.** *Transformation $\varphi$ is called involutory (involution) over $M$ if for any $m \in M$ the equation $\varphi(\varphi(m)) = m$ holds. In other words, $\varphi = \varphi^{-1}$.*

Let $\mathbb{F}_{q^s}$ denote the finite field of $q^s$ elements.

Let $Q$ be some field. Let $\overrightarrow{E}_i$ denote the vector $(0, ..., 0, 1, 0, ..., 0) \in Q^m$ with 1 on the $i$-th place. Identity matrix is equal to

$$E_{m\times m} = \begin{pmatrix} \overrightarrow{E}_{m-1} \\ \overrightarrow{E}_{m-2} \\ \dots \\ \overrightarrow{E}_0 \end{pmatrix}.$$

**Definition 2.** *Let $T$ denote the following matrix:*

$$T_{m\times m} = \begin{pmatrix} \overrightarrow{E}_0 \\ \overrightarrow{E}_1 \\ ... \\ \overrightarrow{E}_{m-1} \end{pmatrix}. \tag{1}$$

Matrix $T$ has the next properties.

**Statement 1.** *1. $T$ is involutory matrix, $T = T^{-1}$.*

*2. Product $TA$ has the opposite to $A$ order of rows, $\overrightarrow{A}_i = \overrightarrow{(TA)}_{m-1-i}$.*

*3. Product $AT$ has the opposite to $A$ order of columns, $A_i^{\downarrow} = (TA)_{m-1-i}^{\downarrow}$.*

*4. Product $TAT$ has the opposite to $A$ order of rows and columns, i. e. $a_{i,j} = (tat)_{m-1-i,m-1-j}$, where $tat$ is the corresponding element of $TAT$ matrix.*

**Definition 3.** *The weight of $\overrightarrow{a} \in Q^m$, denoted $wt(\overrightarrow{a})$, is the number of nonzero coordinates of $\overrightarrow{a}$.*

**Definition 4.** *[5] Branch number of matrix $A \in Q_{m,m}$ is the following number:*

$$\tau(A) = \min_{\overrightarrow{a}\neq\overrightarrow{0}} [wt(\overrightarrow{a}) + wt(\overrightarrow{a}A)].$$

It is obviously that $\tau(A) = \tau(A^{-1})$ and $\tau(A) \leq m + 1$.

**Definition 5.** *[5] If $\tau(A) = m + 1$, $A$ is Maximum Distance Separable (further MDS) matrix.*

**Definition 6.** *$D_{n\times n}$ is diagonal block matrix if:*

$$D_{n\times n} = diag_{m\times m}(A_{m-1}, ..., A_0) = \begin{pmatrix} A_{m-1} & O_{s\times s} & ... & O_{s\times s} \\ O_{s\times s} & A_{m-2} & ... & O_{s\times s} \\ ... & & & \\ O_{s\times s} & O_{s\times s} & ... & A_0 \end{pmatrix}_{m\times m},$$

*where $O_{s\times s}$ is null matrix, $O_{s\times s}, A_i \in Q_{s\times s}$, $n = ms$. If $s = 1$, $D$ is diagonal matrix.*

**Definition 7.** *Matrix $U_{n\times n}$ is permutation block matrix if:*

$$U_{n\times n} = \begin{pmatrix} ... & O_{s\times s} & ... & E_{s\times s} & ... & O_{s\times s} & ... & O_{s\times s} \\ ... & O_{s\times s} & ... & O_{s\times s} & ... & E_{s\times s} & ... & O_{s\times s} \\ ... & ... & ... & ... & ... & ... & ... & ... \\ ... & E_{s\times s} & ... & O_{s\times s} & ... & O_{s\times s} & ... & O_{s\times s} \\ ... & ... & ... & ... & ... & ... & ... & ... \end{pmatrix}_{m\times m},$$

*where $O_{s\times s}, E_{s\times s} \in Q_{s,s}, n = ms$, and every row and column contain exactly one identity matrix $E_{s\times s}$. If $s = 1$, $U$ is permutation matrix.*

Matrix $T$ from Definition 2 is permutation matrix.

Since multiplying any vector by nonsingular diagonal matrix $D$ and permutation matrix $U$ doesn't change it weight, the branch number of any matrix doesn't change after left or right multiplying by matrices $D$ and $U$.

**Definition 8.** *Matrix $C_{m\times m}$ over $Q = \mathbb{F}_{q^s}$ is circulant matrix if every row $\overrightarrow{C_i}$ is the left rotation of the previous row $\overrightarrow{C_{i-1}}, i \in \overline{1, m-1}$.*

$$C_{m\times m} = Circ_{q^s}(c_{m-1}, ..., c_0) = \begin{pmatrix} c_0 & c_{m-1} & ... & c_2 & c_1 \\ c_1 & c_0 & ... & c_3 & c_2 \\ ... & & & & \\ c_{m-2} & c_{m-3} & ... & c_0 & c_{m-1} \\ c_{m-1} & c_{m-2} & ... & c_1 & c_0 \end{pmatrix}$$

**Remark 1.** *Any row in circulant matrix defines other rows, therefore it is correct to say "circulant matrix $C$ is defined by row $\overrightarrow{A_i}$ of matrix $A$", which means $\overrightarrow{A_i} = \overrightarrow{C_i}$ and other rows of matrix $C$ are defined by row $\overrightarrow{C_i}$.*

**Remark 2.** *Let $P = \mathbb{F}_2$ be the field of two elements, $P_n[x] = P[x]/f(x)$ be the polynomial ring over $P$ with addition and multiplication modulo $f(x)$. Note that $P_n[x]$ is vector space of dimention $n$ over $P$. There exist isomorphic mapping between $P^n$ and $P_n[x]$: $\varphi(a_{n-1}, ..., a_1, a_0) = a_{n-1}x^{n-1} + ... + a_1x + a_0$. Further we will equate vector rows of length $n$ with corresponding polynomials from $P_n[x]$.*

**Definition 9.** *Let $P = \mathbb{F}_2$ be the field of two elements, $Q = (P[x]/g(x), +, \cdot)$ and $g(x)$ be irreducible polynomial of degree $s$ over $P$. The field $Q$ is isomorhic to field $\mathbb{F}_{2^s}$. Let $B_{m\times m}$ be a matrix over $Q$, which transforms vectors from $Q^m$. Since elements of $Q$ are vector rows over $P$, it is possible to consider $B$ as linear transformation of vector rows of length $n = ms$ over $P$ and there exist corresponding matrix $A_{n\times n}$ over $P$.*

*In such case we said: matrix $A = A(B, g(x))$ implements linear transformation $B$ on binary vectors.*

Let $\overrightarrow{a} \in P^{ms}$. We split $\overrightarrow{a}$ into *s-subvectors*, which are usually given to S-boxes: subvector $\overrightarrow{a}(i, s)$ with number $i$ is subvector of length $s$ equal to $(a_{(i+1)s-1}, a_{(i+1)s-2}, ..., a_{is}), i \in \{0, ..., m-1\}$. Then

$$\overrightarrow{a} = (\overrightarrow{a}(m-1, s), ..., \overrightarrow{a}(0, s)).$$

**Definition 10.** *S-weight of vector $\overrightarrow{a} \in P^{ms}$, denoted $wt_s(\overrightarrow{a})$, is the number of nonzero s-subvectors of vector $\overrightarrow{a}$.*

**Definition 11.** *Branch number on s-subvectors of matrix $A \in P_{ms,ms}$ is the following number:*

$$\tau_s(A) = \min_{\overrightarrow{a} \in P^{ms} \setminus \overrightarrow{0}} [wt_s(\overrightarrow{a}) + wt_s(\overrightarrow{a}A)].$$

**Remark 3.** *Let $B_{m\times m}$ be the matrix over $Q \cong \mathbb{F}_{2^s}$ and $A_{n\times n}$ implements transformation $B$ on binary vectors, $A = A(B, g(x))$ for some $g(x)$. Then the branch number of matrix $B$ over $Q$ is equal to branch number on s-subvectors of matrix $A$ over $\mathbb{F}_2$.*

**Remark 4.** *Same to matrices over $\mathbb{F}_{2^s}$, the branch number on s-subvectors of any matrix over $\mathbb{F}_2$ does not change after left or right multiplying by nonsingular diagonal block matrix and permutation block matrix because such multiplying does not change s-weight of any vector.*

## 3 Linear transformations, which correspond multiplication in the ring $\mathbb{F}_2[x]/f(x)$

We denote $P = \mathbb{F}_2$. Let us consider the following operations on bit strings, which are implemented on computers as a processor instructions:

1. *XOR($\overrightarrow{\alpha}$, $\overrightarrow{\beta}$)* is bitwise addition of strings modulo 2. This is an analogue of the operation of adding vectors of the same length over $P$.
2. *AND($\overrightarrow{\alpha}$, $\overrightarrow{\beta}$)* is bitwise conjunction of strings. This is an analogue of the operation of multiplying a vector by a diagonal matrix: $AND(\overrightarrow{\alpha}, \overrightarrow{\beta}) = \overrightarrow{\alpha} \cdot diag_{n\times n}(\overrightarrow{\beta})$.
3. *OR($\overrightarrow{\alpha}$, $\overrightarrow{\beta}$)* is bitwise disjunction of strings.
4. *SHFT($\overrightarrow{\alpha}$)* is left (right) shift of the string by $i$ positions with zero padding. This is an analogue of multiplication by matrix with ones on the diagonal, which is below (above) the main one by $i$ positions, and with zero other elements.
5. *CLMUL($\overrightarrow{\alpha}$, $\overrightarrow{\beta}$)* is multiplication of binary strings of length $n$ as polynomials of degree $n-1$ over the field $P$. The result is a string of length $2n$.

**Definition 12.** *Let $f(x)$ be a polynomial of degree $n$ over $P$. Linear transformation, which corresponds multiplication by an element $a(x)$ of the ring $R = P[x]/f(x)$, is the following transformation:*

$$\widehat{a}_{f(x)} : h(x) \to h(x)a(x) \bmod f(x),\ h(x) \in R$$

In view of the Remark 2, this transformation is equivalent to transformation of vectors of length $n$.

The linear transformation matrix has the form:

$$A_{a(x),f(x)} = \begin{pmatrix} \widehat{a}_{f(x)}(x^{n-1}) \\ \dots \\ \widehat{a}_{f(x)}(x^i) \\ \dots \\ \widehat{a}_{f(x)}(x) \\ \widehat{a}_{f(x)}(1) \end{pmatrix} = \begin{pmatrix} a(x)\cdot x^{n-1} \bmod f(x) \\ \dots \\ a(x)\cdot x^i \bmod f(x) \\ \dots \\ a(x)\cdot x \bmod f(x) \\ a(x) \end{pmatrix} \tag{2}$$

It is easy to see that the set of matrices $\{A_{a(x),f(x)},\ a(x) \in R\}$ with the operations of matrix addition and multiplication is a ring isomorphic to the polynomial ring $R$.

**Remark 5.** *For polynomials $a(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \dots + a_0$ and $f(x) = x^n + f_{n-1}x^{n-1} + \dots + f_0$, the vector of coefficients of the polynomial $a(x)\cdot x \bmod f(x)$ has the form:*

$$(a_{n-2} + a_{n-1}f_{n-1},\ \dots,\ a_{i-1} + a_{n-1}f_i,\ \dots,\ a_{n-1}f_0) \tag{3}$$

We show that in some cases the transformation $\widehat{a}_{f(x)}$ can be represented using a small number of processor instructions.

**Statement 2.** *Let $f(x) = x^n + f_{n-1}x^{n-1} + \dots + f_0 = x^n + \overline{f(x)}$ be a polynomial of degree $n$ over $P$, $a(x)$ be a polynomial of degree less than $n$ over $P$. Then the following statements are true for the transformation $\widehat{a} = \widehat{a}_{f(x)}$:*

1. *If deg $\overline{f(x)} \le n/2$, then transformation $\widehat{a}$ can be implemented in five processor instructions: 3 $CLMUL$ + 2 $XOR$.*
2. *If deg $\overline{f(x)}$ + deg $a(x) \le n$, then transformation $\widehat{a}$ can be implemented in three processor instructions: 2 $CLMUL$ + 1 $XOR$.*
3. *If deg $\overline{f(x)} = 0$, then transformation $\widehat{a}$ can be implemented in two processor instructions: 1 $CLMUL$ + 1 $XOR$.*

*4. To implement the transformation $\widehat{a}$, it is necessary to store the polynomials $a(x)$ and $\overline{f(x)}$ in memory in cases 1-2, and only the polynomial $a(x)$ in case 3.*

□ Point 4 is obvious, let us justify points 1-3. We execute the $CLMUL$ instruction for the polynomials $a(x)$ and $h(x)$ and write the result as $b(x) = b_1(x)x^n + b_0(x)$, where the degrees of the polynomials $b_i(x)$ are less than $n$. To obtain the result of a linear transformation, it is necessary to reduce the polynomial $b(x)$ modulo $f(x)$.

1. If $deg\ \overline{f(x)} = 0$, $f(x) = x^n + 1$ (point 3), then $b(x) \bmod f(x) = b(x) + b_1(x)f(x) = b(x) + b_1(x)x^n + b_1(x) = b_1(x) + b_0(x)$. Thus, in addition to the $CLMUL$ instruction, we used 1 $XOR$ instruction.

2. If $deg\ \overline{f(x)} > 0$, then the modulo reduction is a little more complicated. $b(x) \bmod f(x) = b(x) + b_1(x)x^n + b_1(x)\overline{f(x)} \bmod f(x) = b_1(x)\overline{f(x)} + b_0(x) \bmod f(x)$.

   Let $b_1(x)\overline{f(x)} = c(x) = c_1(x)x^n + c_0(x)$, where $deg\ c_0(x) < n$ and $deg\ c_1(x) < n$.

– If $deg\ \overline{f(x)} + deg\ a(x) \le n$, then $deg\ c(x) = deg\ b_1(x) + deg\ \overline{f(x)} = deg\ a(x) + deg\ h(x) + deg\ \overline{f(x)} - n \le deg\ h(x) < n$. So, $b(x) \bmod f(x) = b_0(x) + c_0(x)$, where $c_0(x) = b_1(x)\overline{f(x)}$ can be found in 1 $CLMUL$ instruction. This means that point 2 of the original statement is true.

– If $deg\ \overline{f(x)} \le n/2$, then $deg\ c_1(x) = deg\ b_1(x) + deg\ \overline{f(x)} - n < n/2$ and $deg\ c_1(x) + deg\ \overline{f(x)} < n$. Then $b(x) \bmod f(x) = c_1(x)x^n + c_0(x) + b_0(x) \bmod f(x) = c_1(x)x^n + c_0(x) + b_0(x) + c_1(x)f(x) = c_0(x) + b_0(x) + c_1(x)\overline{f(x)}$. $c_1(x)\overline{f(x)}$ can be computed with 1 $CLMUL$ instruction. In total, to implement a linear transformation, 3 $CLMUL$ and 2 $XOR$ instructions are required. ■

**Remark 6.** *Note that the speed of execution of processor instructions is different, so the number of instructions required to implement the transformation is not a metric of the speed of the transformation. To obtain accurate results, it is necessary to conduct a series of experiments on computers.*

The greatest efficiency of the transformation $\widehat{a}$ is achieved in case 3: the implementation requires only 2 processor instructions, and only the polynomial $a(x)$ needs to be stored in memory. Let us consider other properties of the transformation $\widehat{a}_{x^n+1}$.

**Statement 3.** *Let $f(x) = x^n + 1$ be a polynomial over $P$, $\widehat{a} = \widehat{a}_{f(x)}$. Then:*

1. *Matrix of the linear transformation $\widehat{a}$ is circulant matrix over $P$.*
2. *Branch numbers on s-subvectors of the matrices $A$ and $A^T$ are the same.*
3. *If $n$ is even and the transformation $\widehat{a}$ is an involution, then for any $s \geq 1$ the branch number on s-subvectors of matrix $A_{a(x),f(x)}$ does not exceed 4.*

□ 1. To verify the first point of the statement, it is enough to substitute the polynomial $x^n + 1$ into the formulas (2) and (3).

2. We show that the circulant matrix $A$ satisfies the equality $A^T = TAT$ (the matrix $T$ is defined in (1)). We assume that $i \geq j$, the opposite case is considered similarly. Since the matrix $A^T$ is also a circulant, we have: $a^T_{i,j} = a_{j,i} = a_{0,i-j}$. For the matrix $TAT$ : $tat_{i,j} = a_{m-1-i,m-1-j} = a_{0,m-1-j-(m-1-i)} = a_{0,i-j}$. Since the matrix $T$ is the product of a diagonal block matrix and a permutation block matrix, the branch numbers of the matrices $A$ and $A^T$ are the same.

3. Let $\widehat{a}$ be an involution, $n = 2k$. Then $a(x)^2 \equiv 1 \bmod (x^{2k} + 1)$. This means that $a(x)^2 = 1 + (x^{2k} + 1)t(x)$ is true for some polynomial $t(x)$. So, $(a(x) + 1)^2 = (x^{2k} + 1)t(x)$. So, $t(x)$ is a square, and for some $t_1(x)$ $t(x) = t_1(x)^2$. Then we have $a(x)+1 = (x^k+1)t_1(x)$. Let us consider the action of the transformation $\widehat{a}$ on the polynomial $x^k+1$: $(x^k+1)a(x) = (x^k + 1)((x^k + 1)t_1(x) + 1) = (x^k + 1)(x^k + 1)(t_1(x)) + (x^k + 1) \equiv (x^k + 1) \bmod x^{2k} + 1$. That is, the polynomial $x^k + 1$ corresponding to the vector of weight 2 is mapped onto itself, and the branch number of matrix of the linear transformation $\widehat{a}$ does not exceed 4. ■

Thanks to the first point of the Statement 3, it becomes possible to quickly check the branch number of $A_{a(x),x^n+1}$ by checking the rank of its submatrices [5]. The second point means that for circulant matrices the branch numbers of the matrices $A$, $A^{-1}$, $A^T$, $(A^T)^{-1}$ are the same, which is important for security schemes to differential and linear methods of cryptanalysis [2], [4]. The third point means that among the transformations of this type there are no involutive transformations with a high branch number.

Transformations with the following branch numbers on s-subvectors were found by enumeration on computers among transformations of the form $A_{a(x),x^n+1}$ (see Table 1).

Table 1: Maximum found branch number.

| Matrix size \ s-subvector size | 4-bit | 6-bit | 8-bit |
|---|---|---|---|
| $4 \times 4$ | 5 (MDS) | 5 (MDS) | 5 (MDS) |
| $6 \times 6$ | 6 | 6 | 6 |
| $8 \times 8$ | 7 | - | 8 |
| $16 \times 16$ | 12 | - | - |

## 4 Matrix decomposition into a sum of matrices $A_{a(x),f(x)}$

Since $8 \times 8$ MDS matrices have not been found among matrices $A_{a(x),f(x)}$ over $P = \mathbb{F}_2$, it is necessary to consider matrices with less efficient implementation.

Let $A \in P_{n\times n}$, $f(x) = x^n + f_{n-1}x^{n-1} + ... + f_1 x + 1$ be polynomial over $P$, $a_i(x)$ be polynomials over $P$ of degree less than $n$, $i \in \overline{1,t}$.

We consider the following decomposition:

$$A = \sum_{i=1}^{t} D_i A_i, \tag{4}$$

where $D_i = diag_{n\times n}(d_{i,n-1}, ..., d_{i,0}), d_{i,j} \in \{0,1\}, A_i = A_{a_i(x),f(x)}$ are $n \times n$ matrices over $P$ defined in (2). Note that decomposition (4) for matrix $A$ depends on choise of $f(x)$.

Multiplication by matrices $D_i$ is implemented by instruction $AND$, by matrices $A_i$ – according Statement 2. Sum is implemented by instruction $XOR$. A small amount of summands in (4) is needed for efficient implementation of matrix $A$. Since Statement 2 is true, the next statement holds:

**Statement 4.** *Let decomposition (4) holds for matrix $A$ and polynomial $x^n+1$. Then multiplication by matrix $A$ can be implenented by $t$ instructions $AND$, $t$ instructions $CLMUL$ and $2t-1$ instructions $XOR$.*

**Definition 13.** *Let $Rev_{f(x)} : P_{n,n} \to P_{n,n}$ be transformation, which result on matrix $A$ is matrix $B$ such as every row $\overrightarrow{B_i} = \overrightarrow{A_i} \cdot A^{-i}_{x,f(x)}$. In other words, every row $\overrightarrow{B_i}$ is $i$ times transformed by $\Delta$ $i-th$ row of matrix $A$, where $\Delta$ is inverse transformation to polynomial multiplication by $x$ modulo $f(x)$. The transformation $\Delta$ is exist because $f_0 = 1$.*

**Theorem 1.** *The minimum number of summands $t$ in sum (4) is equal to rank of the matrix $B = Rev_{f(x)}(A)$.*

□ Since transformation $\Delta$ (see Definition 13) is distributive over $XOR$ and commutative with $AND$, the following equalities are equivalent to (4):

$$Rev_{f(x)}(A) = Rev_{f(x)}(\sum_{i=1}^{t} D_i A_i) = \sum_{i=1}^{t} Rev_{f(x)}(D_i A_i) = \sum_{i=1}^{t} D_i Rev_{f(x)}(A_i) \tag{5}$$

Since $A_i$ are matrices with representation (2), all rows in matrix $Rev_{f(x)}(A_i)$ are equal to the $0-th$ row of matrix $A_i$. Then equality (5) is equivalent to the following equalities:

$$\overrightarrow{B_j} = \sum_{i=1}^{t} d_{i,j} \overrightarrow{A_{i,0}}, \quad j \in \overline{1,n}$$

Then equality (5) is equivalent to fact that any row of the matrix $B = Rev_{f(x)}(A)$ is linear combination of the set of rows $\overrightarrow{A_{i,0}}$, $i \in \overline{1,t}$ and $t$ is equal to rank of the matrix $B$. ■

Theorem 1 allows to find the minimum number of summands in sum (4) in case we know $f(x)$.

Let $\overrightarrow{a}$ be a vector of length $n$ and $\alpha \in \{0,1\}$. Let $(\alpha \parallel \overrightarrow{a})$ and $(\overrightarrow{a} \parallel \alpha)$ denote the vectors of length $n+1$, which equal to concatenation of vector $\overrightarrow{a}$ and element $\alpha$.

Let us $A \in P_{n \times n}$. We consider the set of the vectors:

$$\overrightarrow{\Omega_j} = (\overrightarrow{A_j} \parallel 0) + (0 \parallel \overrightarrow{A_{j+1}}), j \in \overline{0, n-2}$$

of length $n+1$ over $P$. Due to (4) we obtain vector $\overrightarrow{\Omega_j}$ is equal:

$$\overrightarrow{\Omega_j} = \sum_{i=1}^{t} d_{i,j}(\overrightarrow{A_{i,j}} \parallel 0) + \sum_{i=1}^{t} d_{i,j+1}(0 \parallel \overrightarrow{A_{i,j+1}}) \tag{6}$$

Let us polynomial $f(x)$ of degree $n$ is fixed, coefficient $f_0 = 1$. Decomposition (4) is defined by coefficients $d_{i,j}$ and coefficients of the polynomials $a_i(x)$.

We define the probability space $\Theta$: let all coefficients be mutually independent random variables with a uniform distribution on $P = \mathbb{F}_2$. In case of probability space $\Theta$ matrix $A$ is random matrix defined by decomposition (4). We construct probability relations for rows of the matrix $A$ in Theorem 2. In particular, we show that if $t$ is a little number, then vector $\overrightarrow{\Omega_j}$ (see (6)) is equal to $\overrightarrow{f}$ with significantly higher probability than random vector of length $n+1$. First, we prove support lemma.

**Lemma 1.** *Let $f(x) = x^n + f_{n-1}x^{n-1} + ... + f_0$ be a polynomial of degree $n$ over $P$ with coefficient $f_0 = 1$. Let $a(x)$ be a random polynomial over $P$ with degree not more than $n-1$ with uniformly distributed coefficients on $P$ ($a_i \sim U\{0;1\}$). Then any element of matrix $A_{a(x),f(x)}$ is also uniformly distributed on $P$.*

□ Let $a_{i,j}$ be an element of the matrix $A_{a(x),f(x)}$. Using formula (2) we give a proof by induction on $i$ – the number of the row, which contains the element $a_{i,j}$.

– If $i = 0$, coefficients of the $0-th$ row are defined by polynomial $a(x)$, each of them has uniform distribution on $P = \mathbb{F}_2$.

– Let Lemma be true for all $i \leq k < n-1$ for some $k$.

– We will evaluate $a_{k+1,j}$. If $j = 0$, then according to $f_0 = 1$, $\Pr(a_{k+1,j} = 1) = \Pr(a_{k,n-1} = 1) = \frac{1}{2}$ by induction hypothesis.

If $j > 0$, then according to Remark 5, we obtain:

$$\Pr(a_{k+1,j} = 1) = \Pr(a_{k,n-1} = 0)\cdot \Pr(a_{k,j-1} = 1) + \\ + \Pr(a_{k,n-1} = 1)\cdot \Pr(a_{k,j-1} = 1 + f_{n-1}) = \frac{1}{2}\cdot\frac{1}{2} + \frac{1}{2}\cdot\frac{1}{2} = \frac{1}{2} \quad \blacksquare$$

**Theorem 2.** *Let us probability space $\Theta$ be defined, $A$ be $n \times n$ random matrix defined by decomposition (4) with $t$ summands. Then for matrix $A$ any $\overrightarrow{\Omega_j}$ equals $\overrightarrow{f}$ with probability:*

$$\Pr(\overrightarrow{\Omega_j} = \overrightarrow{f}) \geq \frac{2^t - 1}{2^{2t+1}}, \tag{7}$$

*where $\overrightarrow{f}$ is vector of coefficients of the polynomial $f(x)$.*

□ We consider sets $\{d_{1,j}, ..., d_{t,j}, d_{1,j+1}, ..., d_{t,j+1}\}$, which satisfy $d_{i,j} = d_{i,j+1}, i \in \overline{1,t}$. In such case the sum (6) is equal:

$$\overrightarrow{\Omega_j} = \sum_{i=1}^{t} d_{i,j}((\overrightarrow{A_{i,j}} \parallel 0) + (0 \parallel \overrightarrow{A_{i,j+1}})) \tag{8}$$

Let at least one coefficient $d_{i,j}$ be nonzero, then the sum (8) is nontrivial. Since the coefficients $d_{i,j},\ i \in \overline{1,t},\ j \in \overline{0,n-1}$ is mutually independent, we obtain probability as ratio the number of elements satisfied conditions above to the number of all elements $d_{i,j}$: $\Pr\{d_{i,j} = d_{i,j+1}, i \in \overline{1,t}, \exists i : d_{i,j} = 1\} = \Pr\{\Delta\} = \frac{2^t-1}{2^{2t}}$.

Each sum

$$(\overrightarrow{A_{r,j}} \parallel 0) + (0 \parallel \overrightarrow{A_{r,j+1}}) \tag{9}$$

is equal to zero vector of length $n+1$, if $(n-1)-th$ element of the vector $\overrightarrow{A_{r,j}}$ is equal to 0. Otherwise it is equal to $\overrightarrow{f}$. Let sum (8) has $s$ nonzero coefficients $d_{i,j}$, then vector $\overrightarrow{\Omega_j}$ is equal to $\overrightarrow{f}$ at least if the number of summands, such as $d_{i,j}$ is nonzero and $(n-1)-th$ element of $\overrightarrow{A_{r,j}}$ is equal to 1, is odd. Since $(n-1)-th$ elements of the vectors $\overrightarrow{A_{r,j}}$, $r \in \overline{1,t}$ is mutually independent and each of them has uniform distribution on $P$, probability of the odd number of summands with $(n-1)-th$ element equal to 1 is equal to ratio of sum of the odd binomial coefficients $C_s^{2k+1}$ to sum of all binomial coefficients $C_s^k$. It is known that for any $s$ such ratio is equal to $\frac{1}{2}$.

According to mutually independence of the coefficients $d_{i,j}$ and elements of matrices $A_i$ we obtain probability:

$$\Pr(\overrightarrow{\Omega_j} = \overrightarrow{f}) \geq \Pr\{\Delta\} \cdot \Pr\{sum(7) = \overrightarrow{f}\} = \frac{2^t - 1}{2^{2t+1}} \tag{10}$$

■

## 5 Decomposition of the circulant matrices over $\mathbb{F}_{2^s}$

We denote $P = \mathbb{F}_2$, $Q = (P[x]/g(x), +, \cdot)$, $g(x)$ is some irreducible polynomial of degree $s$ over $P$, $Q \cong \mathbb{F}_{2^s}$, $f(x) = x^n + 1$.

**Statement 5.** *Let $C_{m\times m}$ be circulant matrix over $Q$, $n = ms$, matrix $A_{n\times n} = A(C, g(x))$ implements corresponding $C$ transformation on binary vectors of length $n$. Then:*

1. *There exist decomposition (4) for matrix $A$ and polynomial $x^n+1$, which consists of no more than $s$ summands.*
2. *If binary representation of any element of matrix $C$ contains $s-k$ zeros in most significant bits, then there exist decomposition (4), which consists of no more $k$ summands.*

□ We denote

$$A_{n\times n} = \begin{pmatrix} \overrightarrow{A_{n-1}} \\ \dots \\ \overrightarrow{A_1} \\ \overrightarrow{A_0} \end{pmatrix}, C_{m\times m} = \begin{pmatrix} c_0 & c_{m-1} & \dots & c_2 & c_1 \\ c_1 & c_0 & \dots & c_3 & c_2 \\ \dots & \dots & \dots & \dots & \dots \\ c_{m-2} & c_{m-3} & \dots & c_0 & c_{m-1} \\ c_{m-1} & c_{m-2} & \dots & c_1 & c_0 \end{pmatrix} = \begin{pmatrix} \overrightarrow{C_{m-1}} \\ \dots \\ \overrightarrow{C_1} \\ \overrightarrow{C_0} \end{pmatrix}.$$

To get matrix $A$ it is needed to replace elements $c_i$ of matrix $C$ by matrices over $P$, which implement corresponding transformations (field multiplication by element). Since such matrices have representation (2) and dimention $s \times s$, every row $\overrightarrow{C_i}$ is represented by rows $\overrightarrow{A_{is}}, \overrightarrow{A_{is+1}}, ..., \overrightarrow{A_{is+s-1}}$. Since every row $\overrightarrow{C_i}$ is left rotation of row $\overrightarrow{C_{i-1}}$, for any $i \in \overline{1, n-1}$, $k \in \overline{0, s-1}$ the row $\overrightarrow{A_{is+k}}$ is $s$-times left rotation of row $\overrightarrow{A_{(i-1)s+k}}$.

Let $A_k$, $k \in \overline{0, s-1}$ be $n \times n$ circulant matrix over $P$, which is defined by row $\overrightarrow{A_k}$ of matrix $A$. Then the rows with numbers $is + k, i \in \overline{0, n-1}$ of matrices $A$ and $A_k$ are equal.

Let $D_k$ be $n \times n$ diagonal matrix over $\mathbb{F}_2$ with ones only on positions $is + k, i \in \overline{0, n-1}$. Then according to the constructions of matrices $A_k$ and $D_k$ the target equality holds:

$$A = \sum_{i=0}^{s-1} D_i A_i \tag{11}$$

We have $s - k$ zeros in most significant bits in binary representation of every element in matrix $C$ in point 2. According to formula (2), for any $j \in \overline{1, s-k}$ the $j-th$ row of matrix $A_{c_i(x),g(x)}$ is left shift of $(j-1)-th$ row. There is no modulo reduction because the most significant bit in $(j-1)-th$ row is zero. Then the matrices $A_0, ..., A_{s-k}$ are the same in sum (11) and we can replace the sum $D_0A_0 + ... + D_{s-k}A_{s-k}$ by one summand and at most $k$ summands will remain in sum (11). ■

We present some examples of Statement 5 implementation. We denote $diag(0x\alpha)$ – diagonal matrix over $\mathbb{F}_2$, which contain the same bytes $\alpha$ on the diagonal. For example, $diag(0x20) = diag(0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20) \in P_{64,64}$ in Example 1. Given examples allow us to offer alternative implementations of the corresponding linear transformations (see Statement 4).

**Example 1.** *Matrix $A(W, g(x))$ is used in the linear transformation of Whirlpool hash function, where $W$ is $8 \times 8$ MDS circulant matrix over $\mathbb{F}_{2^8}$ and $g(x) = x^8 + x^4 + x^3 + x^2 + 1$.*

*$W = Circ_{2^8}(0x01, 0x04, 0x01, 0x08, 0x05, 0x02, 0x09, 0x01)$. Since every element of $W$ has four zeroes in the most significant bits in binary representation, according to Statement 5 there exist matrix $A(W, g(x))$ decom-*

*position (4), which consists of four summands. We present it:*

$$\begin{aligned} A(W,\ g(x)) = & Circ_2(0x01, 0x04, 0x01, 0x08, 0x05, 0x02, 0x09, 0x01) + \\ & + Diag(0x20)Circ_2(0x00, 0x00, 0x00, 0x08, 0xe8, 0x00, 0x08, 0xe8) + \\ & + Diag(0x40)Circ_2(0x00, 0x04, 0x74, 0x08, 0xec, 0x74, 0x08, 0xe8) + \\ & + Diag(0x80)Circ_2(0x00, 0x04, 0x74, 0x08, 0xec, 0x76, 0x32, 0xe8). \end{aligned}$$

**Example 2.** *Let us* $g(x) = x^8 + x^4 + x^3 + x^2 + 1$. *Then the matrix* $V = Circ_{2^8}(0x01, 0x02, 0x03, 0x05, 0x04, 0x03, 0x07, 0x07)$ *is also* $8 \times 8$ *MDS circulant matrix over* $\mathbb{F}_{2^8}$ *and according to Statement 5 there exist matrix* $A(V, g(x))$ *decomposition (4), which consists of three summands. We present it:*

$$\begin{aligned} A(V, g(x)) = & Circ_2(0x01, 0x02, 0x03, 0x05, 0x04, 0x03, 0x07, 0x07) + \\ & + Diag(0x40)Circ_2(0x74, 0x00, 0x00, 0x04, 0x70, 0x74, 0x04, 0x70) + \\ & + Diag(0x80)Circ_2(0x4e, 0x02, 0x38, 0x3e, 0x70, 0x76, 0x3c, 0x48). \end{aligned}$$

**Example 3.** *Matrix* $A(L, g(x))$ *is used in the linear transformation of AES block cipher, where* $L = Circ_{2^8}(0x03, 0x01, 0x01, 0x02)$ *is* $4 \times 4$ *MDS circulant matrix over* $\mathbb{F}_{2^8}$, $g(x) = x^8 + x^4 + x^3 + x + 1$. *According to Statement 5 there exist matrix* $A(L, g(x))$ *decomposition (4), which consists of two summands.*

*We present it:*

$$\begin{aligned} A(L, g(x)) = & Circ_2(0x03, 0x01, 0x01, 0x02) + \\ & + Diag(0x80)Circ_2(0x34, 0x36, 0x00, 0x02). \end{aligned}$$

**Example 4.** *There exist* $4 \times 4$ *MDS matrix on* $8 - subvectors$ *over* $\mathbb{F}_2$. *We present it:*

$$L' = Circ_2(0x01, 0x04, 0x04, 0x05).$$

# References


[1] Claude E. Shannon, "Communication theory of secrecy systems", *Bell System Technical Journal*, **28**:4 (1949), 656-715.

[2] Eli Biham and Adi Shamir, "Differential cryptanalysis of DES-like cryptosystems", *Journal of Cryptology*, **4** (1990), 3-72.

[3] Mitsuru Matsui, "Linear Cryptanalysis Method for DES Cipher", *International Conference on the Theory and Application of Cryptographic Techniques*, **28** (1994).

[4] F. M. Malyshev, "The duality of differential and linear methods in cryptography", *Mathematical Aspects of Cryptography*, **5**:3 (2014), 35-47, In Russian.

[5] Gupta Kishan and Ray Indranil, "Cryptographically significant MDS matrices based on circulant and circulant-like matrices for lightweight applications", *Cryptography and Communications*, **7** (2015).

[6] V. Dolmatov, A. Degtyarev, "GOST R 34.11-2012: Hash Function", *Request for Comments*, **RFC: 6986** (2013).

[7] Guo Jian and Peyrin Thomas and Poschmann Axel, "The PHOTON family of lightweight hash functions", *IACR Cryptology ePrint Archive*, **2011** (2011), 222-239.

[8] V. Dolmatov, "GOST R 34.12-2015: Block Cipher "Kuznyechik"", *Request for Comments*, **RFC: 7801** (2016).

[9] "Information technology – Security techniques – Encryption algorithms – Part 3: Block ciphers", *ISO/IEC*, **18033-3** (2010).

[10] Whitfield Diffie and George Ledin, "SMS4 Encryption Algorithm for Wireless Networks (Translated version)", *IACR Cryptology ePrint Archive*, **4** (2008), 329.

[11] "Information technology – Security techniques – Hash-functions – Part 3: Dedicated hash-functions", *ISO/IEC*, **10118-3** (2004).

[12] Agner Fog, "Instruction tables: Lists of instruction latencies, throughputs and micro-operation breakdowns for Intel, AMD and VIA CPUs", 1996-2022.

[13] Shay Gueron, "Intel's New AES Instructions for Enhanced Performance and Security", *Fast Software Encryption Workshop*, 2009.

[14] S. V. Dorokhin and S. S. Kachkov and A. A. Sidorenko, "Implementation of «Kuznyechik» cipher using vector instructions", *MIPT works*, **10**:4 (2018), In Russian.

[15] Daniel Lemire and Owen Kaser, "Faster 64-bit universal hashing using carry-less multiplications", *Journal of Cryptographic Engineering*, **6** (2015), 171-185.

# Matrix-vector product of a new class of quasi-involutory MDS matrices

Pablo Freyre Arrozarena, Ernesto Dominguez Fiallo,
and Ramsés Rodríguez Aulet

Institute of Cryptography, Cuba
pfreyre@matcom.uh.cu, edominguezfiallo@nauta.cu, ramses@nauta.cu

**Abstract**

We define a new class of quasi-involutive MDS matrices. It is known that the matrix-vector multiplication operation can be expressed through multiplication of two polynomials modulo a generating polynomial of a cyclic code. Using this, the new class of quasi-involutive MDS matrices is defined by a relation that satisfies the MDS matrix and its inverse in the matrix-vector multiplication operation.

**Keywords:** quasi-involutive MDS matrices, matrix-vector multiplication.

## 1 Introduction

Maximum Distance Separable (MDS) matrices are of great importance in the design of block ciphers and hash functions because theoretically ensures a perfect diffusion. However, MDS matrices are in general not sparse, have a large description and induce costly implementations.

To reduce implementation costs, research focuses on circulant [1], recursive matrices [2], and methods for transforming an MDS matrix into other ones [3]. A way of expressing the MDS matrix-vector product based on the multiplication of two polynomials modulo a generating polynomial of the cyclic code was recently proposed [4]. With this result, in addition to not needing to store the MDS matrix explicitly, it does not need to have an additional structure to be implemented efficiently.

It is useful, mainly in lightweight cryptography, to use involutory MDS matrices. The main advantage lies in that both encryption and decryption share the same matrix-vector product. However, finding involutory MDS matrices, in particular large MDS matrices, is not an easy task and this topic has attracted attention recently [6, 7, 8, 9, 10, 11].

The concepts of quasi-involutory MDS matrix captures the intuitive idea of a MDS matrix that is close to being involutory and has been used in research. For example, in [12] means that the inverse matrix is obtained by squaring the coefficients of the MDS matrix a fixed number of times equal to the size of the matrix. From an implementation point of view, this means one additional bit permutation to represent the matrix-vector product of the inverse.

**Our contribution.** A new class of quasi-involutive MDS matrices is proposed. This new class is defined by the relation that satisfies the MDS matrix and its inverse in the matrix-vector multiplication operation expressed through multiplication of two polynomials modulo a generating polynomial of a cyclic code as in [4], but using two special classes of cyclic codes: Reed-Solomon (RS) codes [13] and the code proposed in [5]. We will call this last code the GCMN code. We show that if $p \neq 2$ in $\mathbb{F}_{p^n}$ where the MDS matrix is defined, the multiplication of the vector by the inverse matrix coincides with the multiplication of the vector by the original matrix, that is, the MDS matrix in question is involutive. If $p = 2$, the vector is first transformed one step through an LFSR and the multiplication by the inverse matrix can be performed with the original MDS matrix, that is, the MDS matrix in question is quasi-involutive.

## 2 Preliminaries

Let $q$ be the power of a prime $p$. Let $\mathbb{F}_q$ the finite field containing $q$ elements and $\mathbb{F}_q[x]$ be the polynomial ring with coefficients in $\mathbb{F}_q$. A *linear code* $\mathcal{C}$ of length $n$ and dimension $k$ over $\mathbb{F}_q$, denoted as $[n, k]_q$, is a linear subspace of dimension $k$ of the linear space $\mathbb{F}_q^n$. The *minimum distance* $d$ of $\mathcal{C}$ is the minimum weight if its nonzero vectors and we denote the code as $[n, k, d]_q$. A *generator matrix* for $\mathcal{C}$ is a matrix whose rows form a basis for $\mathcal{C}$ and it is said to be in *standard form* if it has the form $(I_k|R)$ where $I_k$ is a $k \times k$ identity matrix and $R$ is a $k \times (n - k)$ matrix.

A linear code such that $d = n - k + 1$ (Singleton Bound) is called a *Maximum Distance Separable (MDS) code*. In cryptography, we are more interested in the so-called MDS square matrices. A *matrix is MDS* if and only if all its minors are non zero.

An $[n, k]_q$ code is said to be cyclic if a cyclic shift of any element of the code remains in the code and can be seen, in algebraic terms, as ideals of $\mathbb{F}_q[x]/(x^n - 1)$. Every monic polynomial $g(x)$ that divides $x^n - 1$ generates

a distinct ideal, and therefore, completely specifies a cyclic code. For this reason, the polynomial $g(x)$ is called *generating polynomial* and its order $e$ is the smallest positive integer such that $g(x)$ divides $x^e - 1$ with $e$ divide $n$. If the degree of $g(x)$, denoted by $deg(g)$, is $r$, then the code defined by $g(x)$ has dimension $k = n - r$ and a generator matrix, in standard form, can be given by $(I_k| - R)$ with

$$R = \begin{pmatrix} x^{n-k} & \mod g(x) \\ x^{n-k+1} & \mod g(x) \\ & \vdots \\ x^{n-1} & \mod g(x) \end{pmatrix} \tag{1}$$

Reed–Solomon (RS) codes are special cyclic codes. A $q$-ary RS code over $\mathbb{F}_q$ of length $q-1,\ q>2$, is the cyclic code generated by a polynomial of the form

$$g(x) = (x - \alpha^{a+1})(x - \alpha^{a+2}) \cdots (x - \alpha^{a+\delta-1})$$

with $a \geq 0$ and $2 \leq \delta \leq q-1$, where $\alpha$ is a primitive element of $\mathbb{F}_q$. It is an MDS code with parameters $[q-1, q-\delta, \delta]_q$ and the matrix $R$ is a MDS matrix. Since $\alpha$ is a primitive element, the order of the generating polynomial of an RS code is $q-1$.

In [5] a code was proposed using linear recurring sequences. The code is composed of segments of length n of the linear recurring sequences that have characteristic polynomial

$$g(x) = (x - \beta_0) \cdots (x - \beta_{m-1})$$

that is, for $i = 0, 1, \ldots, n-m+1$ the code has the form

$$\mathcal{K} = \{(u(0), \ldots, u(n)) : u(i+m) = g_0 u(i) + \cdots + g_{n-1} u(i+m-1)\}$$

where $g_0, \ldots, g_{m-1}$ are the coefficients of $g(x)$. If $\beta_0, \ldots, \beta_{m-1}$ are selected according to [5, Theorem 4], then it is an MDS code with parameters $[q+1, m, q-m+2]$. If $q$ is even or $m$ is odd, then the code is cyclic and the order of the polynomial $g(x)$ is conditioned by the order of the elements $\beta_0, \ldots, \beta_{m-1}$. In [5] it is shown that

$$ord(\beta_i) | q+1,\ 0 \leq i \leq m-1$$

and $\beta_i \neq \beta_j,\ i \neq j,\ 0 \leq i, j \leq m-1$. Then, if $q+1$ is prime, it can be easily seen that the code is cyclic and the order of $g(x)$ is $q+1$. We will call this code, GCMN code. By this reasoning, it is possible to operate by multiplying polynomials as shown in [4].

## 3 Involutory and quasi-involutory MDS matrix-vector product

In this section we show how to use the representation of the matrix-vector product through the multiplication of two polynomials modulo a polynomial as in [4], for the case in which the MDS matrix to be used is involutive or quasi-involutive. We must first formalize our definition of a quasi-involutive linear transformation.

**Definition 1.** *Let $n \in \mathbb{N}$. The biyective linear transformation*

$$\Psi : P[x]/g(x) \to P[x]/g(x)$$

*defined by*

$$\forall p(x) \in P[x]/g(x) : \ \Psi(p(x)) = p(x) \cdot x^n \mod g(x)$$

*is quasi-involutive if its inverse $\Psi^{-1}$ is*

$$\Psi^{-1}(p(x)) = \Psi(p(x)) \cdot x \mod g(x)$$

As a consequence of the result obtained in [4], it follows that to multiply a vector by any square MDS submatrix of matrix (1), it can be done by multiplying the polynomial that represents the vector by the polynomial corresponding to the first row of the selected submatrix. In algorithms 1 and 2 it is shown how the involutive and quasi-involutive MDS matrices are represented by the multiplication of two polynomials modulo a generating polynomial of RS or GCMN code. The row $i$ of the matrix $M$ is denoted by $M_i$.

**Algorithm 1:** Generation of involutory and quasi-involutory MDS matrix.

**Input :**

- The RS or GCMN generating polynomial $g(x) \in \mathbb{F}_q[x]$ of degree $n-k$.
- The canonical polynomials $x^i,\ 0 \leq i \leq n-1$.

**Output:** involutory or quasi-involutory $n \times n$ MDS matrix $M$.

**Data :** $q = p^t,\ p$ prime and $t \in \mathbb{N}$.

**if** $p > 2$ **then**
  **if** $g(x)$ *is RS* **then**
    $f(x) \leftarrow \left(-x^{\frac{q-1}{2}} \mod g(x)\right)$;
  **if** $g(x)$ *is GCMN* **then**
    $f(x) \leftarrow \left(-x^{\frac{q+1}{2}} \mod g(x)\right)$;
  **for** $i = 1$ **to** $n$ **do**
    $M_i \leftarrow x^{i-1} \cdot f(x) \mod g(x)$;
**if** $p = 2$ **then**
  **if** $g(x)$ *is RS* **then**
    $f(x) \leftarrow x^{2^{t-1}-1} \mod g(x)$;
  **if** $g(x)$ *is GCMN* **then**
    $f(x) \leftarrow x^{2^{t-1}} \mod g(x)$;
  **for** $i = 1$ **to** $n$ **do**
    $M_i \leftarrow x^{i-1} \cdot f(x) \mod g(x)$;
**return** $M$;

**Algorithm 2:** Generation of involutory and quasi-involutory MDS inverse matrix.

**Input :**

- The RS or GCMN generating polynomial $g(x) \in \mathbb{F}_q[x]$ of degree $n-k$.
- The canonical polynomials $x^i,\ 0 \leq i \leq n-1$.

**Output:** involutory or quasi-involutory $n \times n$ MDS inverse matrix $M^{-1}$.

**Data :** $q = p^t,\ p$ prime and $t \in \mathbb{N}$.

**if** $p > 2$ **then**
  **if** $g(x)$ *is RS* **then**
    $f(x) \leftarrow \left(-x^{\frac{q-1}{2}} \mod g(x)\right)$;
  **if** $g(x)$ *is GCMN* **then**
    $f(x) \leftarrow \left(-x^{\frac{q+1}{2}} \mod g(x)\right)$;
  **for** $i = 1$ **to** $n$ **do**
    $M_i \leftarrow x^i \cdot f(x) \mod g(x)$;
**if** $p = 2$ **then**
  **if** $g(x)$ *is RS* **then**
    $f(x) \leftarrow x^{2^{t-1}} \mod g(x)$;
  **if** $g(x)$ *is GCMN* **then**
    $f(x) \leftarrow x^{2^{t-1}+1} \mod g(x)$;
  **for** $i = 1$ **to** $n$ **do**
    $M_i \leftarrow x^{i-1} \cdot f(x) \mod g(x)$;
**return** $M^{-1}$;

Algorithms 3 and 4 show how to perform the matrix-vector product-operation.

**Algorithm 3:** Multiplication of a vector by involutory or quasi-involutory MDS matrix

**Input :**

- The RS or GCMN generating polynomial $g(x) \in \mathbb{F}_q[x]$ of degree $n-k$.
- The vector of coefficients $a = (a_0, a_1, \ldots, a_{n-1})$.

**Output:** The vector $\hat{a} = a \cdot M$.

**Data :** $q = p^t$, $p$ prime and $t \in \mathbb{N}$.

**if** $p > 2$ **then**
  **if** $g(x)$ *is RS* **then**
    $f(x) \leftarrow \left(-x^{\frac{q-1}{2}} \mod g(x)\right)$;
  **if** $g(x)$ *is GCMN* **then**
    $f(x) \leftarrow \left(-x^{\frac{q+1}{2}} \mod g(x)\right)$;
  $\hat{a}(x) \leftarrow a(x) \cdot f(x) \mod g(x)$;
**if** $p = 2$ **then**
  **if** $g(x)$ *is RS* **then**
    $f(x) \leftarrow x^{2^{t-1}-1} \mod g(x)$;
  **if** $g(x)$ *is GCMN* **then**
    $f(x) \leftarrow x^{2^{t-1}} \mod g(x)$;
  $\hat{a}(x) \leftarrow a(x) \cdot f(x) \mod g(x)$;
**return** $\hat{a}$ `//coefficients of` $\hat{a}(x)$

**Algorithm 4:** Multiplication of a vector by the inverse of involutory or quasi-involutory MDS matrix

**Input :**

- The RS or GCMN generating polynomial $g(x) \in \mathbb{F}_q[x]$ of degree $n-k$.
- The vector of coefficients $a = (a_0, a_1, \ldots, a_{n-1})$.

**Output:** The vector $\hat{a} = a \cdot M^{-1}$.

**Data :** $q = p^t$, $p$ prime and $t \in \mathbb{N}$.

**if** $p > 2$ **then**
  **if** $g(x)$ *is RS* **then**
    $f(x) \leftarrow \left(-x^{\frac{q-1}{2}} \mod g(x)\right)$;
  **if** $g(x)$ *is GCMN* **then**
    $f(x) \leftarrow \left(-x^{\frac{q+1}{2}} \mod g(x)\right)$;
  $\hat{a}(x) \leftarrow a(x) \cdot f(x) \mod g(x)$;
**if** $p = 2$ **then**
  **if** $g(x)$ *is RS* **then**
    $f(x) \leftarrow x^{2^{t-1}-1} \mod g(x)$;
  **if** $g(x)$ *is GCMN* **then**
    $f(x) \leftarrow x^{2^{t-1}} \mod g(x)$;
  $a(x) \leftarrow a(x) \cdot x \mod g(x)$;
  $\hat{a}(x) \leftarrow a(x) \cdot f(x) \mod g(x)$;
**return** $\hat{a}$ `//coefficients of` $\hat{a}(x)$

**Example of MDS matrix of size $8 \times 8$ in an RS code:** Let's consider the finite field $\mathbb{F}_{2^8}$ with polynomial $x^8 + x^4 + x^3 + x^2 + 1$. We have then that $n = 2^8 - 1 = 255$, $\delta = 9$, $k = 2^8 - 9 = 247$. The generator polynomial is

$$g(x) = x^8 + \alpha^{176}x^7 + \alpha^{240}x^6 + \alpha^{211}x^5 + \alpha^{253}x^4 + \alpha^{220}x^3 + \alpha^3x^2 + \alpha^{203}x + \alpha^{36}$$

The matrix $R$ en (1) is as follows

$$R = \begin{pmatrix} x^8 & \mod g(x) \\ x^9 & \mod g(x) \\ & \vdots \\ x^{254} & \mod g(x) \end{pmatrix}$$

Applying algorithm 1, the obtained square MDS matrix is

$$M = \begin{pmatrix} x^{127} & \mod g(x) \\ x^{128} & \mod g(x) \\ & \vdots \\ x^{134} & \mod g(x) \end{pmatrix}$$

$$M = \begin{pmatrix} \texttt{0x49} & \texttt{0xe4} & \texttt{0x8e} & \texttt{0xec} & \texttt{0x3a} & \texttt{0x15} & \texttt{0x1d} & \texttt{0xa4} \\ \texttt{0x6d} & \texttt{0xd0} & \texttt{0xdb} & \texttt{0xc0} & \texttt{0xf} & \texttt{0x12} & \texttt{0xea} & \texttt{0x72} \\ \texttt{0xc4} & \texttt{0xa8} & \texttt{0x95} & \texttt{0x3a} & \texttt{0x35} & \texttt{0xdf} & \texttt{0xe6} & \texttt{0x12} \\ \texttt{0x34} & \texttt{0x12} & \texttt{0x6c} & \texttt{0x9f} & \texttt{0x23} & \texttt{0x6b} & \texttt{0x5d} & \texttt{0x9e} \\ \texttt{0x43} & \texttt{0xf3} & \texttt{0xd1} & \texttt{0x7} & \texttt{0xd7} & \texttt{0xab} & \texttt{0x4f} & \texttt{0x93} \\ \texttt{0xe9} & \texttt{0x5d} & \texttt{0x2} & \texttt{0x64} & \texttt{0x92} & \texttt{0xb8} & \texttt{0x6f} & \texttt{0x60} \\ \texttt{0xff} & \texttt{0xf3} & \texttt{0xbd} & \texttt{0xbe} & \texttt{0x96} & \texttt{0x4d} & \texttt{0xc1} & \texttt{0x2c} \\ \texttt{0x3b} & \texttt{0x2b} & \texttt{0xb1} & \texttt{0x3d} & \texttt{0x1a} & \texttt{0x90} & \texttt{0x1f} & \texttt{0x8f} \end{pmatrix}$$

Let the vector $a = \left(\alpha^7, \alpha^{123}, \alpha^{58}, \alpha^{91}, \alpha^{72}, \alpha^{45}, \alpha^{208}, \alpha^{237}\right) \in \mathbb{F}_{2^8}^8$. To perform the operation $a \cdot M$ applying algorithm (3), the operation

$$a(x) \cdot \left(x^{2^{8-1}-1} \mod g(x)\right) \mod g(x)$$

must be performed, where

$$a(x) = \alpha^7 + \alpha^{123}x + \alpha^{58}x^2 + \alpha^{91}x^3 + \alpha^{72}x^4 + \alpha^{45}x^5 + \alpha^{208}x^6 + \alpha^{237}x^7$$

The result is the polynomial

$$\hat{a}(x) = \alpha^{209} + \alpha^{15}x + \alpha^{245}x^2 + \alpha^{90}x^3 + \alpha^{19}x^4 + \alpha^{157}x^5 + \alpha^{52}x^6 + \alpha^{11}x^7$$

which represents the vector $\hat{a} = \left(\alpha^{209}, \alpha^{15}, \alpha^{245}, \alpha^{90}, \alpha^{19}, \alpha^{157}, \alpha^{52}, \alpha^{11}\right)$. It can be verified by means of the usual multiplication of a vector by a matrix that

$$\hat{a} = a \cdot M$$

Applying algorithm (2) is obtained $M^{-1}$

$$M^{-1} = \begin{pmatrix} \texttt{0xe4} & \texttt{0x8e} & \texttt{0xec} & \texttt{0x3a} & \texttt{0x15} & \texttt{0x1d} & \texttt{0xa4} & \texttt{0xb9} \\ \texttt{0xd0} & \texttt{0xdb} & \texttt{0xc0} & \texttt{0xf} & \texttt{0x12} & \texttt{0xea} & \texttt{0x72} & \texttt{0x34} \\ \texttt{0xa8} & \texttt{0x95} & \texttt{0x3a} & \texttt{0x35} & \texttt{0xdf} & \texttt{0xe6} & \texttt{0x12} & \texttt{0x7e} \\ \texttt{0x12} & \texttt{0x6c} & \texttt{0x9f} & \texttt{0x23} & \texttt{0x6b} & \texttt{0x5d} & \texttt{0x9e} & \texttt{0xe8} \\ \texttt{0xf3} & \texttt{0xd1} & \texttt{0x7} & \texttt{0xd7} & \texttt{0xab} & \texttt{0x4f} & \texttt{0x93} & \texttt{0x74} \\ \texttt{0x5d} & \texttt{0x2} & \texttt{0x64} & \texttt{0x92} & \texttt{0xb8} & \texttt{0x6f} & \texttt{0x60} & \texttt{0x78} \\ \texttt{0xf3} & \texttt{0xbd} & \texttt{0xbe} & \texttt{0x96} & \texttt{0x4d} & \texttt{0xc1} & \texttt{0x2c} & \texttt{0x5a} \\ \texttt{0x2b} & \texttt{0xb1} & \texttt{0x3d} & \texttt{0x1a} & \texttt{0x90} & \texttt{0x1f} & \texttt{0x8f} & \texttt{0x30} \end{pmatrix}$$

Let the vector $\hat{a} = a \cdot M = \left(\alpha^{209}, \alpha^{15}, \alpha^{245}, \alpha^{90}, \alpha^{19}, \alpha^{157}, \alpha^{52}, \alpha^{11}\right)$. To perform the operation $\hat{a} \cdot M^{-1}$ applying algorithm (4), the operations

$$\hat{a}(x) \leftarrow \hat{a}(x) \cdot x \mod g(x)$$

$$\hat{a}(x) \cdot \left(x^{2^7-1} \mod g(x)\right) \mod g(x)$$

must be performed, where $\hat{a}(x)$ is the polynomial

$$\hat{a}(x) = \alpha^{209} + \alpha^{15}x + \alpha^{245}x^2 + \alpha^{90}x^3 + \alpha^{19}x^4 + \alpha^{157}x^5 + \alpha^{52}x^6 + \alpha^{11}x^7$$

The result is, in effect, the polynomial

$$a(x) = \alpha^{7} + \alpha^{123}x + \alpha^{58}x^2 + \alpha^{91}x^3 + \alpha^{72}x^4 + \alpha^{45}x^5 + \alpha^{208}x^6 + \alpha^{237}x^7$$

which represents the vector $a = \left(\alpha^{7}, \alpha^{123}, \alpha^{58}, \alpha^{91}, \alpha^{72}, \alpha^{45}, \alpha^{208}, \alpha^{237}\right)$.

## 4 Conclusion

In this paper, a new class of quasi-involutive MDS matrices has been defined. When the matrix is defined in a finite field of characteristic different from 2, the MDS matrix in question is involutive. When the characteristic is 2, the MDS matrix in question is quasi-involutive and in this case, the inverse matrix-vector product, expressed through the multiplication of two polynomials modulo a polynomial that generates a cyclic code that is MDS, is done by shifting the vector one position to the right using an LFSR and subsequently multiplying by the original MDS matrix.

## References

[1] Gupta, K.C., Pandey, S.K., Venkateswarlu, A., "On the direct construction of recursive MDS matrices", Designs, Codes and Cryptography, **82**, Springer, 2017, 77-94.

[2] Gupta, K.C., Pandey, S.K., Samanta, S., "Construction of Recursive MDS Matrices Using DLS Matrices", *Progress in Cryptology - AFRICACRYPT 2022*, Lecture Notes in Computer Science, **13503**, ed. Batina, L., Daemen, J., Springer, Cham., 2022, 3-27.

[3] Luong, T. T., Cuong, N. N., "Direct exponent and scalar multiplication transformations of MDS matrices: some good cryptographic results for dynamic diffusion layers of block ciphers", Journal of Computer Science and Cybernetics, **32** (2016), 1-17.

[4] Freyre, P., Fiallo, E. D., "Efficient multiplication of a vector by a matrix MDS", Journal of Science and Technology on Information security, **3**, 17, 2022, 26–36.

[5] Couselo, E., Gonzalez, S., Markov, V., Nechaev, A., "The parameters of recursive MDS-codes", *Discrete Math. Appl.*, **10**:5 (2000), 433–453.

[6] Sajadieh, M., Dakhilalian, M., Mala, H., Omoomi, B., "On construction of involutory MDS matrices from Vandermonde Matrices in $GF(2^q)$", Designs, Codes and Cryptography, **64**, Springer, 2012, 287–308.

[7] Gupta, K. C, Ray, I. G., "On constructions of involutory MDS matrices", *Lecture Notes in Computer Science*, Progress in Cryptology – AFRICACRYPT 2013, **7918**, Springer, Berlin, Heidelberg, 2013, 43–60.

[8] Sim, S. M., Khoo, K., Oggier, F., Peyrin, T., "Lightweight MDS involution matrices", *Lecture Notes in Computer Science*, International Workshop on Fast Software Encryption, **9054**, ed. Leander, G, Springer, Berlin, Heidelberg, 2015, 471-493.

[9] Cauchois, V., Loidreau, P., "On circulant involutory MDS matrices", Designs, Codes and Cryptography, **87**, 2019, 249-260.

[10] Gupta, K. C., Pandey, S. K., Venkateswarlu, A., "Almost involutory recursive MDS diffusion layers", Designs, Codes and Cryptography, **87**, 2019, 609–626.

[11] Zhou, X., Cong, T., "Construction of generalized-involutory MDS matrices", Cryptology ePrint Archive, 2022.

[12] Cauchois, V., Loidreau, P., Merkiche, N., "Direct construction of quasi-involutory recursive-like MDS matrices from 2-cyclic codes", IACR Transactions on Symmetric Cryptology, 2016.

[13] Reed, I. S., Solomon, G., "Polynomial codes over certain finite fields", Journal of the society for industrial and applied mathematics, **8**, SIAM, 1960, 300–304.

# Construction of linear involutory transformations over finite fields through the multiplication of polynomials modulus a polynomial

Ramsés Rodríguez Aulet, Alejandro Freyre Echevarría, and Pablo Freyre Arrozarena

Institute of Cryptography, Cuba
ramsesrusia@yahoo.com, freyreealejandro@gmail.com, pfreyre@matcom.uh.cu

**Abstract**

Matrices have a fundamental role in many encryption schemes. In the present paper we analyze the construction of involutory matrices obtained as the matrix associated to a linear transformation which is formed by the multiplication of polynomials modulus a polynomial over a finite field. We also propose a method to generate involutory MDS matrices with random coefficients which can be used as key-dependent components in dynamic cryptographic schemes.

**Keywords:** Finite ring, finite field, nilpotent element, involutory MDS matrix.

## Introduction

Let $\mathcal{P} = \mathbb{F}_q$ be a finite field of $q = p^t$ elements being $p$ a prime number and $t \in \mathbb{N}$. We have that $\mathcal{P}[x]$ is the ring of polynomials having coefficients in $\mathcal{P}$ and using the polynomial $F(x) \in \mathcal{P}[x]$ let us build the ring $\mathcal{R} = \mathcal{P}[x]/_{F(x)}$. Let $g(x), \mu(x)$ be elements of $\mathcal{R}$ we define the transformation $\Psi : \mathcal{R} \to \mathcal{R}$ as

$$\Psi(g(x)) = g(x)\mu(x) \bmod F(x). \tag{1}$$

It is known that the transformation from (1) is linear [1] and it is a bijective transformation iff $\gcd(\mu(x), F(x)) = 1$, where gcd denotes the greatest common divisor of two polynomials, i.e. the polynomials are coprime. This type of transformations are used in encryption algorithms like AES [2]. Moreover the linear transformation of Kuznyechik algorithm [3] and the matrix of the hash function Photon [4], among other examples, can be expressed in terms of equation (1). Matrices are used to provide diffusion within these schemes. In many encryption schemes the **M**aximum **D**istance of **S**eparation

(MDS) matrices are used to ensure efficiently, and within the lowest number of rounds, the dependence of all input bits with the key, and equally reach the randomness state.

In practice, any encryption scheme must be secure and easy to implement. One of the strategies that can be follow through the implementation of these schemes is that the encryption and decryption procedures are very similar. Under such design principles are designed the encryption algorithms GOST [3] and Simon [5] having equal encryption and decryption operations. Ciphers like Khazad [6] and Anubis [7] use involutory linear transformations, i.e. the inverse of the used matrix is itself. In [8] some methods for constructing invulotory, and particularly MDS matrices are studied.

The current paper is devoted to the analysis of when the matrices associated to linear transformations constructed according to equation (1) are involutory. We study the branch number of the involutory linear transformations constructed using (1) and within them we analyze those which contain MDS matrices of size 4 . Although there exist several ways to design involutory matrices, our paper focus on using the nilpotent elements of a ring. We first establish a relation between the nilpotent and involutory elements of a ring and such relation is used to the latter construction of involutory linear transformations.

The structure of the paper is organized as follows. Section 1 present the relation between nilpotent and involutory elements of a ring. In Section 2 we establish a criteria of when the linear transformation presented in (1) is involutory. In addition, we analyze the branch number of different classes and MDS matrices are determined. In Section 3 we present a new algorithm for the generation of involutory MDS matrices. We summarize our results in the conclusions.

## 1 Relation between nilpotent and involutory elements of a ring of even characteristic

Through this section we will treat $\mathcal{R}$ as a commutative ring with identity [9]. Within this algebraic structure we will work with the ideals and the characteristic of the ring [9].

**Definition 1** ([9]). *The element $r \in \mathcal{R},\ r \neq 0$ is called a zero divisor if exists $r' \in \mathcal{R},\ r' \neq 0$ such that*

$$r * r' = 0.$$

*A zero divisor $r \in \mathcal{R}$ is called nilpotent if exists $n \in \mathbb{N}$ such that*

$$r^n = 0.$$

*The lowest $n = n(r)$ which satisfies the previous property is called nilpotency index.*

**Definition 2** ([9])**.** *An element $r \in \mathcal{R}$, $r \neq e$ is called involutory if*

$$r^2 = e.$$

*i.e., the multiplicative order of $r$ is 2.*

In simple words, if the multiplicative order of $r$ is equal to 2 then $r$ is called an involutory element. If $r$ is a nilpotent element of a ring $\mathcal{R}$ then one can construct the invertible element $e - r$ [9]. From this fact we state the following proposition.

**Proposition 1.** *Let $\mathcal{R}$ be a ring of characteristic equal to 2. The following propositions holds*

1. *If $\mathcal{R}$ contains at least one involutory element $r \in \mathcal{R}$ then $\mathcal{R}$ contains at least one nilpotent element with nilpotency index 2.*
2. *Let $r'$ be one involutory element of $\mathcal{R}$ then the remaining involutory elements $r$ are uniquely defined by*

   $$r = r' + w$$

   *where $w$ is a nilpotent element with nilpotency index 2, i.e., any two involutory elements are related by a nilpotent element.*

*Proof.* We split the demonstration of the proposition into the demonstration of each of its statements.

1. Let $r \in \mathcal{R}$ an involutory element. Let us analyze the element $a = r + e$. By the application of the properties of the characteristic of a ring [9] we obtain that

   $$a^2 = (r + e)^2 = r^2 + e^2 = e + e = 0.$$

   From the previous relation we can see that $a$ is a nilpotent element.

2. Let us demonstrate that two distinct involutory elements $r, r' \in \mathcal{R}$ are related by a nilpotent element $a \in \mathcal{R}$. Let the element $a \in \mathcal{R}$ be $a = r + r'$ then

   $$a^2 = (r + r')^2 = r^2 + (r')^2 = e + e = 0.$$

   Hence, it is demonstrated that two distinct involutory elements are related by a nilponent element with nilpotency index 2.

Therefore the proof is complete. □

Let $\mathcal{P}_{n,n}$ denote the commutative ring of quadratic matrices of size $n \times n$, $\mathcal{I}_{n,n}$ the identity matrix and $\mathcal{O}_{n,n}$ the null matrix. Then we can enunciate the following corollary.

**Corollary 1.** *If $\mathcal{A} \in \mathcal{P}_{n,n}$ is an involutory matrix then $\mathcal{A}+\mathcal{I}_{n,n}$ is singular.*

*Proof.* The set of $n \times n$ matrices with elements over a finite field $\mathcal{P}$ make an associative ring [9]. If $\mathcal{A}$ is an involutory matrix then using the first statement of Proposition 1 for matrix $\mathcal{B} = \mathcal{A}+\mathcal{I}_{n,n}$ we have that $\mathcal{B}^2 = \mathcal{O}_{n,n}$ and therefore $\mathcal{B}$ is singular. □

Other way to address the demonstration of Corollary 1 is the following. Let $d$ denote the determinant of the matrix $\mathcal{A}+\mathcal{I}_{n,n}$, then it is true that

$$\begin{aligned} d^2 &= |(\mathcal{A}+\mathcal{I}_{n,n})||(\mathcal{A}+\mathcal{I}_{n,n})| = |(\mathcal{A}+\mathcal{I}_{n,n})(\mathcal{A}+\mathcal{I}_{n,n})| \\ &= |\mathcal{A}^2+\mathcal{A}+\mathcal{A}+\mathcal{I}_{n,n}| = |\mathcal{I}_{n,n}+\mathcal{I}_{n,n}| = 0 \end{aligned}$$

At this point, is it easy to check that $d = 0$. Thus, the matrix $(\mathcal{A}+\mathcal{I}_{n,n})$ is not invertible.

Using Proposition 1 one can define a simple form of constructing involutory elements from nilpotent elements with nilpotency index 2. It is well known that if $a \in \mathcal{R}$ is a nilpotent element then $\mathcal{R}a$ induces an ideal of $\mathcal{R}$. Let denote $D(2)$ the union of all the ideal $\mathcal{R}a$ where $a$ is a nilpotent element with nilpotency index 2 and let $b \in D(2)$ be a fixed element, then for any $r \in \mathcal{R}$ such that $rb \neq 0$ the element

$$a = e + rb \tag{2}$$

is an involutory element. Hence, any transformation $\Psi : \mathcal{R} \to \mathcal{R}$ constructed as

$$\Psi(x) = x(e + rb) \tag{3}$$

is a linear involutory transformation. The above relation opens an alternative way to construct involutory transformations. Instead of directly search for involutory elements, we can now search for nilpotent elements with nilpotency index 2.

## 2 Involutions as the result of the multiplication of polynomials modulus a polynomial with coefficients in a finite field

In the previous section a relation between involutory and nilpotent elements with nilpotency index 2 was established. Through this section we

construct linear involutory transformations as the result of the multiplication of polynomials within a polynomial quotient ring.

### 2.1 Existence criteria over rings of any characteristic

Recall the initial notations where $\mathcal{P} = \mathbb{F}_q$ is a finite field of $q = p^t$ elements being $p$ a prime number and $t \in \mathbb{N}$. We also have that $\mathcal{R} = P[x]/_{F(x)}$, $F(x) \in P[x]$. Considering that $F(x)$ has canonical factorization

$$F(x) = f_1(x)^{k_1} \cdots f_s(x)^{k_s}$$

with $\mathsf{deg}(F(x)) = m$ and $\mathsf{deg}(f_i(x)) = m_i$, such that $m = k_1 m_1 + \cdots + k_s m_s$.

Let be $\mathcal{R}_1 = \mathcal{P}[x]/_{f_1(x)^{k_1}} \oplus \cdots \oplus \mathcal{P}[x]/_{f_t(x)^{k_t}}$ and let $\Phi$ be an application from $\mathcal{R}$ to $\mathcal{R}_1$ defined as follows:

$$\begin{aligned} \forall g(x) \in \mathcal{R}, g_i(x) &\equiv g(x) \mod f_i(x)_i^k, \ \forall i \in \overline{1,s} \\ \Phi(g(x)) &= (g_1(x), \ldots, g_s(x)) \end{aligned}$$

It should be noted that $|\mathcal{R}_1| = |\mathcal{R}|$. Given that polynomials $f_1(x)^{k_1}, \ldots, f_s(x)^{k_s}$ are coprimes, the equation system

$$\begin{aligned} y &\equiv p_1(x) \bmod f_1(x)^{k_1} \\ &\vdots \\ y &\equiv p_s(x) \bmod f_s(x)^{k_s} \end{aligned}$$

has unique solution. Hence, $\Phi$ is a surjective application of finite sets and using the stated in [1] we obtain that $\Phi$ is a bijective application. In this fashion we have the following lemma.

**Lemma 1.** *For the ring $\mathcal{R}$ hold the following decomposition.*

$$\mathcal{R} \cong \mathcal{P}[x]/_{f_1(x)^{k_1}} \oplus \cdots \oplus \mathcal{P}[x]/_{f_t(x)^{k_t}} \tag{4}$$

*Proof.* It is straightforward from the result of the previous analysis. □

As result of the previous lemma we enunciate the following proposition.

**Proposition 2.** *The amount of polynomials $h(x) \in \mathcal{P}[x]$ such that $\mathsf{deg}(h(x)) < \mathsf{deg}(F(x))$ and $\gcd(h(x), F(x)) = 1$ is*

$$q^m \left(1 - \frac{1}{q^{m_1}}\right) \cdots \left(1 - \frac{1}{q^{m_t}}\right) \tag{5}$$

*Proof.* For all $i \in \{1, \ldots, t\}$ y $h(x) \in \mathcal{P}[x]/_{f_i(x)^{k_i}}$ hold that $gcd(h(x), f_i(x)^{k_i}) \neq 1$ if and only if $f(x)|h(x)$. Analyzing all elements $h(x) = h_{k_t-1}x^{k_t-1} + \cdots + h_1x + h_0$ and deleting their elements that is multiple of $h(X)$ giving

$$q^{m_i k_i} - q^{m_i(k_i-1)}.$$

It using decomposition 4 we give that

$$q^m \left(1 - \frac{1}{q^{m_1}}\right) \cdots \left(1 - \frac{1}{q^{m_t}}\right)$$

□

Proposition 2 gives the number of invertible elements w.r.t multiplication in $\mathcal{R}$. From its result we obtain the following criteria.

**Theorem 1.** *The ring $\mathcal{R}$ contains involutory elements iff one of the following conditions holds*

1. *$p$ is an odd value*
2. *If $p = 2$ in the canonical factorization of $F(x)$ exists $i \in \{1, \ldots, t\}$ such that $k_i \geq 2$.*

*Proof.* From [1] we have that if $G$ is an abelian group of order $n$ such that $2|n$ the $G$ contains a subgroup of order 2. In our case, due the structure of $\mathcal{R}$ the algebra $(\mathcal{R}^*, *)$ form an abelian group. This way we have that:

1. If $p$ is odd, the value of the formula determined in (5) is even, thus 2 is a divisor of $|\mathcal{R}^*|$ and therefore $\mathcal{R}^*$ contains subgroups of order 2 $\{r, e\}$ being $r$ an involutory element;
2. If $p = 2$ then the value of the formula determined in (5) is even if exist $i \in \{1, \ldots, s\}$ such that $k_i > 1$ we have that equally exist a subgroup of 2 elements which contain an involutory element.

Therefore the proof is complete. □

From the result of Theorem 1 and the analysis conducted through this section we can enunciate the following corollaries.

**Corollary 2.** *For any $n \in \mathbb{N}$, $n > 1$ it is possible to build a ring $\mathcal{R}$ of $p^{tn}$ elements which contain involutory elements.*

*Proof.* The existence of the values of $p$ and $t$ is ensured by finite fields theory [1, 10]. The size of $\mathcal{R}$ is $p^{tn}$ where $n$ represent the degree of the polynomial. Taking the polynomial $F(x) = (x + a)^n$ where $a \in \mathcal{P}$, $a \neq 0$ then by Therorem 1 we have that $\mathcal{R}$ contains involutory elements. □

**Corollary 3.** *For any $n \in \mathbb{N}$ such that $n > 1$ exist one linear involutory transformation*

$$\Psi : \mathcal{R} \to \mathcal{R}$$

*build by the rule in (1).*

*Proof.* For the demonstration of the corollary is enough to use the result from the previous corollary and the definition of the function $\Psi$ according to rule (1). □

## 2.2 Construction of linear involutory transformations using polynomials over rings of even characteristic and their branch number

From the result of Theorem 1 we settle the basis to construct linear involutory transformations, although the real question lies in how to detect the involutory elements to form such transformations. Notice that if the order of $F(x)$, denoted by $e$, is even, then the application $\Psi : \mathcal{R} \to \mathcal{R}$ of the form

$$\Psi(g(x)) = g(x)x^{\frac{e}{2}} \bmod F(x) \tag{6}$$

is involutory. As we already mentioned, the involutory and nilpotent elements with nilpotency index 2 are closely related. In the remaining of the section we are dedicated to the construction of involutory transformations using such relation and the rule from (1) where $\mathcal{R}$ is a ring of characteristic 2. We also provide the branch number of such transformations.

Hereinafter we consider, without loss of generality, that $k_1 > 1$. It is known that $F(x) \bmod F(x) = [0]_{F(x)}$ and since $F(x) = f_1(x)^{k_1} \cdots f_s(x)^{k_s}$ then we have that

$$\left(f_1(x)^{\lceil \frac{k_1}{2} \rceil} \cdots f_s(x)^{k_s}\right)^2 \bmod F(x) = [0]_{F(x)}.$$

where $\lceil\ \rceil$ denotes the ceil function. The previous relation implies that the element $f(x) = f_1(x)^{\lceil \frac{k_1}{2} \rceil} \cdots f_s(x)^{k_s}$ is a nilpotent element of $\mathcal{R}$ with nilpotency index 2. From this element one can construct involutory elements using rule (1) taking $r$ as an arbitrary element $h(x) \in \mathcal{R}$. For the sake of simplicity, the elements $[h(x)]_{F(x)}$ will be written simply as $h(x)$. In addition, the element $\mu(x)$ of the linear transformation $\Psi$ is of the form $\mu(x) = 1 + h(x)f(x)$ for $\Psi$ to be involutory. Thus, $\Psi$ is expressed

$$\Psi(g(x)) = g(x)(1 + f(x)h(x)) \bmod F(x). \tag{7}$$

A parameter associated to linear transformations is the branch number. In order to define this parameter, which is usually used for vectors, it is necessary to introduce the function $\tau : \mathcal{R} \to \mathbb{N}_0$ defined as:

$$\tau(g(x)) = \sum_{i=0}^{m-1} \delta_{0,f_i}$$

where

$$\delta_{0,f_i} = \begin{cases} 1, & \text{if } f_i \neq 0 \\ 0, & \text{if } f_i = 0, \end{cases}$$

i.e., $\delta_{0,f_i}$ is the number of nonzero coefficients of an element of the ring $\mathcal{R}$. In the case of rings with characteristic 2 we have the following relation

$$\delta_{0,f_i} = N_2^{2^t}(f_i),$$

where $N_2^{2^t}(f_i)$ represents the norm of the element $f_i$ in the prime field.

**Definition 3.** *The branch number $\rho$ of a linear transformation $\Psi : \mathcal{R} \to \mathcal{R}$ is defined as*

$$\rho(\Psi) = \min_{g(x)\neq 0} \{\tau(g(x)) + \tau(\Psi(g(x)))\}.$$

*The linear transformation $\Psi$ is called MDS if $\rho(\Psi) = m + 1$.*

**Proposition 3.** *Let $(\alpha_0, \ldots, \alpha_s)$ be elements of the set $\mathcal{P}^*$, such that $\forall i, j \in \{0, \ldots, s\}$ $\alpha_i \neq \alpha_j$ and let be the polynomials such that*

$$f(x) = (x + \alpha_0)^{n-1}(x + \alpha_2) \cdots (x + \alpha_s) \text{ and } F(x) = (x + \alpha_0) f(x).$$

*For all element $h(x) \in \mathcal{R}$ the transformation $\Psi$ defined by the rule (7) is a linear involutory transformation whose branch number satisfies that*

$$\rho(\Psi) \leqslant 4.$$

*If the matrix associated to $\Psi$ does not contain any zero coefficient then*

$$\rho(\Psi) = 4.$$

*Proof.* For the construction of $\Psi$ and the element $g(x) = (x + \alpha_0) \in \mathcal{R}$ is true that

$$\Psi(g(x)) = g(x)(1 + f(x)h(x)) = g(x),$$

and, given that $\tau(g(x)) = \tau(\Psi(g(x))) = 2$ we obtain that $\rho(\Psi) \leq 4$.

For demonstration of the second statement of the proposition one have to notice that is only necessary to analyze the vectors having one or two nonzero coordinates. Firstly, we construct the basis $1, x, \ldots, x^{n-1}$ for the matrix $\mathcal{A}$

associated to the transformation $\Psi$ [1]. If all the elements of the matrix are distinct from zero then all vectors which have a coordinate distinct from zero will be mapped to elements will all nonzero coordinates, the vectors which have two coordinates distinct from zero will be mapped to vectors having two or more coordinates distinct from zero. Thus, $\rho(\Psi) = 4$. □

**Proposition 4.** *Let $a$ be element of field $\mathcal{P}$ such that $a \neq 1$, and let be the polynomials such that*

$$f(x) = (x^n + a) \text{ and } F(x) = f(x)^2.$$

*For all element $h(x) \in \mathcal{R}$ the transformation $\Psi$ form by the rule (7) is a linear involutory transformation whose branch number satisfies that*

$$\rho(\Psi) \leqslant 4.$$

*If the matrix associated to $\Psi$ does not contain any zero coefficient then*

$$\rho(\Psi) = 4.$$

*Proof.* The proof of the proposition is similar to the one of Proposition 3. □

Although within the classes, shown in the previous propositions, does not exist involutory MDS matrices of sizes $4 \times 4$ or greater, it does not imply that they don't exist within any other class. Next, we show two examples of polynomials outside of these classes from which one can construct involutory MDS matrices of sizes $4 \times 4$ and $6 \times 6$.

### Example 1

Take $\mathcal{P} = \mathbb{F}_2[x]/_{x^8+x^4+x^3+x^2+1}$ having $\beta$ a primitive element on $\mathcal{P}$ and selecting the following parameters

| $n$ | $f(x)$ | $F(x)$ | $h(x)$ |
|---|---|---|---|
| 4 | $(x+\beta)(x+\beta^2)$ | $f(x)^2$ | $x$ |
| 6 | $(x+\beta)(x+\beta^2)(x+\beta^3)$ | $f(x)^2$ | $x^2+x$ |

the transformation

$$\Psi(g(x)) = g(x)(1 + f(x)h(x)) \bmod F(x)$$

from equation (7) is an involutory MDS transformation.

# 3 Generation of $4 \times 4$ involutory MDS matrices over $\mathbb{F}_{2^t}$

In the preceding section we show that within the class of functions $\Psi$ such that

$$\Psi(g(x)) = g(x)(1 + (x^n + a)h(x)) \bmod x^{2n} + a^2,$$

$\forall a \in \mathcal{P}^*$ y $\forall\, h(x) \in \mathcal{R}$ does not exist MDS matrices. In this section we show how to generate MDS matrices over a finite field of the form $\mathbb{F}_{2^t}$ using the previous construction and alternating between two different rings where the polynomial $F(x) = (x + a)^4$.

## 3.1 Construction of sets of involutory $4 \times 4$ MDS matrices over $\mathbb{F}_{2^t}$

Let $\mathcal{P} = \mathbb{F}_{2^t}$, given $a \in \mathcal{P}^*$, $f_1(x) = x^n + a$, $F_1(x) = x^{2n} + a^2$, $f_2(x) = x^n + 1$, $F_2(x) = x^{2n} + 1$, we construct the transformations $\Psi_1$ and $\Psi_2$ from $\mathcal{P}_i$ to $\mathcal{P}_i$ where $\mathcal{P}_i = P[x]/_{F_i(x)}$

$$\Psi_1(g(x)) = g(x)(1 + f_1(x)h(x)) \bmod F_1(x)$$

and

$$\Psi_2(g(x)) = g(x)(1 + f_2(x)x) \bmod F_2(x).$$

Let us denote by " $\circ$ " the composition of two applications [1] and by $\mathcal{R}(2n)$ the set of all polynomials with coefficients in $\mathcal{P}$ whose degree is lower than $2n$. We define the transformation

$$\Psi : \mathcal{R}(2n) \to \mathcal{R}(2n)$$

of the following form

$$\forall g(x) \in \mathcal{R}(2n),\ \Psi(g(x)) = \Psi_1 \circ \Psi_2 \circ \Psi_1(g(x)). \tag{8}$$

It is easy to see that transformation $\Psi$ is linear and involutory [1]. A similar method to construct involutory matrices was applied in [11] to generate new involutory matrices from previous involutory transformations.

The search for MDS matrices is closely related to the definition of the polynomial $h(x) = h_3x^3 + h_2x^2 + h_1x + h_0$. For determining such matrices let us consider that the values of $h_3$, $h_2$ and $h_1$ were selected leaving $h_0$ as unknown. Using the basis $1, x^2, \ldots, x^{2n-1}$ we construct the matrix $\mathcal{A}_{h_0}$ associated to transformation $\Psi$ as follows

$$\Psi(x^i) = \delta_i(x) = \delta_{i,4}(h_0)x^3 + \delta_{i,3}(h_0)x^2 + \delta_{i,2}(h_0)x + \delta_{i,1}(h_0)$$

Finally, matrix $\mathcal{A}_{h_0}$ is of the form

$$\mathcal{A}_{h_0} = \begin{pmatrix} \delta_{1,1}(h_0) & \dots & \delta_{1,2n}(h_0) \\ & \dots & \\ \delta_{2n,1}(h_0) & \dots & \delta_{2n,2n}(h_0) \end{pmatrix}$$

Notice that every polynomial $\delta_{i,j}(h_0)$ have, at most, degree equal to 2.

In order to determine if a matrix is MDS it is necessary to discard all roots of the polynomials $\hat{\delta}_{i,j}(h_0)$ formed after calculating the different matrix minors. It is clear that the amount of roots should not exceed the size of the multiplicative group of the field for the existence of possible values.

Let review the case of $4 \times 4$ matrices. We have that the 36 minors of order 2 form equations of degree 4, thus it is necessary to remove $36 \times 4 = 144$ elements. Similarly we have to remove all elements of degree 2 which are $16 \times 2 = 32$ leaving that the total number of elements to be removed is at most 176. Given that $p^t - 176 > 0$ then for $p = 2$ we have that in $\mathbb{F}_{2^8}$ exist MDS matrices which are associated to transformation $\Psi$. In addition, is well known the following proposition related to MDS matrices.

**Proposition 5** ([1])**.** *Let $\mathcal{A}$ be an arbitrary matrix. We say that $\mathcal{A}$ is MDS if the following statements hold:*

1. *All the coefficients of $\mathcal{A}$ are distinct from 0.*
2. *All the coefficients of $\mathcal{A}^{-1}$ are distinct from 0.*
3. *All order 2 minors are distinct from 0.*

Let $\mathcal{A}_{h_0}$ be the matrix associated to transformation $\Psi$ as defined above. Since $\mathcal{A}_{h_0}$ is an involutory matrix then we have that only checking that all its coefficients are distinct from zero and that this holds for all of its order 2 minors, then we can say $\mathcal{A}_{h_0}$ is a MDS matrix.

**Proposition 6.** *Let be $n = 4$. For a given polynomial $h(x)$ such that $h_1 \neq ah_3$ and for any value of $h_2$ there exist a value of $h_0$ such that the matrix associated to transformation $\Psi$ is MDS. Particularly one can take $h_1 = ah_3 + 1$.*

*Proof.* For the basis $\left(1, x, x^2, x^3\right)$ we construct the associate matrix for the application. Using the previous analysis, we obtain the proof of the proposition. □

The procedure to obtain the polynomial $h(x)$ from the knowledge of the values of $a \in \mathcal{P} \setminus \{0\}$ and $h_2, h_3 \in \mathcal{P}$ where $h_0$ remains unknown is as follows:

**INPUT:** $a \in \mathcal{P} \setminus \{0\}$ and $h_2, h_3 \in \mathcal{P}$

**STEP 1:** Make $h_1 = ah_3 + 1$, $i = 1$.

**STEP 2:** Make the i-th row of the matrix $\mathcal{A}_{h_0}$, $\delta_i(x) = \Psi(x^{i-1})$.

**STEP 3.1:** Discard all roots of the polynomials $\delta_{i,1}(h_0), \ldots, \delta_{i,4}(h_0)$.

**STEP 3.2:** Make $i = i + 1$

**STEP 3.3:** If $i = 5$ go to **STEP 4**, otherwise go to **STEP 2**.

**STEP 4:** Discard from $\mathcal{A}_{h_0}$ all the roots of the polynomials result from calculating its order 2 minors.

**RETURN:** The set of values for $h_0$ such that the polynomials $h(x) = h_3x^3 + h_2x^2 + h_1x + h_0$ generate an involutory MDS matrix.

The methodology followed through this section ensure that the set of possible values of $h_0$ is not empty and therefore exist some polynomial $h(x)$ such that $\mathcal{A}_{h_0}$ is MDS as shown in the next example.

### Example 2

Make $\mathcal{P} = \mathbb{F}_2[x]/_{x^8+x^4+x^3+x^2+1}$ having $\beta$ a primitive element on $\mathcal{P}$ and selecting the following parameters:

- $f(x) = x^2 + \beta$
- $F(x) = f(x)^2$
- $h_3 = 1$
- $h_2 = 0$

Applying the above procedure we obtain 230 distinct values of $h_0$ such that the polynomial $h(x) = h_3x^3 + h_2x^2 + h_1x + h_0$ generate an involutory MDS matrix, i.e., we obtained 230 distinct involutory MDS matrices for the aforementioned configuration. This was checked using the mathematical software SAGE [12]. According to our analysis, the least amount of matrices we can obtain using the described procedure was 80. The amount that we obtain in the example is given by the fact that the majority of the obtained polynomial after calculating the order one and two minors does not have roots. Moreover, if we assume that for every possible value of $a \in \mathcal{P} \setminus \{0, 1\}, h_1 \neq ah_3, h_2, h_3$ the set of values of $h_0$ have the same size as in our example then one can

obtain, at most, approximately $2^{40}$ involutory MDS matrices using our procedure.

The method presented in this section to generate involutory MDS matrices can be also used to generate key-dependent MDS matrices for dynamic encryption algorithms such the transformation of AES presented in [13]. It should be pointed that, although the search space for involutory MDS matrices is lower than the general MDS matrix's search space, a matrix generated using our proposal remove the necessity of generating the inverse matrix for the decryption process, thus reducing the time complexity of creating key-dependent components for the encryption algorithm.

## Conclusions

In this paper we board the construction of involutory elements using the nilpotent elements of a ring with nilpotency index 2. From such construction we study the involutory linear applications generated by multiplication of polynomials with coefficients in a finite field, characterizing the branch number of such applications. Finally, we introduce a new proposal which allow to generate MDS matrices using the results of the paper towards involutory elements and alternating between two different rings. However, how to reduce from quadratic to linear equations, which allow to increase the dimension of the matrix, remains as open question and future line of work in our research.

## References

[1] Glujov M. M., Elizarov V. P. and Nechaev A. A., *Algebra*, LAN Ed., 2015.

[2] NIST, "Advanced Encryption Standard", *Federal Information Processing Standard, FIPS-197*, 2001.

[3] GOST R 34.12-2015, "Information technology. Cryptographic protection of information. Block ciphers", *Standartinform*, Moscow, 2015.

[4] Guo J., Peyrin T. and Poschmann A., "The PHOTON family of lightweight hash functions", *Advances in Cryptology–CRYPTO 2011: 31st Annual Cryptology Conference*, Springer, 2011, pp 222–239.

[5] Beaulieu R., *et. al*, "The SIMON and SPECK lightweight block ciphers", *Proceedings of the 52nd annual design automation conference*, 2015, pp 1–6.

[6] Barreto P. and Rijmen V., "The Khazad legacy-level block cipher", *Primitive submitted to NESSIE*, **97 106** (2000).

[7] Barreto P., "The Anubis block cipher", *Primitive submitted to NESSIE*, 2000.

[8] Chand Gupta K. and Ghosh Ray I., "On constructions of involutory MDS matrices", *Progress in Cryptology–AFRICACRYPT 2013: 6th International Conference on Cryptology in Africa*, Springer, 2013, pp 43–60.

[9] Elizarov V. P., *Finite Rings*, Gelios APB Ed., 2006.

[10] Lidl R. and Niederreiter H., "Introduction to finite fields and their applications", *Cambridge University Press*, 1994.

[11] Aulet R. R., de la Cruz Jiménes R. A., "Construction of MDS matrices combining the Feistel, Misty and Lai-Massey schemes", *Mat. Vopr. Kriptogr.*, **12**:2 (2021), 57–74.

[12] *Sage Mathematics Software (Version 8.1)*, 2018, `http://www.sagemath.org`.

[13] Freyre P., Cuellar O., Díaz N. and Alfonso A., "From AES to Dynamic AES", *Journal of Science and Technology on Information security*, **1 11** (2020), 11–22.

# Symmetric Cryptography
# Permutations

# Class of piecewise-monomial mappings: differentially 4-uniform permutations of $\mathbb{F}_{2^8}$ with graph algebraic immunity 3 exist

Dmitry Burov[1], Sergey Kostarev[2], and Andrey Menyachikhin[1]

[1]TVP Laboratories, Russia
[2]Secure Information Technology Assistance Foundation, Russia
dmitry.burov@inbox.ru, kursovic57@yandex.ru, and88@list.ru

**Abstract**

In this paper, we propose a new class of piecewise-monomial mappings. Some cryptographic characteristics of piecewise-monomial mappings are provided. We experimentally show the influence of the automorhism group on the differential $\delta$-uniformity and the nonlinearity of mappings. Using adapted spectral-differential method, we find differentially 4-uniform permutations of $\mathbb{F}_{2^8}$ with the graph algebraic immunity 3.

**Keywords:** permutations of finite fields, $S$-box, piecewise-monomial mappings, differential $\delta$-uniformity, nonlinearity, graph algebraic immunity, automorphism group, adapted spectral-differential method.

## 1 Introduction

Nonlinear mappings (S-boxes) are widely used in modern stream and block ciphers, hash functions. S-box has high resistance against differential and linear cryptanalysis in case it has the low differential uniformity and the high nonlinearity. There are a lot of methods constructing s-boxes, for example: algebraic methods [7], [15], heuristic methods and algorithm optimization [8], [13], using lower-dimensional s-boxes [2], [6], [11], [12].

Using algebraic methods we can construct mappings with optimal or almost optimal cryptographic characteristics. For example, the inverse permutation on the finite field $\mathbb{F}_{2^8}$ (using in AES, Camellia, Aria) has the best-known differential uniformity 4 and the the best-known nonlinearity 112 and the highest possible algebraic degree 7 among all known permutations on $\mathbb{F}_{2^8}$. However, algebraic mappings may have some weaknesses. For example, the inverse permutation has graph algebraic immunity 2 [4], which can be used in XSL-method [5]. The following problem existing of differentially 4-uniform

permutations with the graph algebraic immunity 3 on $\mathbb{F}_{2^n}$ when $n = 8$ remains open [3], [16].

Using heuristic methods and algorithm optimization methods we can construct permutations with a little worse characteristics. For example, using spectral-linear and spectral-differential methods we can construct permutations of $\mathbb{F}_{2^8}$ with the differential uniformity 6 and the nonlinearity 104 [13], while the inverse permutation of $\mathbb{F}_{2^8}$ has the differential uniformity 4 and the nonlinearity 112. Permutations that were found by random searching have no algebraicity. In particular, permutations constructed with spectral-differential method mostly have graph algebraic immunity 3.

Almost all algebraic mappings have the nontrivial automorphism group. In [1] it is experimentaly shown that all known 2-differentially uniform mappings (APN-mappings) have the nontrivial automorphism group for low values of $n$. The inverse permutation of $\mathbb{F}_{2^n}$ also has nontrivial automorphism group.

In [9], [10] the authors study methods for constructing permutations with the automorphism group containing cyclic shifts of the vector coordinates.

In [17] the class of piecewise-linear permutations of $\mathbb{F}_{2^n}$ is proposed. This class of permutations also has the nontrivial automorphism group. In [14] spectral-differential method has been adapted to construct permutations with the low differential uniformity. Using this method, we can construct differential 4-uniformity piecewise-linear permutations of $\mathbb{F}_{2^8}$. Also we notice that the random search with $10^8$ iterations gives no results [14].

In this paper, we introduce the class of $k$-piecewise-monomial mappings of finite field $\mathbb{F}_{2^n}$. The restriction of $k$-piecewise-monomial mapping to each coset $H_i$ of fixed subgroup $H$ in $\mathbb{F}_{2^n}^*$ has monomial structure: $x \mapsto x^k A_i$, where $x \in H_i$, $A_i \in \mathbb{F}_{2^n}$, $i = \{0, \ldots, r_H - 1\}$, $r_H = \frac{2^n-1}{|H|}$. When $k = 1$ this class equals the piecewise-linear class of mappings. Some cryptographic characteristics $k$-piecewise-monomial mappings are provided.

The paper is organized as follows. In section 2, we introduce main definitions and notations.

In section 3, we define $k$-piecewise-monomial mappings. We prove a necessary and sufficient condition for a $k$-piecewise-monomial mapping to be a bijection. Also we show that the automorphism group of a $k$-piecewise-monomial mapping, which restrictions to cosets of $H = \langle\zeta\rangle$ in $\mathbb{F}_{2^n}^*$ are monomials, contains the group $\langle (x, y) \mapsto (x\zeta, y\zeta^k) \,|x, y \in \mathbb{F}_{2^n} \rangle$.

In sections 4 and 5, we consider two subclasses of $k$-piecewise-monomial mappings whose the automorphism group is strictly bigger than the group $\langle (x, y) \mapsto (x\zeta, y\zeta^k) \,|x, y \in \mathbb{F}_{2^n} \rangle$. In section 4, we describe mappings which

belong to both $k$-piecewise-monomial mappings on $H$ and $k'$-piecewise-monomial mappings on $H'$ classes. The subgroup of the automorphism group piecewise-monomial mappings also is described. In section 5, we describe conditions for a choice of elements $A_0, \ldots, A_{r_H-1}$, when the automorphism group of a $k$-piecewise-monomial mapping contains the transformation $(x, y) \mapsto (x^{2^m}, y^{k2^m})$, $x, y \in \mathbb{F}_{2^n}$.

In section 6, we experimentally show the influence of the automorhism group on the differential $\delta$-uniformity and the nonlinearity of mappings. Using adapted spectral-differential method, we find differentially 4-uniform permutations of $\mathbb{F}_{2^8}$ with graph algebraic immunity 3.

## 2 Definitions and notations

Denote by $\mathbb{F}_{2^n}$ the finite field of $2^n$ elements, $\mathbb{F}_{2^n}^* = \mathbb{F}_{2^n} \setminus \{0\}$, $tr_n \colon x \mapsto x + x^2 + \ldots + x^{2^{n-1}}$ — the absolute trace function from $\mathbb{F}_{2^n}$ to $\mathbb{F}_2$.

**Definition 1.** *A mapping $f \colon \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ is called differentially $\delta$-uniform if the equation $f(x+a) + f(x) = b$ has at most $\delta$ solutions for every $a \in \mathbb{F}_{2^n}^*$, $b \in \mathbb{F}_{2^n}$. Minimal $\delta$ with this property is called the differential uniformity of $f$ and denoted by $\delta(f)$.*

**Definition 2.** *A nonlinearity of a mapping $f \colon \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ is called a value*

$$nl(f) = 2^{n-1} - \frac{1}{2} \max_{u \in \mathbb{F}_{2^n}, v \in \mathbb{F}_{2^n}^*} |W_f(u, v)|,$$

*where* $W_f(u, v) = \sum\limits_{x \in \mathbb{F}_{2^n}} (-1)^{tr_n(f(x)v + xu)}$.

S-boxes with the low differential $\delta$-uniformity and the high nonlinearity increase the resistance of block ciphers against differential and linear cryptanalysis.

**Definition 3.** *A value*

$$\deg(f) = \min_{\alpha \in \mathbb{F}_{2^n}^*} \deg\left(tr_n\left(\alpha f\left(x\right)\right)\right),$$

*is called the algebraic degree of a mapping $f \colon \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ and denoted by* $\deg(f)$ *where* $\deg\left(tr_n\left(\alpha f\left(x\right)\right)\right)$ *is the degree of the ANF of a boolean function $tr_n\left(\alpha f\left(x\right)\right)$. If $f \in S\left(\mathbb{F}_{2^n}\right)$ than a value $\overline{\deg}(f) = \min\left\{\deg(f), \deg\left(f^{-1}\right)\right\}$ is called the generalized algebraic degree of a permutation $f$.*

S-boxes with the high generalized algebraic degree increase the resistance of block ciphers against algebraic methods of cryptanalysis.

**Definition 4.** *A graph algebraic immunity of mapping $f\colon \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ is the algebraic immunity of the graph $\{(x, f(x)) \,|x \in \mathbb{F}_{2^n})\}$, and it is denoted by $AI(f)$.*

S-boxes with the high graph algebraic immunity increase the resistance of block ciphers against XSL-method [5].

**Definition 5** ([1])**.** *The set*

$$\mathrm{Aut}(f) = \{\sigma \in AGL_{2n}(2) \,|\{\sigma(x, f(x)) \,|x \in \mathbb{F}_{2^n}\} = \{(x, f(x)) \,|x \in \mathbb{F}_{2^n}\}\}$$

*is called the automorphism group of a mapping $f\colon \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$.*

By $r_{G,H}$ we denote the index of a subgroup $H$ in the group $G$. For $r_{\mathbb{F}_{2^n}^*,H}$ we use the notation $r_H$. Denote the greatest common divisor of integers $a$ and $b$ by $GCD(a, b)$.

## 3 Class $PM_{n,k}(H)$ $k$-piecewise monomial mappings of the finite field

Let us consider the partitions $\mathbb{F}_{2^n}^* = \bigcup\limits_{i=0}^{r_H-1} H_i = \bigcup\limits_{i=0}^{r_S-1} S_i = \bigcup\limits_{i=0}^{r_D-1} D_i$ of $\mathbb{F}_{2^n}^*$ into cosets of subgroups $H, S, D = \langle H, S\rangle$.

**Definition 6.** *A mapping $f\colon \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ is called $k$-piecewise-monomial on $H < \mathbb{F}_{2^n}^*$ if the following properties hold:*

*1) $f(0) = 0$,*

*2) there are $A_0, A_1, \ldots, A_{r_H-1} \in \mathbb{F}_{2^n}$ such that for every $x \in H_i$ the equality $f(x) = x^k A_i, i \in \{0, \ldots, r_H - 1\}$, holds.*

*Denote the set of $k$-piecewise-monomial on $H$ mappings by $PM_{n,k}(H)$.*

Let's show that automorphism group of mappings from $PM_{n,k}$ is nontrivial.

**Proposition 1.** *Let $H = \langle\zeta\rangle$, $k \in \{0, \ldots, n-1\}$, then $f \in PM_{n,k}(H)$ if and only if mapping $(x, y) \mapsto (x\zeta, y\zeta^k)$, $x, y \in \mathbb{F}_{2^n}$, is an automorphism of $f$.*

*Proof.* Since $x\zeta \in H_i$ for $x \in H_i$, it follows that

$$f(x\zeta) = x^k \zeta^k A_i = f(x)\zeta^k, x \in H_i,$$

Let $A_i = f(\theta^i)\theta^{-ik}$, where $\langle\theta\rangle = \mathbb{F}_{2^n}$. Since mapping $(x, y) \mapsto (x\zeta, y\zeta^k)$, $x, y \in \mathbb{F}_{2^n}$, is an automorphism of $f$, it follows that for ordinary element $\theta^i\zeta^j$ of coset $H_i = \theta^i H$ we have

$$f(\theta^i h) = f(\theta^i\zeta^j) = f(\theta^i)\zeta^{jk} = f(\theta^i)\theta^{-ik}\theta^{ik}\zeta^{jk} = (\theta^i\zeta^j)^k A_i = (\theta^i h)^k A_i.$$

It follows that $f \in PM_{n,k}(H)$. This concludes the proof. □

Let us prove a necessary and sufficient condition for a $k$-piecewise-monomial mapping to be a bijection.

**Proposition 2.** *Let $f \in PM_{n,k}(H)$, $f(x) = x^k A_i, x \in H_i$, where $H < \mathbb{F}_{2^n}^* = \langle\theta\rangle$, $H_i = H\theta^i$, $A_i = \theta^{a_i}$, $a_i \in \{0, \ldots, 2^n - 2\}$, $i = 0, \ldots, r_H - 1$. Then $f$ is a bijection if and only if $GCD(k, |H|) = 1$ and the set $\{a_i + ik \,|i \in \{0, \ldots, r_H - 1\}\}$ is a complete set of residues modulo $r_H$.*

*Proof.* We have $f(H_i) \subseteq H_{a_i+ki}$, $i \in \{0, \ldots, r_H - 1\}$, where index $a_i + ki$ is taken modulo $r_H$. Hence $f$ is a bijection if and only if the following conditions are satisfied:

1) $f(H_i) = H_{a_i+ki}$,

2) $f(H_i) \neq f(H_j)$ for all $i, j \in \{0, \ldots, r_H - 1\}$, $i \neq j$.

Condition 1 is true if and only if $\mathrm{GCD}(k, |H|) = 1$. Condition 2 is true if and only if the set $\{a_i + ik \,|i \in \{0, \ldots, r_H - 1\}\}$ is a complete set of residues modulo $r_H$. □

Let us describe values of $k, k'$ for which the equality $PM_{n,k}(H) = PM_{n,k'}(H)$ holds.

**Proposition 3.** *Let $H < \mathbb{F}_{2^n}^*$, $k, k' \in \mathbb{N}$, then the following conditions are equivalent:*

*1) $PM_{n,k}(H) \cap PL_{n,k'}(H) \neq \{\mathbb{O}\}$, where $\mathbb{O}\colon x \mapsto 0$, $x \in \mathbb{F}_{2^n}$,*

*2) $|H| \mid k - k'$,*

*3) $PM_{n,k}(H) = PM_{n,k'}(H)$.*

*Proof.* Let us show that the implication 1) $\Rightarrow$ 2) is true. Let $f \in PM_{n,k}(H) \cap PM_{n,k'}(H)$, $f \neq \{\mathbb{O}\}$, then there are $j \in \{0, \ldots, r_H - 1\}$, $A_j, B_j \in \mathbb{F}_{2^n}^*$ such that the equality $f(x) = x^k A_j = x^{k'} B_j$ holds for all $x \in H_j$. It follows that the equality $x^{k-k'} = B_j A_j^{-1}$ holds for all $x \in H_j$. Since $x = h\theta^j$, $h \in H$, we obtain that $h^{k-k'} = 1$. Hence we have $|H| \mid k - k'$.

Let us show that the implication 2) $\Rightarrow$ 3) is true. Let $f \in PM_{n,k}(H)$, $f\colon x \mapsto x^k A_j$, $x \in H_j$, $A_j \in \mathbb{F}_{2^n}$, $j = 0, \ldots, r_H - 1$. Since an arbitrary element $x \in H_j$ has the form $x = h\theta^j$, $h \in H$, it follows that

$$f(x) = x^k A_j = h^k \theta^{kj} A_j = h^{k'} \theta^{k'j} \theta^{(k-k')j} A_j = x^{k'} B_j, \quad x \in H_j,$$

where $B_j = \theta^{j(k-k')} A_j$. It follows that $f \in PM_{n,k'}(H)$ and $PL_{n,k}(H) \subseteq PM_{n,k'}(H)$. Similarly as above we can prove the reverse inclusion.

Implication 3) $\Rightarrow$ 1) is obvious. □

Proposition 3 implies that it is enough to study $k$-piecewise-monomial mappings when $k \in \{0, \ldots, |H| - 1\}$. Mappings $f$ and $g\colon x \mapsto f(x)^2$ are linear-equivalent and if $f \in PM_{n,k}(H)$ then $g \in PM_{n,2k}(H)$, where index $2k$ is taken modulo $r_H$. Hence if $k, k'$ belong to the same cyclotomic class modulo $|H|$ then the differential $\delta$-uniformity and the nonlinearity of a function $f \in PM_{n,k}(H)$ equal correspondingly to the differential $\delta$-uniformity and the nonlinearity of an appropriate function $g \in PM_{n,k'}(H)$. Therefore, it is enough to study sets $PM_{n,k}(H)$ up to representatives of cyclotomic class modulo $|H|$.

## 4 Decscription of $PM_{n,k}(H) \cap PM_{n,k'}(S)$ class

Here we consider mappings from $PM_{n,k}(H) \cap PM_{n,k'}(S)$. For this reason we use the double numeration for cosets of subgroups $H, S < \mathbb{F}_{2^n}^*$.

**Definition 7.** *Let $S, H, D = \langle H, S\rangle = \langle\psi\rangle < \mathbb{F}_{2^n}^* = \langle\theta\rangle$, $\psi = \theta^{r_D}$. Let us define the following numeration of cosets:*

$$\begin{aligned} D_i &= D\theta^i, \quad i = 0, \ldots, r_D - 1, \\ H_{i,j} &= H\theta^i\psi^j, \quad i = 0, \ldots, r_D - 1, j = 0, \ldots, r_{D,H} - 1, \\ S_{i,j} &= S\theta^i\psi^j, \quad i = 0, \ldots, r_D - 1, j = 0, \ldots, r_{D,S} - 1. \end{aligned}$$

It's easy to see that

$$D_i = \bigcup_{j=0}^{r_{D,H}-1} H_{i,j} = \bigcup_{j=0}^{r_{D,S}-1} S_{i,j}, \quad i = 0, \ldots, r_D - 1.$$

**Lemma 1.** *Let $S, H, D = \langle H, S\rangle = \langle\psi\rangle < \mathbb{F}_{2^n}^* = \langle\theta\rangle$, and the numeration of cosets is defined by definition 7, $F = H \cap S = \langle\xi\rangle$, $\xi = \theta^{r_F} = \psi^{r_{D,F}}$, $\gamma_{H,S} = r_{D,S}^{-1} \mod r_{D,H}$. Then for all $j = 0, \ldots, r_{D,H} - 1$, $j' = 0, \ldots, r_{D,S} - 1$ we have*

$$H_{0,j} \cap S_{0,j'} = F\psi^{r_{D,S}(j-j')\gamma_{H,S}} = F\theta^{r_S(j-j')\gamma_{H,S}}.$$

*Proof.* Using

$$H_{0,j} = \left\{\psi^{r_{D,H}t+j} \,|t = 0, \ldots, |H| - 1\right\},$$
$$S_{0,j'} = \left\{\psi^{r_{D,S}l+j'} \,|l = 0, \ldots, |S| - 1\right\},$$

we obtain that an element $\psi^{r_{D,S}l+j'}$ belongs to $H_{0,j} \cap S_{0,j'}$ if and only if $l$ is a solution of the congruence

$$r_{D,S}l + j' \equiv j \mod r_{D,H}. \tag{1}$$

Since

$$GCD(r_{D,S}, r_{D,H}) = GCD\left(\frac{|D|}{|S|}, \frac{|D|}{|H|}\right) =$$
$$= GCD\left(\frac{|H|}{GCD(|H|, |S|)}, \frac{|S|}{GCD(|H|, |S|)}\right) = 1,$$

it follows that $l = (j - j')\left(r_{D,S}^{-1} \mod r_{D,H}\right)$ is a solution of the congruence (1). □

Suppose that $f \in PM_{n,k}(H)$, $f\colon x \mapsto x^k A_{i,j}$, $x \in H_{i,j}$, $i = 0, \ldots, r_D - 1$, $j = 0, \ldots, r_{D,H} - 1$. Let us describe necessary and sufficient conditions for $A_{0,0}, \ldots, A_{r_D-1,r_{D,H}-1}$ which imply that $f$ belongs to $PM_{n,k'}(S)$.

**Theorem 1.** *Let $H, S, D = \langle H, S\rangle < \mathbb{F}_{2^n}^* = \langle\theta\rangle$, and the numeration of cosets is defined by definition 7. Let $f \in PM_{n,k}(H)$, $f\colon x \mapsto x^k A_{i,j}, x \in H_{i,j}$, $f \neq \mathbb{O}$, $\gamma_{H,S} = r_{D,S}^{-1} \mod r_{D,H}$. Then $f \in PL_{n,k'}(S)$, $f\colon x \mapsto x^{k'} B_{i,j}$, $x \in S_{i,j}$, if and only if $|H \cap S|$ divides $k' - k$ and for all $j_1, j_2 \in \{0, \ldots, r_{D,H} - 1\}$ the following equality holds*

$$A_{i,j_1} = \theta^{r_S(k'-k)(j_1-j_2)\gamma_{H,S}} A_{i,j_2}, i \in \{0, \ldots, r_D - 1\}. \tag{2}$$

*If the theorem's conditions are satisfied then the set of elements*

$$\{B_{i,j} \,|i \in \{0, \ldots, r_D - 1\}, j \in \{0, \ldots, r_{D,S} - 1\}\}$$

*is uniquely defined and can be obtained by the formula*

$$B_{i,j} = \theta^{(k-k')(i-r_S j\gamma_{H,S})} A_{i,0}.$$

*Proof.* Let $D = \langle\psi\rangle$, where $\psi = \theta^{r_D}$, $f \in PM_{n,k'}(S)$, $f \neq \mathbb{O}$,

$$f\colon x \mapsto x^{k'}B_{i,j}\ , x \in S_{i,j},$$
$$i \in \{0,\dots,r_D-1\}, j \in \{0,\dots,r_{D,S}-1\}.$$

We have $f \in PM_{n,k}(F) \cap PM_{n,k'}(F)$, where $F = H \cap S = \langle\xi\rangle$, $\xi = \theta^{r_F}$. Using proposition 3, we obtain that $|F|$ divides $k'-k$.

Let us choose arbitrary $x_{j_1} \in H_{0,j_1} \cap S_{0,0}$ and $x_{j_2} \in H_{0,j_2} \cap S_{0,0}$. It follows that $x_{j_1}\theta^i \in H_{i,j_1} \cap S_{i,0}$ and $x_{j_2}\theta^i \in H_{i,j_2} \cap S_{i,0}$. By the definition $f$ we have

$$f(x_{j_1}\theta^i) = (x_{j_1}\theta^i)^k A_{i,j_1}, f(x_{j_1}\theta^i) = (x_{j_1}\theta^i)^{k'}B_{i,0},$$
$$f(x_{j_2}\theta^i) = (x_{j_2}\theta^i)^k A_{i,j_2}, f(x_{j_2}\theta^i) = (x_{j_2}\theta^i)^{k'}B_{i,0},$$

The last equalities imply that

$$B_{i,0} = (x_{j_1}\theta^i)^{k-k'}A_{i,j_1}, B_{i,0} = (x_{j_2}\theta^i)^{k-k'}A_{i,j_2},$$

hence

$$A_{i,j_1} = x_{j_2}^{k-k'}x_{j_1}^{k'-k}A_{i,j_2}. \tag{3}$$

Since $x_{j_1} \in H_{0,j_1} \cap S_{0,0}$, $x_{j_2} \in H_{0,j_2} \cap S_{0,0}$ and $|F|$ divides $k'-k$, using lemma 1, we obtain $x_{j_1}^{k'-k} = \psi^{(k'-k)r_{D,S}j_1\gamma_{H,S}}$, $x_{j_2}^{k-k'} = \psi^{(k-k')r_{D,S}j_2\gamma_{H,S}}$. Equations (3) and $\psi = \theta^{r_D}$ imply (2). Since $j_1, j_2$ have been arbitrarily, it follows that the proof of the sufficiency is finished. It remains to prove the necessity.

Let $B_{i,j} = \theta^{(k-k')(i-r_S j\gamma_{H,S})}A_{i,0}$, $i = \{0,\dots,r_D-1\}$, $j = \{0,\dots,r_{D,S}-1\}$. Let us show that the equality $f(x) = x^{k'}B_{i,j}$ holds for all $x \in S_{i,j}$ .

For some $j' \in \{0,\dots,r_{D,H}-1\}$ we have $x \in S_{i,j} \cap H_{i,j'}$. Using lemma 1, we obtain $x = \xi^l\theta^{i+r_S(j'-j)\gamma_{H,S}}$ for some $l \in \{0,\dots,|F|-1\}$. Hence we have:

$$x^{k'} = \xi^{k'l}\theta^{k'(i+r_S(j'-j)\gamma_{H,S})} = \xi^{kl}\theta^{k(i+r_S(j'-j)\gamma_{H,S})}\xi^{(k'-k)l}\theta^{(k'-k)(i+r_S(j'-j)\gamma_{H,S})} =$$
$$= x^k\theta^{(k'-k)(i+r_S(j'-j)\gamma_{H,S})}.$$

The equation (2) implies that

$$B_{i,j} = \theta^{(k-k')(i-r_S j\gamma_{H,S})}A_{i,0} = \theta^{(k-k')(i+r_S(j'-j)\gamma_{H,S})}A_{i,j'}.$$

Therefore, we obtain

$$f(x) = xA_{i,j'} = x\theta^{(k'-k)(i+r_S(j'-j)\gamma_{H,S})}B_{i,j} = x^{k'}B_{i,j}.$$

□

Let us describe a group contained in the automorphism group of mappings from $PM_{n,k}(H) \cap PM_{n,k'}(S)$. Since $r_D = GCD(r_H, r_S)$, it follows that there are integer numbers $u, v$, such that $r_D = ur_H + vr_S$.

**Proposition 4.** *Let* $H, S < \mathbb{F}_{2^n}^* = \langle\theta\rangle$, $D = \langle H, S\rangle = \langle\psi\rangle$, $\psi = \theta^{r_D}$. *If* $f \in PM_{n,k}(H) \cap PM_{n,k'}(S)$, *then*

$$\langle (x, y) \mapsto (x\psi, y\psi^{ukr_{D,H}+vk'r_{D,S}}) \,|x, y \in \mathbb{F}_{2^n} \rangle < \mathrm{Aut}(f),$$

*where* $u, v$ *satisfy* $r_D = r_H u + r_S v$.

*Proof.* Since $r_D = GCD(r_H, r_S)$, it follows that $\psi = \theta^{r_H u + r_S v}$. For all $x \in \mathbb{F}_{2^n}$ we have:

$$f(x\psi) = f\left((x\theta^{r_S v})\theta^{r_H u}\right) = f\left(x\theta^{r_S v}\right)\theta^{kr_H u} =$$
$$= f\left(x\right)\theta^{k'r_S v + kr_H u} = f(x)\psi^{uk\frac{r_H}{r_D} + vk'\frac{r_S}{r_D}} = f(x)\psi^{ukr_{D,H}+vk'r_{D,S}}.$$

This concludes the proof. □

Let us describe a necessary and sufficient condition for mapping from $f \in PM_{n,k}(H) \cap PM_{n,k'}(S)$ to be a bijection.

**Proposition 5.** *A mapping* $f \in PM_{n,k}(H) \cap PM_{n,k'}(S)$, $f\colon x \mapsto x^k A_{i,j}$, $x \in H_{i,j}$, $A_{i,j} = \theta^{a_{i,j}}$, $i = \{0, \ldots, r_D - 1\}$, $j = \{0, \ldots, r_{D,H} - 1\}$ *is a bijection if and only if* $GCD(k, |H|) = 1$, $GCD(r_D(k' - k + 1), r_H) = r_D$ *and the set* $\{a_{i,0} + ki \,|i \in \{0, \ldots, r_D - 1\}\}$ *is a complete set of residues modulo* $r_D$.

*Proof.* Let us select an arbitrary number $i \in \{0, \ldots, r_D - 1\}$. Using theorem 1, we obtain that for all $j \in \{0, \ldots, r_{D,H} - 1\}$ the equality

$$A_{i,j} = \theta^{jr_D(k'-k)} A_{i,0} \tag{4}$$

holds. Hence for all $j_1, j_2 \in \{0, \ldots, r_{D,H} - 1\}$ elements $A_{i,j_1}$, $A_{i,j_2}$ belong to the same coset of $D$ in $\mathbb{F}_{2^n}^*$. Since $f(D_i) \subseteq D_{a_{i,0}+ki}$, it follows that a mapping $f$ is bijective if and only if the following conditions are satisfied:

1) for all $i \in \{0, \ldots, r_D - 1\}$ the equality $f(D_i) = D_{a_{i,0}+ki}$ holds,

2) for all $i_1, i_2 \in \{0, \ldots, r_D - 1\}$, $i_1 \neq i_2$, the inequality $f(D_{i_1}) \neq f(D_{i_2})$ holds.

Condition 2 is equivalent to the fact that the set $\{a_{i,0} + ki \,|i \in \{0, \ldots, r_D - 1\}\}$ is a complete set of residues modulo $r_D$.

Since $f$ maps any coset of subgroup $H$ into the coset of $H$, it follows that condition 1 is equivalent to the following conditions:

a) for all $i \in \{0, \ldots, r_D - 1\}$, $j \in \{0, \ldots, r_{D,H} - 1\}$ equality $|f(H_{i,j})| = |H|$ holds (i.e., $f$ is an injective mapping on any cosets of $H$),

b) for all $i \in \{0, \dots, r_D - 1\}$ and $j_1, j_2 \in \{0, \dots, r_{D,H} - 1\}$, $j_1 \neq j_2$, inequality $f(H_{i,j_1}) \neq f(H_{i,j_2})$ holds.

Condition a is equivalent to the equality $GCD(k, |H|) = 1$.

Let us show that condition b is satisfied if and only if

$$GCD(r_D(k' - k + 1), r_H) = r_D.$$

Suppose that $f(H_{i,j_1}) = f(H_{i,j_2})$ for some $i \in \{0, \dots, r_D - 1\}$ and $j_1, j_2 \in \{0, \dots, r_{D,H} - 1\}$, $j_1 \neq j_2$. Using equation (4), we obtain that the equality $f(H_{i,j_1}) = f(H_{i,j_2})$ is equivalent to the following congruence:

$$j_1 r_D(k' - k) + a_{i,0} + i + j_1 r_D \equiv j_2 r_D(k' - k) + a_{i,0} + i + j_2 r_D \mod r_H.$$

This congruence equivalent to the following congruence:

$$r_D(k' - k + 1)(j_1 - j_2) \equiv 0 \mod r_H. \tag{5}$$

Congruence (5) has a unique solution $j_1 = j_2$ if and only if $GCD(r_D(k' - k + 1), r_H) = r_D$. Indeed, suppose $\mathrm{GCD}(r_D(k' - k + 1), r_H) = r_D$. Since $j_1, j_2 \in \{0, \dots, r_{D,H} - 1\}$ and $\frac{r_H}{r_D} = r_{D,H}$, it follows that the congruence (5) is equivalent to the congruence $j_1 - j_2 \equiv 0 \mod r_{D,H}$. Therefore, we have $j_1 = j_2$. If $GCD(r_D(k' - k + 1), r_H) = d = r_D t$, $t > 1$, then $(j_1, j_2) = \left(\frac{r_{D,H}}{d}, 0\right)$ is a solution of the congruence (5). □

Let us compute the cardinality of classes $PM_{n,k}(H) \cap PM_{n,k'}(S)$, $PM_{n,k}(H) \cap PM_{n,k'}(S) \cap S(\mathbb{F}_{2^n})$, $k, k' \in \{0, \dots, n - 1\}$.

**Proposition 6.** *Let $H, S, D = \langle H, S \rangle < \mathbb{F}_{2^n}^*$, then*

$$|PM_{n,k}(H) \cap PM_{n,k'}(S)| = \begin{cases} 2^{nr_D} & \text{if } |H \cap S| \mid k' - k, \\ 1 & \text{otherwise.} \end{cases}$$

$$|PM_{n,k}(H) \cap PM_{n,k'}(S) \cap S(\mathbb{F}_{2^n})| =$$

$$= \begin{cases} r_D! |D|^{r_D} & \text{if } |H \cap S| \mid k' - k, GCD(r_D(k' - k + 1), r_H) = r_D, \\ & GCD(k, |H|) = 1, \\ 0 & \text{otherwise.} \end{cases}$$

*Proof.* Let $f \in PM_{n,k}(H) \cap PM_{n,k'}(S)$. If $|H \cap S| \nmid k' - k$, then using proposition 3, we obtain that $f = \mathbb{O}$ and $|PM_{n,k}(H) \cap PM_{n,k'}(S) \cap S(\mathbb{F}_{2^n})| = 1$. If $|H \cap S| \mid k' - k$, then theorem 1 implies that the mapping $f$ is uniquely

defined by the set of elements $A_{0,0}, \ldots, A_{r_D-1,0} \in \mathbb{F}_{2^n}$. It follows that $|PM_{n,k}(H) \cap PM_{n,k'}(S) \cap S(\mathbb{F}_{2^n})| = 2^{nr_D}$.

If conditions $|H \cap S| \mid k' - k$, $GCD(r_D(k' - k + 1), r_H) = r_D$, $GCD(k, |H|) = 1$ are not satisfied, then proposition 5 implies that there are no permutations in the set $PM_{n,k}(H) \cap PM_{n,k'}(S)$. Using proposition 5, we obtain that the mapping $f$ is a permutation if and only if $A_{0,0} = \theta^{a_{0,0}}, \ldots, A_{r_D-1,0} = \theta^{a_{r_D-1,0}} \in \mathbb{F}_{2^n}^*$ and the set $\{a_{i,0} + ki \,|i = 0, \ldots, r_D - 1\}$ is a complete set of residues modulo $r_D$. The number of such collections is $r_D!|D|^{r_D}$. □

## 5 Class $k$-piecewise-monomial mappings with automorphism group containing automorphisms of field

By $\sigma_m \colon \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ we define the automorphism $x \mapsto x^{2^m}$ of a field $\mathbb{F}_{2^n}$.

**Definition 8.** *Define the set $RS_{n,m}$, $m \mid n$ as a set of all mappings of a field $\mathbb{F}_{2^n}$ satisfying the following equality*

$$f(\sigma_m(x)) = \sigma_m\left(f(x)\right), x \in \mathbb{F}_{2^n}. \tag{6}$$

Let $f \in PM_{n,k}(H)$, $f \colon x \mapsto x^k A_i$, $A_i \in \mathbb{F}_{2^n}$, $i = 0, \ldots, r_H - 1$. Let us describe necessary and sufficient conditions for $A_0, \ldots, A_{r_H-1}$ which imply that $f$ belongs to $RS_{n,m}$.

**Proposition 7.** *Let $\mathbb{F}_{2^n}^* = \bigcup\limits_{i=0}^{r_H-1} H_i$ — a partition of $\mathbb{F}_{2^n}^*$ into the cosets of subgroup $H < \mathbb{F}_{2^n}^* = \langle\theta\rangle$, $H_i = H\theta^i$, $i = 0, \ldots, r_H - 1$, $f \in PM_{n,k}(H)$, $f \colon x \mapsto x^k A_i$, $x \in H_i$, $A_i \in \mathbb{F}_{2^n}$, $i = 0, \ldots, r_H - 1$. Then $f \in RS_{n,m}$ if and only if for all $i \in \{0, \ldots, r_H - 1\}$ the equality $A_i^{2^m} = A_{i2^m}$ holds.*

*Proof.* Suppose $f \in PM_{n,k}(H)$. Mapping $f$ belongs to $RS_{n,m}$ if and only if the equalities $f\left(\sigma_m(x)\right) = x^{k2^m} A_{i2^m}$, $\sigma_m\left(f(x)\right) = x^{k2^m} A_i^{2^m}$ hold for all $x \in H_i$, $i \in \{0, \ldots, r_H - 1\}$. Hence (6) is equivalent to the equalities $A_i^{2^m} = A_{i2^m}$, $i = \{0, \ldots, r_H - 1\}$. □

## 6 Experimental results

For constructing differentially 4-uniform piecewise-monomial permutations we applied adapted spectral differential method [14]. The main idea of this method is a sequential definition of the action $f \in PM_{n,k}\left(H\right)$ on cosets $H_i$, $i = 0, ..., r_H - 1$, by special choice of elements $a_i \in \{0, ..., 2^n - 2\}$. An

element $a_i$ on step $i = 0, ..., r_H - 1$ is defined by conditions of the mapping $f \in PM_{n,k}(H)$ bijectivity and the differential $\delta$-uniformity of the mapping $f_{H_{(i)}} \colon H_{(i)} \to \mathbb{F}_{2^n}$, representing a restriction of permutation $f \in PM_{n,k}(H)$ to the set $H_{(i)} = \bigcup_{j=0}^{i} H_j \cup \{0\}$, $i \in \{0, ..., r-1\}$.

Table 1 in appendix contains some examples of differentially 4-uniform permutations $f \in PM_{8,k}(H)$, $|H| = 17$, $k \in \{3, 5, 7, 11\}$ with the graph algebraic immunity 3. Also this table contains an example of piecewise-monomial permutations $f$ with the nonlinearity 108.

Table 2 in appendix contains some examples of differentially 4-uniform permutations $f \in PM_{8,7}(H)$, $|H| = 15$, with $AI(f) = 3$.

Table 3 in appendix contains some examples of differentially 4-uniform permutation $f \in PM_{8,7}(H)$, $|H| = 51$, with $AI(f) = 3$.

Table 4 in appendix contains an example of differentially 4-uniform permutations $f \in PM_{8,3}(H)$, $|H| = 85$, with $AI(f) = 3$.

Table 5 in appendix contains some examples of differentially 4-uniform permutations $f \in PM_{8,1}(H) \cap RS_{8,2}$, $|H| = 5$, with $AI(f) = 3$.

Tables 6, 7, 8 in appendix contain up to affine-equivalence all classes representatives of differentially 4-uniform permutations from sets $PM_{8,2}(H) \cap PM_{8,1}(S)$, $|H| = 17$, $|S| = 5$; $PM_{8,2}(H) \cap PM_{8,1}(S)$, $|H| = 17$, $|S| = 3$; $PM_{8,1}(H) \cap RS_{8,2}$, $|H| = 17$. All permutations from 6, 7, 8 have the graph algebraic immunity equal to 2.

Random search in $PM_{8,k}(H)$ when $|H| \in \{15, 17\}$, $PM_{8,1}(H) \cap RS_{8,2}$, $|H| = 5$ (sample size $10^8$) gives no differentially 4-uniform permutations. In the other sets some differentially 4-uniform permutations can be found using random search in the sample size of $10^8$ due to a bigger automorphism group.

# References


[1] Beierle C., Brinkmann M., Leander G., "Linearly self-equivalent APN permutations in small dimension", *IEEE Transactions on Information Theory*, **67**:7 (2021), 4863–4875.

[2] Canteaut A., Duval S., Leurent G., "Construction of Lightweight S-Boxes Using Feistel and MISTY Structures", SAC 2015, Lect. Notes Comput. Sci., **8731**, 2016, 373–393.

[3] Carlet C., *Boolean functions for cryptography and coding theory*, Cambridge University Press, 2020, 574 pp.

[4] Courtois N.T., Debraize B., Garrido E., "On Exact Algebraic [Non-]Immunity of S-Boxes Based on Power Functions", Information Security and Privacy. ACISP 2006, Lect. Notes Comput. Sci., **4058**, 2006, 76–86

[5] Courtois N.T., Pieprzyk J., "Cryptanalysis of Block Ciphers with Overdefined Systems of Equations", Asiacrypt 2002, Lect. Notes Comput. Sci., **2501**, 2002, 267–287

[6] de la Cruz Jimenez R.A., "Generation of 8-Bit S-Boxes Having Almost Optimal Cryptographic Properties Using Smaller 4-Bit S-Boxes and Finite Field Multiplication", Latincrypt 2017, Lect. Notes Comput. Sci., **11368**, 2019, 191–206

[7] Davydov S.A., Kruglov I.A., "A method of construction of differentially 4-uniform permutations over $V_m$ for even $m$", *Discrete Math. Appl.*, **31**:6 (2021), 383–388

[8] Ivanov G., Nikolov N., Nikova S., "Cryptographically Strong S-Boxes Generated by Modified Immune Algorithm", BalkanCryptSec 2015, Lect. Notes Comput. Sci., **9540**, 2016, 31–42

[9] Kavut S., "Results on rotation-symmetric S-boxes", *Information Sciences*, **201** (2012), 93–113

[10] Kavut S., Baloglu S., "Results on symmetric S-boxes constructed by concatenation of RSSBs", *Cryptography and Communications*, **11** (2019), 641–660

[11] Kovrizhnykh M. A., Fomin D. B., "Heuristic algorithm for obtaining permutations with given cryptographic properties using a generalized construction", *Prikl. Diskr. Mat.*, **57** (2022), 5–21

[12] Li Y., Wang M., "Constructing S-boxes for Lightweight Cryptography with Feistel Structure", CHES 2014, Lect. Notes Comput. Sci., **8731**, 2014, 127–146

[13] Menyachikhin A. V., "Spectral-linear and spectral-differential methods for generating S-boxes having almost optimal cryptographic parameters", *Матем. вопр. криптогр.*, **8**:2 (2017), 97–116

[14] Menyachikhin A. V., "Adapted spectral-differential method for generating differentially 4-uniform piecewise-linear permutations, orthomorphisms, involutions ща the field $\mathbb{F}_{2^n}$", *Discrete Math. Appl. (under review)*

[15] Nyberg K., "Differentially uniform mappings for cryptography", Eurocrypt 1993, Lect. Notes Comput. Sci., **765**, 1994, 55–64

[16] Prabowo T.F., Tan C.H., "Implicit quadratic property of differentially 4-uniform permutations", Indocrypt 2016, Lect. Notes Comput. Sci., **10095**, 2016, 364–379

[17] Trishin A.E., "The nonlinearity index for a piecewise-linear substitution of the additive group of the field $\mathbb{F}_{2^n}$", *Prikl. Diskr. Mat.*, **4**:30 (2015), 32–42

# Appendix

## Examples of differential 4-uniform piecewise monomial permutations with graph algebraic immunity equals to 3

Table 1.
Substitutions $f \in PM_{8,k}(H)$, $|H| = 17$
$\mathbb{F}_{2^8} = \mathbb{F}_2[x]/x^8 + x^4 + x^3 + x + 1$, $\theta = 3$ and
vector $\vec{a}_g \in \{0, ..., 2^8 - 2\}^{15}$ is written in hex-notation

| $k$ | $\vec{a}_g = (a_0, a_1, ..., a_{14})$ | | | | | | | | | | | | | | | $nl(f)$ | $\overline{\deg}(f)$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 7d | d3 | 92 | f1 | f6 | 3d | 12 | 8f | 86 | 0 | 36 | cf | d7 | c1 | 88 | 104 | 6 |
| | 6e | d0 | 7 | ba | c0 | ac | 77 | a7 | 1b | fd | e4 | 10 | f8 | b7 | c2 | 100 | 6 |
| | 97 | a7 | f2 | 3a | 1b | 6 | 25 | 80 | f4 | f6 | a9 | 8f | 1a | 8a | b7 | 104 | 6 |
| | 99 | 1 | 2 | f5 | da | f6 | 9 | 28 | 17 | 10 | 96 | 62 | e2 | 4b | 76 | 108 | 6 |
| | c1 | 5b | 7d | 4b | 9c | 51 | df | 1d | 1 | a8 | 71 | e | da | 3 | dc | 104 | 6 |
| | 22 | d4 | 9d | 6f | a1 | a | 54 | f8 | 35 | 95 | 45 | 4b | 1e | f7 | be | 104 | 6 |
| | 9b | d3 | b4 | 52 | b5 | 1e | 50 | 57 | 50 | 1e | 45 | 25 | 92 | fd | 95 | 104 | 6 |
| 5 | cb | eb | e | 66 | ba | 80 | 16 | 3b | 83 | 3e | d7 | 4e | e2 | 54 | 28 | 100 | 6 |
| | 5 | f7 | f3 | 5c | 8a | ab | d9 | b8 | e1 | 3f | 90 | f5 | bf | c2 | 92 | 104 | 6 |
| | 48 | 28 | 53 | e6 | 54 | 1 | 5b | f3 | 63 | ae | cf | 1e | fd | a6 | 48 | 104 | 6 |
| | e1 | 53 | 22 | f3 | b | 94 | a9 | 34 | be | 89 | 51 | 3b | 70 | 79 | 4b | 104 | 6 |
| 7 | 2 | a6 | 3e | ba | 60 | 2d | 6c | 36 | 7b | 31 | 1e | f4 | 27 | 46 | f1 | 104 | 6 |
| | d3 | d0 | b0 | c | 46 | 48 | de | af | 2 | da | ea | 2b | 58 | 56 | 67 | 104 | 6 |
| | 66 | 91 | e5 | 79 | c3 | c2 | 40 | 64 | d1 | ab | fb | 3f | 7a | 1d | 5a | 104 | 6 |
| | df | 76 | 82 | 8d | 64 | 8c | bb | 7b | 8c | c3 | 7d | de | 53 | dc | 58 | 104 | 6 |
| | 66 | af | eb | b1 | 0 | af | 91 | 60 | f3 | 8a | a0 | ba | af | 4b | a8 | 100 | 6 |
| | 2 | 67 | 8e | fa | 46 | 8f | 98 | 8a | 9d | 6a | c2 | df | 4c | 82 | e5 | 104 | 6 |
| | 7c | 31 | 8e | 7a | 40 | 18 | 22 | 9f | 56 | 60 | 3e | 94 | 29 | 63 | af | 100 | 6 |
| 11 | 5 | 55 | 72 | 8b | 4 | 49 | 43 | e8 | e5 | b9 | 25 | c2 | 59 | 2 | f | 104 | 6 |
| | 45 | 17 | 3c | e6 | 88 | 2 | 28 | b | 4 | 92 | df | dc | 9 | 3e | 64 | 100 | 6 |
| | f3 | c6 | f8 | 6e | bb | 55 | 3a | 82 | c3 | 2d | 50 | f | 5 | fe | d9 | 104 | 6 |
| | 3f | 6 | 8b | fb | 2f | b1 | 31 | 98 | 56 | 7c | 4e | b8 | c3 | 16 | 4d | 104 | 6 |
| | b7 | 90 | de | ef | df | 5d | 3 | d | 35 | 5c | fb | 7e | 3e | 20 | 7c | 100 | 6 |
| | 95 | 66 | a4 | db | 9b | b9 | e5 | e | 5 | 71 | 78 | 6 | a6 | 8a | 14 | 104 | 6 |
| | a | b1 | 73 | 7c | 4b | 7a | 2a | 83 | 94 | f0 | 2c | 23 | e4 | e9 | d3 | 104 | 6 |

Table 2.
Permutations $f \in PM_{8,k}(H)$, $|H| = 15$,
$\mathbb{F}_{2^8} = \mathbb{F}_2[x]/x^8 + x^4 + x^3 + x + 1$, $\theta = 3$ and
vector $\vec{a}_g \in \{0, ..., 2^8 - 2\}^{17}$ is written in hex-notation

| $k$ | $\vec{a}_g = (a_0, a_1, ..., a_{16})$ | | | | | | | | | | | | | | | | $nl(f)$ | $\overline{\deg}(f)$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 7 | c7 | d9 | 46 | 88 | ac | 3a | ef | 7f | 55 | c | 40 | a | fd | b6 | 1f | fc | 89 | 102 | 7 |
| | 13 | b | 78 | 99 | 64 | d2 | 46 | 98 | e9 | f9 | 4f | bd | c8 | 35 | 91 | 88 | 39 | 104 | 7 |

Table 3.
Permutations $f \in PM_{8,k}(H)$, $|H| = 51$,
$\mathbb{F}_{2^8} = \mathbb{F}_2[x]/x^8 + x^4 + x^3 + x + 1$, $\theta = 3$ and
vector $\vec{a}_g \in \{0, ..., 2^8 - 2\}^{5}$ is written in hex-notation

| $k$ | $\vec{a}_g = (a_0, a_1, ..., a_4)$ | | | | | $nl(f)$ | $\overline{\deg}(f)$ |
|---|---|---|---|---|---|---|---|
| 7 | 3b | f1 | 7f | fa | 26 | 104 | 5 |
| | 3b | ee | 57 | fb | 50 | 104 | 5 |
| | 3b | a2 | 2e | 32 | b7 | 104 | 5 |
| | 3b | d5 | 46 | b | 84 | 100 | 5 |
| | 3b | e9 | d7 | e7 | f2 | 104 | 5 |
| | 3b | 2f | 9c | 12 | f0 | 104 | 5 |
| | 3b | 76 | 8d | c5 | 4b | 104 | 5 |

Table 4.
Permutation $f \in PM_{8,k}(H)$, $|H| = 85$,
$\mathbb{F}_{2^8} = \mathbb{F}_2[x]/x^8 + x^4 + x^3 + x + 1$, $\theta = 3$ and
vector $\vec{a}_g \in \{0, ..., 2^8 - 2\}^{3}$ is written in hex-notation

| $k$ | $a_0, a_1, a_2$ | | | $nl(f)$ | $\overline{\deg}(f)$ |
|---|---|---|---|---|---|
| 3 | f7 | b0 | 4b | 104 | 5 |

Table 5.
Permutations $f \in PM_{8,1}(H) \cap RS_{8,2}$, $|H| = 5$,
$\mathbb{F}_{2^8} = \mathbb{F}_2[x]/x^8 + x^4 + x^3 + x + 1$, $\theta = 3$ and
vector $\vec{a}_g \in \{0, ..., 2^8 - 2\}^{51}$ is written in hex-notation

| $\vec{a}_g = (a_0, a_1, ..., a_{16})$ | | | | | | | | | | | | | | | | | $nl(f)$ | $\overline{\deg}(f)$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ff | d6 | 77 | 3 | 5b | f7 | 68 | 3 | dd | 83 | 30 | 9e | c | b5 | fd | e0 | 6d | | |
| aa | 35 | 8f | df | d4 | 6f | e9 | a1 | 3e | dd | 1a | c | 7f | 4d | db | 77 | 53 | 104 | 6 |
| 55 | 22 | e | bd | 88 | c0 | c0 | a7 | 38 | e3 | 7a | 86 | f6 | 88 | 30 | f8 | 22 | | |
| ff | ee | 51 | d9 | bb | 5e | ea | 7b | 45 | f | b7 | 84 | 67 | bb | 97 | c3 | ee | | |
| aa | 46 | 2 | 79 | 19 | 31 | 48 | ab | 8 | 54 | ba | ed | e5 | 91 | 4c | 15 | 64 | 100 | 7 |
| 55 | 74 | 3c | c4 | d1 | 76 | de | 21 | f0 | 80 | 12 | ae | 13 | 1d | 9d | 20 | 47 | | |
| ff | c1 | d6 | 44 | 7 | 6a | b3 | c4 | 5b | e4 | 4c | e0 | 11 | 70 | 9a | 39 | 1c | | |
| aa | be | d5 | a9 | fa | 6b | e | ce | 57 | b5 | ec | 13 | a6 | af | da | 6d | eb | 104 | 5 |
| 55 | 56 | 93 | ad | 59 | 11 | 31 | 38 | 4e | 75 | 83 | 3b | b6 | 95 | 44 | 5d | 65 | | |
| ff | b5 | 1d | 73 | d6 | 17 | 24 | 72 | 74 | 51 | 27 | 88 | cd | 6d | c5 | 54 | 5b | | |
| aa | 8f | e | 5c | 3e | d | 88 | 90 | 38 | 47 | 9 | c9 | 71 | e3 | 43 | d1 | f8 | 104 | 7 |
| 55 | a2 | 45 | 34 | 8a | dc | 9c | 22 | 15 | 83 | 22 | 42 | d0 | a8 | 37 | e0 | 2a | | |
| ff | 86 | f8 | 7f | 1a | 46 | 83 | c2 | e3 | ce | 2c | e1 | fd | a1 | 91 | b3 | 68 | | |
| aa | 2a | c6 | 19 | a8 | 3 | 1e | e | 1b | 3e | e0 | b | 64 | 8a | c0 | 8f | a2 | 102 | 7 |
| 55 | 2d | 3b | c | b4 | df | b0 | 78 | ec | b1 | 87 | 38 | 30 | 4b | f7 | 6c | d2 | | |
| ff | 62 | 16 | 42 | 89 | 18 | a3 | ce | 58 | 5c | ec | 5d | 9 | 98 | 6 | 17 | 26 | | |
| aa | 13 | d2 | 60 | 4c | 12 | d5 | 8e | 4b | 85 | e8 | 3b | 81 | c4 | 84 | 61 | 31 | 104 | 7 |
| 55 | 39 | 71 | 48 | e4 | 90 | b3 | 57 | c5 | b4 | 75 | 3a | 21 | 4e | 24 | 2d | 93 | | |
| ff | d5 | 63 | 98 | 57 | af | 85 | 12 | 8d | 2d | 21 | 99 | 62 | 75 | eb | 4b | 5d | | |
| aa | 5d | 8a | be | 75 | 97 | 99 | 16 | 2a | d8 | 61 | 48 | fa | 57 | e5 | 36 | d5 | 104 | 6 |
| 55 | 31 | b4 | 5e | c4 | 26 | 84 | 66 | d2 | a2 | 66 | 58 | 79 | 4c | 89 | a8 | 13 | | |
| ff | 26 | 6a | 5e | 98 | 1b | a8 | b8 | a9 | 94 | 8b | 32 | 79 | 89 | c6 | 25 | 62 | | |
| aa | ce | f5 | 6c | 3b | 15 | 23 | a2 | d7 | 9a | 2a | e2 | b1 | b3 | 45 | a6 | ec | 102 | 6 |
| 55 | 81 | 52 | 54 | 6 | 97 | 2e | 8c | 49 | 7d | c8 | 8a | 51 | 60 | e5 | 5f | 18 | | |
| ff | db | fb | f6 | 6f | d3 | 9c | 87 | ef | 1d | 78 | 78 | db | f6 | f4 | 47 | bd | | |
| aa | df | 6a | 4f | 7f | cb | 87 | 72 | a9 | fe | 27 | 1e | 3d | f7 | f2 | bf | fd | 102 | 7 |
| 55 | c3 | 74 | 2f | f | bd | e1 | 1e | d1 | 9a | e1 | c9 | bc | f0 | 6f | a6 | 3c | | |

## Examples of differential 4-uniform piecewise monomial permutations with graph algebraic immunity equals to 2

Table 6.
Permutations $f \in PM_{8,2}(H) \cap PM_{8,1}(S)$, $|H| = 17$, $|S| = 5$,
$\mathbb{F}_{2^8} = \mathbb{F}_2[x]/x^8 + x^4 + x^3 + x^2 + 1$, $\theta = 2$ and
vector $\vec{a}_g \in \{0, ..., 2^8 - 2\}^{15}$ is written in hex-notation

| $k$ | $\vec{a}_g = (a_0, a_1, ..., a_{14})$ | $nl(f)$ | $\overline{\deg}(f)$ |
|---|---|---|---|
| 2 | 0 0 fc 39 39 36 72 72 6f ab ab a8 e4 e4 e1 | 106 | 7 |
| | 0 bd 7b 39 f6 b4 72 30 ed ab 69 27 e4 a2 60 | 112 | 7 |

Table 7.
Permutations $f \in PM_{8,2}(H) \cap PM_{8,1}(S)$, $|H| = 17$, $|S| = 3$,
$\mathbb{F}_{2^8} = \mathbb{F}_2[x]/x^8 + x^4 + x^3 + x^2 + 1$, $\theta = 2$ and
vector $\vec{a}_g \in \{0, ..., 2^8 - 2\}^{15}$ is written in hex-notation

| $k$ | $\vec{a}_g = (a_0, a_1, ..., a_{14})$ | $nl(f)$ | $\overline{\deg}(f)$ |
|---|---|---|---|
| 2 | 0 fe e4 89 d8 55 54 3a de 2e aa a9 8f 34 83 | 102 | 7 |
| | 0 e da ad 6e 55 63 30 3 c3 aa b8 85 58 19 | 106 | 7 |
| | 0 22 44 11 dd 55 77 99 66 33 aa cc ee bb 88 | 104 | 5 |
| | 0 1 11 62 8b 55 56 66 b7 e0 aa ab bb d 36 | 104 | 7 |
| | 0 1 34 ad b3 55 56 89 3 9 aa ab de 58 5e | 100 | 7 |
| | 0 1 84 e4 63 55 56 d9 3a b8 aa ab 2f 8f e | 104 | 7 |
| | 0 6 39 44 7c 55 5b 8e 99 d1 aa b0 e3 ee 27 | 98 | 7 |
| | 0 dd bb 99 77 55 33 11 ee cc aa 88 66 44 22 | 112 | 7 |
| | 0 0 e7 c9 c9 b1 93 93 7b 5d 5d 45 27 27 f | 106 | 7 |

Table 8.
Permutations $f \in PM_{8,1}(H) \cap RS_{8,2}$, $|H| = 17$,
$\mathbb{F}_{2^8} = \mathbb{F}_2[x]/x^8 + x^4 + x^3 + x^2 + 1$, $\theta = 2$ and
vector $\vec{a}_g \in \{0, ..., 2^8 - 2\}^{15}$ is written in hex-notation

| $k$ | $\vec{a}_g = (a_0, a_1, ..., a_{14})$ | $nl(f)$ | $\overline{\deg}(f)$ |
|---|---|---|---|
| 2 | 0 ee dd cc bb aa 99 88 77 66 55 44 33 22 11 | 112 | 7 |
| | 0 11 22 dd 44 aa bb cc 88 ee 55 66 77 33 99 | 108 | 7 |
| | 0 dd bb 44 77 aa 88 66 ee 22 55 33 11 99 cc | 104 | 5 |
| | 0 44 88 dd 11 aa bb 66 22 ee 55 33 77 99 cc | 104 | 6 |

# On the Bit-Slice representations of some nonlinear bijective transformations

Oliver Coy Puente, Rene Fernández Leal and
Reynier Antonio de la Cruz Jiménez

Institute of Cryptography, University of Havana, Cuba
{coypuente.o, renefdzl, djr.antonio537}@gmail.com

**Abstract**

In this paper we study how to obtain efficient Bit-Slice representations in some classes of nonlinear bijective transformations having almost optimal cryptographic properties. For some 8-bit instances belonging to these classes, we determine (by combining analytical methods with a open source tool) its low gate count logic circuit representations trough binary logic operations `AND`, `XOR`, `OR` and `NOT`. In particular, for the S-Box used in the Russian cryptographic standard GOST R 34.12-2015 "Kuznyechik", we derive a Bit-Slice implementation which consume a total of 179 binary logical operations. The new representation requires 48 Boolean operations less than in previously known one (that need 227 bitwise logical operations).

**Keywords:** Bit-Slice, block cipher, nonlinear bijective transformation, S-Box, Kuznyechik

## 1 Introduction

In one of its broader definitions, Cryptography is the field of theoretical and applied research and practical activities related to the development and application of cryptographic information protection methods. As a field of theoretical and applied research, Cryptography is subdivided into cryptographic synthesis and cryptographic analysis, and as a field of practical activity it solves the issues of development and application of a cryptographic module that implement cryptographic systems [20].

Any cryptographic algorithm included in those cryptosystems admits three forms of representations: *mathematical*, *software* and *hardware*. Very often, we say that a cryptographic algorithm is secure if it is mathematically secure. However, if we want to put a cryptographic algorithm to practical use, it has to be implemented in software or hardware form. A cryptographic algorithm that is secure mathematically is not necessarily secure in terms of software or hardware. For example, side-channel attack (SCA) uses the

physical properties associated to the implementation of the cryptographic algorithm to obtain exposed secret parameters in cryptographic operations. The computational effort then needed in theoretical analysis can be reduced greatly.

In wild of the symmetric Cryptography, the (S)ubstitution-Boxes used in the design of many of its primitives[a] must satisfy various security requirements (which evolve over time). Each such requirement corresponds to an specific goal of resisting certain cryptographic attacks, which use (most of them) the linear, differential and algebraic properties of S-Boxes. Efficiency of the implementation (i.e. reduce the number of instructions and allow the operations to be performed in small fields) and Higher-Order Masking against SCA attacks are as important as the security for the S-Box (see, for example,[8, pages 142 and 425]). These aspects of criteria form a trade-off and are often impossible to both be satisfied at a good level.

The basic concept of Bitslicing[b] is to simulate a hardware implementation in software. The entire algorithm is represented as a sequence of logical operations. In software, Bitslicing may be considered as an implementation strategy enabling fast, constant-time implementations of cryptographic algorithms.

In the Bit-Slice implementation context, nonlinear bijective transformations are computed by using binary logical operations (represented in Figure 1) rather than Look-Up Tables (LUT, in brief). It should be noted that the Bit-Slice representation of certain S-Box, allows (automatically) to insert the masking countermeasure against SCA attacks through the so-called ISW scheme given in [17].

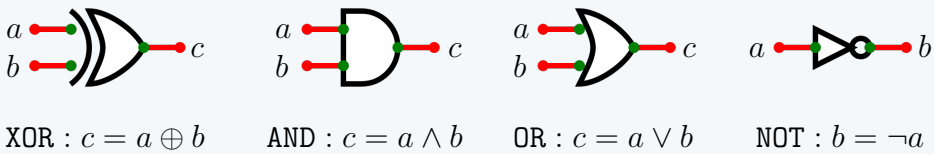



Figure 1: Logical Gates of operations XOR, AND, OR, NOT.

In this paper we study, through analytical work complemented with a computer-aided software provided in [10], how to obtain the Bit-Slice implementations for some specific representatives from certain classes of $2k$-bit S-Boxes proposed in [11, 12, 13]. In what follow, these classes are

[a] To provide nonlinear relationship between the input bits and the output bits.

[b] Introduced by Biham in 1997 as a technique for implementing cryptographic algorithms to improve the software performance of DES [4].

denoted by $\pi'_{h_1,h_2,\mathcal{P}_d}$ and $\hat{\pi}_{\psi,h}$ respectively where $h, h_1, h_2$ are arbitrary permutations of $k$-bit, $\psi$ is a non-bijective function of $k$-bit without preimage for the null element and by $\mathcal{P}_d$ is denoted the permutation polynomial $X^d$ defined over a finite field of $2^k$ elements, where $gcd(d, 2^k - 1) = 1$. Trough the paper, an 8-bit nonlinear bijective transformation without fixed points is called *almost optimal permutation* if it has:

- (algebraic) minimum degree equal to 7;
- (graph) algebraic immunity 3 and 441 equations;
- differential uniformity under 8;
- nonlinearity over 100.

Because of the similarities shared by the design of these classes with the TU-decomposition (given in [5]) of the Kuznyechik S-Box, we are interested (by using the proposed method) in getting the Bit-Slice representation of this (almost optimal) permutation and then compare it with other result, given in [1].

Our work is structured as follows: In Section 2 we give the basic notations and definitions. In Section 3 we construct some 8-bit instances derived from the permutations classes $\pi'_{h_1,h_2,\mathcal{P}_d}$ and $\hat{\pi}_{\psi,h}$ respectively and we re-visit the compact representation of the S-Box used in the Russian cryptographic standard GOST R 34.12-2015 "Kuznyechik". Combining analytical methods with a open source tool we get the Bit-Slice representations for the 8-bit instances having almost optimal properties in Section 4. In this section, we also correct the previously known Bit-Slice representation of S-Box used in "Kuznyechik" block cipher (that requires 227 binary logical operations) given in [1] and by using our approach we found a (new) Bit-Slice representation of this S-Box which consume a total of 179 Boolean operations. In Section 5 we compare our results with previous designs of 8-bit permutations, reported in the state of the art. The paper is concluded in Section 6.

## 2 Preliminaries

Let $\mathbb{F}_2$ be a finite field of two elements. For any $n \in \mathbb{N}$ we denote by $\mathbb{F}_2^n = \mathbb{F}_2 \times \cdots \times \mathbb{F}_2$, the vector space of dimension $n$ with the components from the field $\mathbb{F}_2$. The transpose of a vector $u \in \mathbb{F}_2^n$ is denoted $u^\top$ and by $\oplus$ we denote the addition operation of $\mathbb{F}_2^n$. For $n > 1$, the finite field of $2^n$ elements (denoted by $\mathbb{F}_{2^n}$) is defined as $\mathbb{F}_{2^n} = \mathbb{F}_2[\xi]/g(\xi)$, where $g(\xi)$ is an irreducible

polynomial of degree $n$ over $\mathbb{F}_2$. By $S(\mathbb{F}_2^n)$ we denote the *symmetric group* on $\mathbb{F}_2^n$.

For any $n \in \mathbb{N}$, the vectors from $\mathbb{F}_2^n$ can be interpreted as integers, such that the leftmost bits correspond to the most significant bits. Let $u \in \mathbb{F}_2^n$, then the same vector can be written as:

$$u = (u_{n-1}, \ldots, u_0) \in \mathbb{F}_2^n, \quad \Leftrightarrow \quad u = \sum_{i=0}^{n-1} u_i 2^i \in \mathbb{Z}_{2^n}.$$

We define the indicator function as follows

$$\mathsf{Ind}(x, y) = \begin{cases} 1, & \text{if} \quad x = y; \\ 0, & \text{if} \quad x \neq y. \end{cases}$$

Each output bit of the S-Box $\Phi \in S(\mathbb{F}_2^n)$ naturally defines a Boolean function. The corresponding $n$ Boolean functions are called *coordinates* of $\Phi$.

Any $n$-bit S-Box $\Phi \in S(\mathbb{F}_2^n)$ can be specified by the so-called tabular representation as follows:

$$\Phi = \begin{pmatrix} 0 & 1 & \ldots & i & \ldots & 2^n - 1 \\ \Phi(0) & \Phi(1) & \ldots & \Phi(i) & \ldots & \Phi(2^n - 1) \end{pmatrix}.$$

But very often for simplicity, we represent S-Boxes by the vector of its values (the so-called Look-Up Table) using the following notation:

$$\mathsf{LUT}(\Phi) = (\Phi(0), \Phi(1), \ldots, \Phi(2^n - 1)), \text{ where } \Phi(x) \in \mathbb{F}_2^n.$$

A bijective transformation $\Phi \in S(\mathbb{F}_2^n)$ is called *linear* (resp. *affine*) if all its coordinates are linear (resp. affine). If $\Phi$ is affine, then it can be expressed as $\Phi(x) = x \times A \oplus b$ for a unique $n \times n$ matrix $A$ over $\mathbb{F}_2$ and $b = \Phi(0) \in \mathbb{F}_2^n$, where $b = 0$ if and only if $\Phi$ is linear.

For $n \in \mathbb{N}$, the set of all $n \times n$ matrices over $\mathbb{F}_2$ is denoted $(\mathbb{F}_2)_{n,n}$. Any such matrix $M$ defines a linear map from $\mathbb{F}_2^n$ to $\mathbb{F}_2^n$, given by $x \mapsto x \times M$. The set of all bijective linear maps are denoted $\mathsf{GL}_n(\mathbb{F}_2) \subseteq (\mathbb{F}_2)_{n,n}$. The set of all bijective affine maps is denoted $\mathsf{GA}_n(\mathbb{F}_2)$. By $\Lambda \circ \Psi$ we denote the composition of mappings $\Lambda : \mathbb{F}_2^n \to \mathbb{F}_2^n$ and $\Psi : \mathbb{F}_2^n \to \mathbb{F}_2^n$, where $\Psi$ is the first to operate.

In what follows, we shall use the following notions of Bit-Slice and Gate Equivalent Complexities as implementation criteria.

**Definition 1** (Bit-Slice Gate Complexity (`BGC`) [19])**.** *The smallest number of operations in* `XOR`*,* `AND`*,* `OR`*,* `NOT` *required to implement an S-Box.*

**Definition 2** (Gate Equivalent complexity (GEC) [19]). *The smallest number of Gate Equivalents (GEs) required to implement an S-Box, given the cost of atomic operations. (see, for example, Table 1)*

| Techniques | NAND | XNOR | XOR | AND OR | NOT |
|---|---|---|---|---|---|
| UMC 180nm | 1.00 | 3.00 | 3.00 | 1.33 | 0.67 |
| TSMC 65nm | 1.00 | 3.00 | 3.00 | 1.50 | 0.50 |
| Software | - | - | 1.00 | 1.00 | 1.00 |

Table 1: Cost of atomic operations under various techniques

# 3 Some nonlinear bijective transformations

In this section, we describe some 8-bit instances belonging to the classes of nonlinear bijective transformations that are studied in [11, 13] and we revisit the TU-decomposition of the S-Box used in the Russian cryptographic standard GOST R 34.12-2015 "Kuznyechik". For all of these almost optimal permutations, we compile its LUTs with the basic cryptographic parameters (i.e., differential uniformity, nonlinearity, (algebraic) minimum degree and (graph) algebraic immunity).

## 3.1 An instance of the class of nonlinear bijective transformations $\pi'_{h_1,h_2,\mathcal{P}_d}$

Let $\mathbb{F}_{2^4} = \mathbb{F}_2[\xi]/\xi^4 \oplus \xi \oplus 1$ and $\pi'_{h,\mathcal{I}}$ be an instance of the class of nonlinear bijective transformation $\pi'_{h_1,h_2,\mathcal{P}_d}$ by choosing:

- An affine bijective transformation $\mathcal{A} \in \mathsf{GA}_8(\mathbb{F}_2)$ and linear bijective transformation $\mathcal{L} \in \mathsf{GL}_8(\mathbb{F}_2)$;
- The finite field inversion function $\mathcal{I}$ over $\mathbb{F}_{2^4}$ defined by

$$\mathcal{I}(X) = \mathcal{P}_{14}(X) = X^{14}; \tag{1}$$

- A random permutation $h = h_1 = h_2 \in S(\mathbb{F}_{2^4})$.

For the input value $(l\|r) \in \mathbb{F}_2^8$, the corresponding output value $(l''\|r'') \in \mathbb{F}_2^8$ is computed as $(l''\|r'') = (\mathcal{L} \circ \pi_{h,\mathcal{I}} \circ \mathcal{A})\,(l\|r)$, where

1. $\pi_{h,\mathcal{I}}(l\|r) = (l'\|r')$, being

$$l' = \begin{cases} h(l), & \text{if } r = 0; \\ \mathcal{I}(l \otimes r), & \text{if } r \neq 0; \end{cases} \quad r' = \begin{cases} h(r), & \text{if } l' = 0; \\ l' \otimes \mathcal{I}(r), & \text{if } l' \neq 0. \end{cases} \tag{2}$$

2. The affine and linear transformations are given by

$$\mathcal{A}(l_3, l_2, l_1, l_0, r_3, r_2, r_1, r_0) = (r_0, r_2, l_3, r_3, l_2 \oplus 1, r_1, l_0, l_1), \quad (3)$$
$$\mathcal{L}(l_3, l_2, l_1, l_0, r_3, r_2, r_1, r_0) = (r_3, l_2, r_0, r_1, l_3, r_2, l_0, l_1), \quad (4)$$

respectively.

The LUTs of the nonlinear bijective transformations $h$ and $\mathcal{I}$ are given in Table 2.

These components were selected in such a way that the resulting almost optimal permutation has the best possible properties (see, [12]), in particular, the 4-bit permutation $h$ was found by using the algorithm described in [11]. For the mentioned components of $\pi'_{h_1,h_2,\mathcal{P}_d}$ we obtain the S-Box $\pi'_{h,\mathcal{I}}$. The LUT of $\pi'_{h,\mathcal{I}}$ is showed in Table 3, where, for example, for the input value BA the corresponding output value is 0E, i.e., $\pi'_{h,\mathcal{I}}(\texttt{BA}) = \texttt{0E}$.

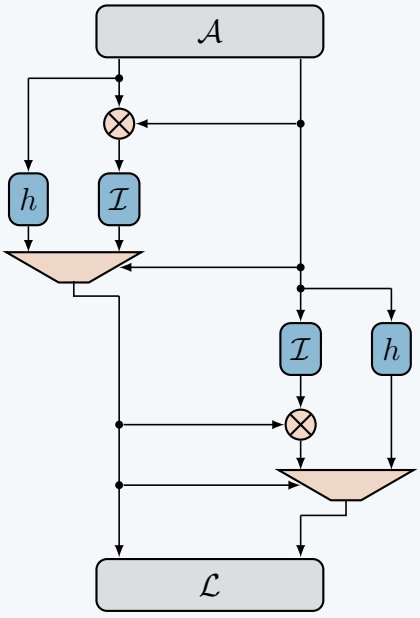



Figure 2: High level of view of $\pi'_{h,\mathcal{I}}$.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $h$ | 0 | 1 | E | 9 | F | 5 | C | 2 | B | A | 4 | 8 | D | 6 | 3 | 7 |
| $\mathcal{I}$ | 0 | 1 | 9 | E | D | B | 7 | 6 | F | 2 | C | 5 | A | 4 | 3 | 8 |

Table 2: The LUTs of the 4-bit nonlinear transformations of $\pi'_{h,\mathcal{I}}$.

| Cryptographic Properties of $\pi'_{h,\mathcal{I}}$ | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Nonlinearity – 108 Algebraic Degree – 7 | | | | | | | | | | | | | | | | |
| Differential Uniformity – 6 Graph Algebraic Immunity – 3(441) | | | | | | | | | | | | | | | | |
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| 0 | B0 | 8D | A4 | 68 | F3 | C8 | 1B | BC | DB | 7E | 89 | 73 | 13 | 9E | 35 | 5D |
| 1 | 04 | 98 | 30 | 15 | 37 | 86 | C4 | 4E | FC | AF | 27 | D1 | 7A | E2 | 69 | 7C |
| 2 | 90 | EA | 14 | 1A | 2A | D7 | 06 | ED | 05 | C0 | 70 | 1C | D2 | 38 | 9D | 87 |
| 3 | 80 | E5 | 34 | 82 | E8 | 74 | 31 | 6B | 52 | 0D | 8C | B3 | 26 | C3 | E7 | 65 |
| 4 | 00 | 0B | B4 | 67 | 4B | 4A | D9 | 1F | 02 | 09 | DE | BE | 42 | 41 | C1 | A6 |
| 5 | 94 | 7D | 84 | 0F | DF | F2 | C2 | A3 | AE | A2 | 57 | CD | 5C | 21 | F4 | FB |
| 6 | 20 | FF | 24 | 97 | EE | 99 | F5 | 25 | 22 | 11 | AB | 62 | BB | 44 | 8E | 19 |
| 7 | A0 | 72 | 10 | F0 | 1D | 51 | 2C | 3A | F9 | 6F | 75 | DC | A8 | DA | 4F | BF |

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 8 | 6D | 28 | 46 | 92 | 3B | A5 | A7 | FA | 56 | 45 | E1 | D4 | E0 | B6 | 2E | CF |
| 9 | D5 | CB | B8 | E3 | B1 | 53 | 8A | F6 | 64 | 1E | 32 | 5B | 4D | 29 | AD | 9F |
| A | 12 | 2F | 81 | 76 | FD | C5 | EB | 6C | EF | 3D | 6A | F7 | F8 | 17 | 9B | F1 |
| B | 2B | BA | 54 | BD | 9C | 5F | 5A | 3F | B7 | 91 | 0E | E9 | CE | 79 | D6 | D8 |
| C | 49 | 40 | 7F | 07 | 48 | 03 | C6 | 60 | 0A | 08 | B9 | 78 | 01 | 43 | 18 | A1 |
| D | FE | 71 | 39 | 95 | 2D | 0C | 61 | 9A | D3 | 8F | 58 | AC | 83 | 50 | 36 | 6E |
| E | AA | CC | EC | 5E | 77 | 33 | D0 | C9 | DD | 66 | 3C | B2 | 55 | 88 | 7B | 47 |
| F | C7 | E4 | 93 | 59 | 4C | 96 | 16 | A9 | 8B | 23 | 85 | CA | B5 | 3E | 63 | E6 |

Table 3: The LUT of $\pi'_{h,\mathcal{I}}$.

### 3.1.1 A variant of $\pi'_{h,\mathcal{I}}$

The following instance was obtained as a result of a oriented search on the structural elements used in one of the possible modification of $\pi'_{h_1,h_2,\mathcal{P}_d}$ that offer the best implementation cost (achieved in this paper) of the resulting almost optimal permutation.

The nonlinear bijective transformation $\dot{\pi}_{\lambda,\tau}$ employ the following components:

- An affine bijective transformation $\mathcal{A} \in \mathsf{GA}_8(\mathbb{F}_2)$ and the linear transformations $\mathcal{L} \in \mathsf{GL}_8(\mathbb{F}_2)$ and $\lambda \in \mathsf{GL}_4(\mathbb{F}_2)$;
- The 4-bit nonlinear bijective transformation $\tau$ defined over $\mathbb{F}_{2^4}$.

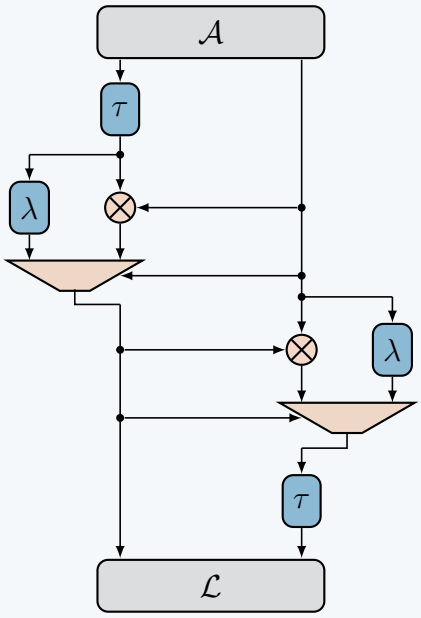



Figure 3: High level of view of $\dot{\pi}_{\lambda,\tau}$.

For the input value $(l\|r) \in \mathbb{F}_2^8$, the corresponding output value $(l''\|r'') \in \mathbb{F}_2^8$ is computed as $(l''\|r'') = \dot{\pi}_{\lambda,\tau} = (\mathcal{L} \circ \pi_{\lambda,\tau} \circ \mathcal{A})\,(l\|r)$, where

1. $\pi_{\lambda,\tau}(l\|r) = (l'\|r')$, being

$$l' = \begin{cases} \lambda(\tau(l)), & \text{if } r = 0, \\ \tau(l) \otimes r, & \text{if } r \neq 0, \end{cases} \qquad r' = \begin{cases} \tau(\lambda(r)), & \text{if } l' = 0, \\ \tau(l' \otimes r) & \text{if } l' \neq 0. \end{cases} \tag{5}$$

2. The affine and linear transformations are defined by

$$\mathcal{A}(l_3, l_2, l_1, l_0, r_3, r_2, r_1, r_0) = (r_1, r_3, r_0, l_0, l_3, r_2, l_1, l_2 \oplus 1), \quad (6)$$
$$\mathcal{L}(l_3, l_2, l_1, l_0, r_3, r_2, r_1, r_0) = (l_0, l_1, l_2, r_2, r_1, r_3, r_0, l_3), \quad (7)$$
$$\lambda(x_3, x_2, x_1, x_0) = (x_2, x_3, x_1, x_0). \quad (8)$$

The LUT of the nonlinear bijective transformations $\tau$ is given in Table 4.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\tau$ | 0 | 1 | D | B | E | 9 | 6 | 7 | A | 4 | F | 2 | 8 | 3 | 5 | C |

Table 4: The LUT of the 4-bit nonlinear transformation $\tau$.

The LUT of the S-Box $\dot{\pi}_{\lambda,\tau}$ is given in Table 5, where, for example, for the input value A7 the corresponding output value is 45, i.e., $\dot{\pi}_{\lambda,\tau}(\mathtt{A7}) = \mathtt{45}$.

| Cryptographic Properties of $\dot{\pi}_{\lambda,\tau}$ | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Nonlinearity – 108 Algebraic Degree – 7<br>Differential Uniformity – 6 Graph Algebraic Immunity – 3(441) | | | | | | | | | | | | | | | | |
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| 0 | 02 | AB | 5F | F5 | 10 | 31 | 3A | 6A | 73 | 78 | 0D | A6 | D4 | A5 | 6F | 5E |
| 1 | 82 | C9 | 3C | 56 | B6 | 86 | EC | 5D | 91 | FA | CE | 25 | C3 | 13 | F9 | 89 |
| 2 | 0E | 24 | BD | 58 | 08 | AC | D0 | DD | 8E | 4B | D3 | F7 | 37 | 9A | A7 | 0B |
| 3 | C6 | 63 | 3B | 7E | F8 | 34 | E9 | 65 | 09 | 8D | B4 | F0 | 4F | 62 | 9F | 52 |
| 4 | 00 | A1 | 60 | E1 | 0C | 9C | 6B | 97 | 61 | 41 | 20 | 81 | A3 | DF | 68 | F4 |
| 5 | 80 | E0 | 01 | 40 | 2E | 53 | C4 | 19 | A0 | C1 | C0 | 21 | 4C | F1 | 27 | BA |
| 6 | 16 | 83 | F2 | B1 | 1E | 07 | 8F | 28 | EB | 29 | D8 | 5B | 44 | E2 | AA | AD |
| 7 | 5C | BE | 0F | 2C | 7A | FD | D5 | 33 | 96 | 75 | 64 | C7 | DE | 98 | 51 | 77 |
| 8 | 06 | F3 | A8 | 7B | 0A | 7F | D9 | E6 | FE | CC | 36 | C5 | 49 | 17 | 50 | 2F |
| 9 | 8B | 39 | 42 | 90 | B3 | 6C | 8A | 95 | B5 | 47 | 1D | 6E | FB | A4 | 22 | DC |
| A | 1A | 74 | 46 | D2 | 14 | E8 | 2D | 45 | 1B | EF | FC | 88 | B8 | 30 | 94 | 7C |
| B | D1 | 85 | 57 | A2 | FF | D6 | 93 | BB | 2B | 3E | 6D | B9 | 67 | CF | 4A | 03 |
| C | 1C | 4E | E3 | 8C | 04 | DA | 84 | 1F | 9D | B2 | 3F | 71 | 26 | 7D | 43 | 99 |
| D | 05 | EA | 66 | C8 | 35 | AF | B0 | CB | 38 | B7 | DB | 55 | 72 | 48 | F6 | ED |
| E | 18 | D7 | 15 | 23 | 12 | 59 | 76 | AE | 70 | 87 | E5 | 32 | CD | 54 | 9B | C2 |
| F | 4D | 5A | 79 | EE | 69 | 11 | BF | E7 | BC | CA | A9 | 9E | E4 | 3D | 92 | 2A |

Table 5: The LUT of $\dot{\pi}_{\lambda,\tau}$.

### 3.2 An instance of the class of nonlinear bijective transformations $\hat{\pi}_{\psi,h}$

Let $\mathbb{F}_{2^4} = \mathbb{F}_2[\xi]/\xi^4 \oplus \xi \oplus 1$ and $\hat{\pi}_{\psi,\mathcal{I}}$ be an instance of the class of nonlinear bijective transformation $\hat{\pi}_{\psi,h}$ by choosing:

- An affine bijective transformation $\mathcal{A} \in \mathsf{GA}_8(\mathbb{F}_2)$ and linear bijective transformation $\mathcal{L} \in \mathsf{GL}_8(\mathbb{F}_2)$;
- The finite field inversion function $\mathcal{I}$ over $\mathbb{F}_{2^4}$ defined by Equation (1);
- A non-bijective function $\psi$ of 4-bit, defined over $\mathbb{F}_{2^4}$, which have not preimage for $\mathsf{0}$.

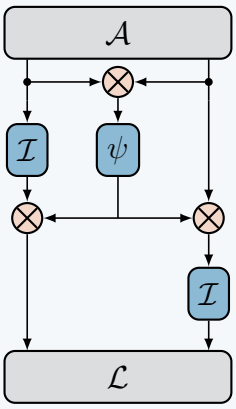


Figure 4: High level of view of $\hat{\pi}_{\psi,\mathcal{I}}$.

For the input value $(l\|r) \in \mathbb{F}_2^8$, the corresponding output value $(l''\|r'') \in \mathbb{F}_2^8$ is computed as $(l''\|r'') = (\mathcal{L} \circ \pi_{\psi,\mathcal{I}} \circ \mathcal{A})\,(l\|r)$, where

1. $\pi_{\psi,\mathcal{I}}(l\|r) = (l'\|r')$, being
$$l' = \mathcal{I}(l) \otimes \psi(l \otimes r),\ r' = \mathcal{I}(r \otimes \psi(l \otimes r)), \tag{9}$$
2. the affine and linear transformations are defined by
$$\mathcal{A}(l_3, l_2, l_1, l_0, r_3, r_2, r_1, r_0) = (l_1, r_3, r_0, l_0, r_1, l_3, r_2 \oplus 1, l_2), \tag{10}$$
$$\mathcal{L}(l_3, l_2, l_1, l_0, r_3, r_2, r_1, r_0) = (r_1, l_1, l_0, r_3, l_2, r_0, r_2, l_3), \tag{11}$$
respectively.

The tabular representations of the nonlinear transformations $\mathcal{I}$ and $\psi$ are given in Table 2 and Table 6 respectively.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\psi$ | E | 9 | F | 4 | 6 | D | 7 | 1 | 6 | 4 | 2 | 9 | D | D | F | B |

Table 6: Tabular representation of the 4-bit non-bijective function $\psi$.

The non-bijective function $\psi$ was found by using the algorithm described in [13]. For the mentioned structural elements of $\hat{\pi}_{\psi,h}$ we obtain the S-Box $\hat{\pi}_{\psi,\mathcal{I}}$. The $\mathsf{LUT}$ of this S-Box is showed in Table 7, where, for example, for the input value $\mathsf{02}$ the corresponding output value is $\mathsf{86}$, i.e., $\hat{\pi}_{\psi,\mathcal{I}}(\mathsf{02}) = \mathsf{86}$.

| Cryptographic Properties of $\hat{\pi}_{\psi,\mathcal{I}}$ | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Nonlinearity – 104 Algebraic Degree – 7<br>Differential Uniformity – 6 Graph Algebraic Immunity – 3(441) | | | | | | | | | | | | | | | | |
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| 0 | 10 | F0 | 86 | 33 | 00 | 68 | 80 | C6 | 91 | A8 | F8 | ED | 41 | 09 | 2A | 3C |
| 1 | 6B | 8D | C2 | 47 | 49 | 61 | DC | 7E | 3E | 0A | E9 | E5 | 01 | 40 | 64 | 15 |
| 2 | B7 | 1D | 74 | C9 | 28 | 08 | 7C | AB | 2C | DD | BC | B3 | 48 | 20 | FA | 8F |
| 3 | 24 | 89 | 27 | D8 | 69 | 60 | 9D | BE | F6 | C0 | D4 | 46 | 29 | 21 | 25 | 1E |
| 4 | 04 | C1 | 96 | EF | 84 | 59 | 82 | 2B | 5F | CF | B5 | 88 | 87 | CA | EC | 7D |
| 5 | 98 | 57 | B1 | BA | A1 | 3F | 1C | D2 | A5 | F1 | 99 | 78 | 4A | 4C | 0B | 0D |
| 6 | DF | 76 | 6F | 93 | 8E | 75 | 9A | 70 | D3 | 97 | E7 | 51 | 63 | 50 | 72 | E0 |
| 7 | 1B | 5C | A7 | 7A | 17 | E1 | 31 | 19 | 73 | 34 | 7F | 5A | 22 | 6E | D1 | A4 |
| 8 | 14 | FF | 06 | 9F | 02 | 66 | 16 | AF | FB | EE | 5B | F5 | B2 | 55 | DB | F7 |
| 9 | 6C | F9 | FD | EA | 4E | 45 | BB | D9 | DE | F2 | 43 | 13 | 77 | 7B | D5 | 9E |
| A | FE | 2E | 23 | D0 | 71 | 81 | 2F | CE | 1F | 56 | 35 | 83 | 65 | 0C | 42 | C5 |
| B | 07 | 30 | C4 | 3D | A9 | D7 | 5D | 9B | E3 | AA | 11 | 52 | 54 | DA | F4 | AC |
| C | 90 | 4F | 94 | EB | 92 | 32 | 12 | 44 | 5E | 53 | A0 | 36 | BD | B8 | 79 | B6 |
| D | A2 | 39 | F3 | 3A | B9 | 05 | AD | D6 | 6D | A6 | 0E | E4 | E6 | 6A | CC | 85 |
| E | FC | 38 | CD | E2 | C7 | 4B | C3 | 58 | 0F | 9C | 37 | 8A | CB | 8B | 4D | C8 |
| F | 8C | 2D | 3B | E8 | 26 | 67 | AE | 95 | B4 | 1A | 18 | 62 | B0 | BF | A3 | 03 |

Table 7: The LUT of $\hat{\pi}_{\psi,\mathcal{I}}$.

### 3.3 The S-Box of the block cipher Kuznyechik

In the block cipher Kuznyechik (described in [14]) is used a 8-bit S-Box denoted in this paper as $\tilde{\pi}_{\mathsf{Kuz}}$. The authors of [5] have suggested a representation of this S-Box by it TU-decomposition based on 4-bit nonlinear transformations and the multiplication over the finite field $\mathbb{F}_{2^4} = \mathbb{F}_2[\xi]/\xi^4 \oplus \xi^3 \oplus 1$. The S-Box $\tilde{\pi}_{\mathsf{Kuz}}$ employ:

– A linear bijective transformations $\mathcal{L}_i \in \mathsf{GL}_8(\mathbb{F}_2), i = 1, 2$;

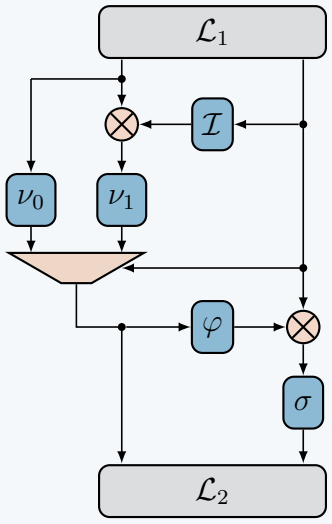



Figure 5: High level of view of $\tilde{\pi}_{\mathsf{Kuz}}$.

– The finite field inversion function $\mathcal{I}$ over $\mathbb{F}_{2^4} = \mathbb{F}_2[\xi]/\xi^4 \oplus \xi^3 \oplus 1$ defined by Equation (1);

– The 4-bit nonlinear transformations $\nu_0, \nu_1, \varphi$ and $\sigma$.

Then for the input value $(l\|r) \in \mathbb{F}_2^8$, the corresponding output value $(l''\|r'') \in \mathbb{F}_2^8$ is computed as $\tilde{\pi}_{\mathsf{Kuz}}(l\|r) = (\mathcal{L}_2 \circ \pi_{\mathsf{Kuz}} \circ \mathcal{L}_1)(l\|r)$, where

1. $\pi_{\mathsf{Kuz}}(l\|r) = (l'\|r')$, being

$$l' = \begin{cases} \nu_0(l), \text{ if } r = 0, \\ \nu_1(l \otimes \mathcal{I}(r)), \text{ if } r \neq 0, \end{cases} \qquad r' = \sigma(r \otimes \varphi(l')). \tag{12}$$

2. The linear bijective transformations $\mathcal{L}_i$[c] $\in \mathsf{GL}_8(\mathbb{F}_2)$ are defined as:

$$\begin{gathered} \mathcal{L}_1(l_3, l_2, l_1, l_0, r_3, r_2, r_1, r_0) = \\ = (r_3, x_0, x_0 \oplus r_1, x_0 \oplus x_2 \oplus l_1 \oplus r_2, x_2, l_2 \oplus r_2, x_1 \oplus l_0, l_1), \end{gathered} \tag{13}$$

where $x_0 = l_2 \oplus r_0,\ x_1 = r_3 \oplus r_1,\ x_2 = l_3 \oplus x_1$;

$$\begin{gathered} \mathcal{L}_2(l_3, l_2, l_1, l_0, r_3, r_2, r_1, r_0) = \\ = (y, r_2, l_1, y \oplus l_0 \oplus l_3, r_3, r_2 \oplus l_2, r_1 \oplus l_3, r_0), \end{gathered} \tag{14}$$

where $y = r_1 \oplus r_3$.

The tabular representations of the 4-bit nonlinear transformations $\mathcal{I}, \nu_0, \nu_1, \phi$ and $\sigma$ are given in Table 8.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\mathcal{I}$ | 0 | 1 | C | 8 | 6 | F | 4 | E | 3 | D | B | A | 2 | 9 | 7 | 5 |
| $\nu_0$ | 2 | 5 | 3 | B | 6 | 9 | E | A | 0 | 4 | F | 1 | 8 | D | C | 7 |
| $\nu_1$ | 7 | 6 | C | 9 | 0 | F | 8 | 1 | 4 | 5 | B | E | D | 2 | 3 | A |
| $\varphi$ | B | 2 | B | 8 | C | 4 | 1 | C | 6 | 3 | 5 | 8 | E | 3 | 6 | B |
| $\sigma$ | C | D | 0 | 4 | 8 | B | A | E | 3 | 9 | 5 | 2 | F | 1 | 6 | 7 |

Table 8: The LUT of the 4-bit nonlinear transformations of $\tilde{\pi}_{\mathsf{Kuz}}$.

The LUT of the S-Box $\tilde{\pi}_{\mathsf{Kuz}}$ is compiled in Table 9, where, for example, for the input value 4B the corresponding output value is A2, i.e., $\tilde{\pi}_{\mathsf{Kuz}}(4B) = A2$.

| Cryptographic Properties of $\tilde{\pi}_{\mathsf{Kuz}}$ | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Nonlinearity – 100 | | | | | | | | Algebraic Degree – 7 | | | | | | | | |
| Differential Uniformity – 8 | | | | | | | | Graph Algebraic Immunity – 3(441) | | | | | | | | |
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |

[c]In this case, $\mathcal{L}_i(x) = \mathcal{L}_i \times x^\top, i = 1, 2$.

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | FC | EE | DD | 11 | CF | 6E | 31 | 16 | FB | C4 | FA | DA | 23 | C5 | 04 | 4D |
| 1 | E9 | 77 | F0 | DB | 93 | 2E | 99 | BA | 17 | 36 | F1 | BB | 14 | CD | 5F | C1 |
| 2 | F9 | 18 | 65 | 5A | E2 | 5C | EF | 21 | 81 | 1C | 3C | 42 | 8B | 01 | 8E | 4F |
| 3 | 05 | 84 | 02 | AE | E3 | 6A | 8F | A0 | 06 | 0B | ED | 98 | 7F | D4 | D3 | 1F |
| 4 | EB | 34 | 2C | 51 | EA | C8 | 48 | AB | F2 | 2A | 68 | A2 | FD | 3A | CE | CC |
| 5 | B5 | 70 | 0E | 56 | 08 | 0C | 76 | 12 | BF | 72 | 13 | 47 | 9C | B7 | 5D | 87 |
| 6 | 15 | A1 | 96 | 29 | 10 | 7B | 9A | C7 | F3 | 91 | 78 | 6F | 9D | 9E | B2 | B1 |
| 7 | 32 | 75 | 19 | 3D | FF | 35 | 8A | 7E | 6D | 54 | C6 | 80 | C3 | BD | 0D | 57 |
| 8 | DF | F5 | 24 | A9 | 3E | A8 | 43 | C9 | D7 | 79 | D6 | F6 | 7C | 22 | B9 | 03 |
| 9 | E0 | 0F | EC | DE | 7A | 94 | B0 | BC | DC | E8 | 28 | 50 | 4E | 33 | 0A | 4A |
| A | A7 | 97 | 60 | 73 | 1E | 00 | 62 | 44 | 1A | B8 | 38 | 82 | 64 | 9F | 26 | 41 |
| B | AD | 45 | 46 | 92 | 27 | 5E | 55 | 2F | 8C | A3 | A5 | 7D | 69 | D5 | 95 | 3B |
| C | 07 | 58 | B3 | 40 | 86 | AC | 1D | F7 | 30 | 37 | 6B | E4 | 88 | D9 | E7 | 89 |
| D | E1 | 1B | 83 | 49 | 4C | 3F | F8 | FE | 8D | 53 | AA | 90 | CA | D8 | 85 | 61 |
| E | 20 | 71 | 67 | A4 | 2D | 2B | 09 | 5B | CB | 9B | 25 | D0 | BE | E5 | 6C | 52 |
| F | 59 | A6 | 74 | D2 | E6 | F4 | B4 | C0 | D1 | 66 | AF | C2 | 39 | 4B | 63 | B6 |

Table 9: The LUT of $\tilde{\pi}_{\mathsf{Kuz}}$.

# 4 Bit-Slice representations of $\pi'_{h,\mathcal{I}}$, $\dot{\pi}_{\lambda,\tau}$, $\hat{\pi}_{\psi,\mathcal{I}}$ and $\tilde{\pi}_{\mathsf{Kuz}}$

The current section is devoted to the problem of finding low gate count logic circuit representations for the S-Boxes $\pi'_{h,\mathcal{I}}$, $\dot{\pi}_{\lambda,\tau}$, $\hat{\pi}_{\psi,\mathcal{I}}$ and $\tilde{\pi}_{\mathsf{Kuz}}$. Our solution to this problem is based on a combination of analytical methods with a open source tool provided `sboxgates`. Trough the section, the logical operations `XOR`, `AND`, `OR` and `NOT` showed in Figure 1 shall be denoted, for simplicity, by $\oplus, \wedge, \vee$ and $\neg$ respectively.

## 4.1 The open source tool `sboxgates`

Finding the logic circuit representation with the fewest possible gates is known to be an NP-complete problem [7, 15]. However, there exist a way to find a low gate count logic circuit representation of an S-Box given its lookup table by using the Kwan's algorithm, which performs a heuristic search. Although not optimal, this method has been shown to generate significantly better results than previous approaches [16].

The open source tool `sboxgates` implements Kwan's algorithm and supports generation of logic circuits for S-Boxes with up to 8 input bits using any subset of the 16 possible two-input boolean functions. The generated

logic circuit representation of an S-box can be directly used in applications such as: creating Bit-Slice implementations in software for CPUs and GPUs, creating small chip area or low gate count S-Boxes for application specific integrated circuits (ASICs) or field programmable gate arrays (FPGAs), and compact satisfiability (SAT) problem generation. More information about this tool can be found in [10].

We shall use `sboxgates` to find the Bit-Slice implementations of the 4-bit components employed in the design of $\pi'_{h,\mathcal{I}}$, $\dot{\pi}_{\lambda,\tau}$, $\hat{\pi}_{\psi,\mathcal{I}}$ and $\tilde{\pi}_{\mathsf{Kuz}}$.

### 4.2 Bit-Slice representation of $\pi'_{h,\mathcal{I}}$

As was described in the Subsection 3.1, from the high level of view of $\pi'_{h,\mathcal{I}}$ we obtain that it Bit-Slice Gate Complexity can be expressed by

$$\begin{aligned} \mathtt{BGC}\left(\pi'_{h,\mathcal{I}}\right) &= \mathtt{BGC}\left(\mathcal{A}\right) + 2\cdot\mathtt{BGC}\left(\otimes\right) + \mathtt{BGC}\left(\mathcal{F}_1\right) + \mathtt{BGC}\left(\mathcal{F}_2\right) + \\ &+ 2\cdot\mathtt{BGC}\left(\mathcal{I}\right) + 2\cdot\mathtt{BGC}\left(h\right) + \mathtt{BGC}\left(\mathcal{L}\right), \end{aligned} \tag{15}$$

where by $\mathcal{F}_i, i \in \{1,2\}$, are denoted the left and right branches given by Equation (2) containing the conditionals *if*.

**The affine and linear transformations $\mathcal{A}$ and $\mathcal{L}$:**
From equations (3) and (4) is evident that $\mathtt{BGC}\left(\mathcal{A}\right) = 1$ and $\mathtt{BGC}\left(\mathcal{L}\right) = 0$ respectively.

**The multiplication in the finite field $\mathbb{F}_{2^4} = \mathbb{F}_2[\xi]/\xi^4 \oplus \xi \oplus 1$:**
Let $a(\xi),\ b(\xi) \in \mathbb{F}_{2^4}$. The element $c(\xi) = a(\xi) \otimes b(\xi) \in \mathbb{F}_{2^4}$ can be calculated by

$$\begin{aligned} c(\xi) &= a(\xi)\otimes b(\xi) = (a_3\cdot\xi^3 \oplus a_2\cdot\xi^2 \oplus a_1\cdot\xi \oplus a_0)(b_3\cdot\xi^3 \oplus b_2\cdot\xi^2 \oplus b_1\cdot\xi \oplus b_0) = \\ &= a_3b_3(\xi^3\oplus\xi^2) \oplus a_3b_2(\xi^2\oplus\xi) \oplus a_3b_1(\xi\oplus 1) \oplus a_3b_0\xi^3 \oplus \\ &\quad \oplus a_2b_3(\xi^2\oplus\xi) \oplus a_2b_2(\xi\oplus 1) \oplus a_2b_1\xi^3 \oplus a_2b_0\xi^2 \oplus \\ &\quad \oplus a_1b_3(\xi\oplus 1) \oplus a_1b_2\xi^3 \oplus a_1b_1\xi^2 \oplus a_1b_0\xi \oplus \\ &\quad \oplus a_0b_3\xi^3 \oplus a_0b_2\xi^2 \oplus a_0b_1\xi \oplus a_0b_0 = \\ &= (a_3b_3 \oplus a_3b_0 \oplus a_2b_1 \oplus a_1b_2 \oplus a_0b_3)\xi^3 \oplus \\ &\quad \oplus (a_3b_3 \oplus a_3b_2 \oplus a_2b_3 \oplus a_2b_0 \oplus a_1b_1 \oplus a_0b_2)\xi^2 \oplus \\ &\quad \oplus (a_3b_2 \oplus a_3b_1 \oplus a_2b_3 \oplus a_2b_2 \oplus a_1b_3 \oplus a_1b_0 \oplus a_0b_1)\xi \oplus \\ &\quad \oplus a_3b_1 \oplus a_2b_2 \oplus a_1b_3 \oplus a_0b_0 \quad . \end{aligned}$$

This expression can be reduced to the following

$$\begin{aligned} c_3 &= a_3 \wedge t_0 \oplus a_2 \wedge b_1 \oplus a_1 \wedge b_2 \oplus a_0 \wedge b_3, \\ c_2 &= a_3 \wedge t_1 \oplus a_2 \wedge t_0 \oplus a_1 \wedge b_1 \oplus a_0 \wedge b_2, \\ c_1 &= a_3 \wedge t_2 \oplus a_2 \wedge t_1 \oplus a_1 \wedge t_0 \oplus a_0 \wedge b_1, \\ c_0 &= a_3 \wedge b_1 \oplus a_2 \wedge b_2 \oplus a_1 \wedge b_3 \oplus a_0 \wedge b_0, \end{aligned}$$

where $t_2 = b_1 \oplus b_2$, $t_1 = b_2 \oplus b_3$ and $t_0 = b_0 \oplus b_3$.
As we can see, the multiplication of any two elements from $\mathbb{F}_{2^4} = \mathbb{F}_2[\xi]/\xi^4 \oplus \xi \oplus 1$ require 16 `AND` and 15 `XOR`, which mean that $\texttt{BGC}(\otimes) = 31$. The Circuit representation of the finite field multiplication is showed in Figure 6.

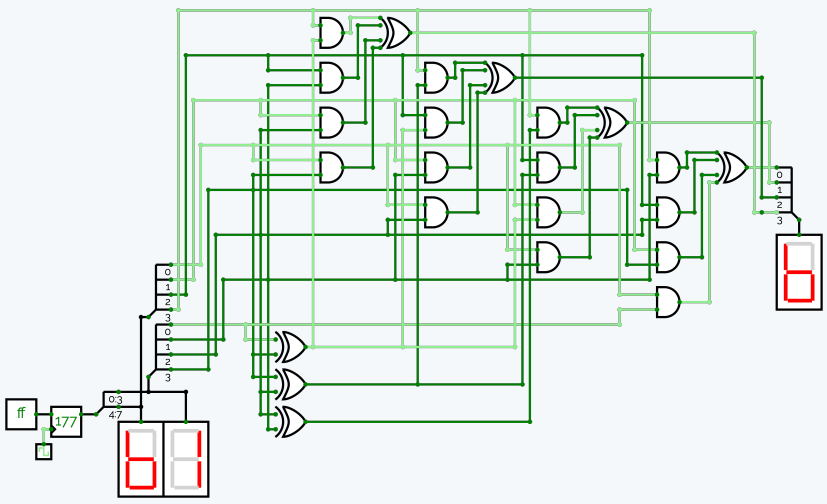



Figure 6: Combinatorial circuit of the multiplication in $\mathbb{F}_{2^4} = \mathbb{F}_2[\xi]/\xi^4 \oplus \xi \oplus 1$ for the elements `B` and `1`.

**Eliminating the conditionals *if* inside the functions $\mathcal{F}_i, i \in \{1, 2\}$:** The Bit-Slice representations of $\mathcal{F}_i, i \in \{1, 2\}$ is obtained from the following relations $\mathcal{F}_1(l\|r) = h(l) \cdot \mathsf{Ind}(r, 0) \oplus \mathcal{I}(l \otimes r)$ and $\mathcal{F}_2(l'\|r) = h(r) \cdot \mathsf{Ind}(l', 0) \oplus l' \otimes \mathcal{I}(r)$, which at the same time requires that $h, \mathcal{I}$ and $\otimes$ to be expressed in terms of Boolean operations $\oplus, \wedge, \vee$ and $\neg$ respectively. From the definition of the Indicator function, we obtain that

$$\mathsf{Ind}(r, 0) = \neg\left(r_3 \vee r_2 \vee r_1 \vee r_0\right), \mathsf{Ind}(l', 0) = \neg\left(l'_3 \vee l'_2 \vee l'_1 \vee l'_0\right).$$

Using the fixed points of $\mathcal{I}$, it is not hard to see that functions $\mathcal{F}_1$ and $\mathcal{F}_2$ can be rewritten as

$$\begin{aligned}
\mathcal{F}_1(l\|r) &= h(l) \cdot \mathsf{Ind}(r, 0) \oplus \mathcal{I}(l \otimes r) \oplus \left(h(l) \cdot \mathsf{Ind}(r, 0)\right) \cdot \mathcal{I}(l \otimes r), \\
\mathcal{F}_2(l'\|r) &= h(r) \cdot \mathsf{Ind}(l', 0) \oplus \mathcal{I}\left((l')^{14} \otimes r\right) \oplus \left(h(r) \cdot \mathsf{Ind}(l', 0)\right) \cdot \mathcal{I}\left((l')^{14} \otimes r\right).
\end{aligned}$$

If denote by $\mathcal{F}_i^{(j)}, h^{(j)}, \mathcal{I}^{(j)}$, $i = 1, 2$ the coordinate functions of the transformations $\mathcal{F}_i, h, \mathcal{I}$ and using the Boolean identity $a \vee b = a \oplus b \oplus a \wedge b$[d], then

$$\mathcal{F}_1^{(j)}(l\|r) = h^{(j)}(l_3, l_2, l_1, l_0) \wedge \neg\left(r_3 \vee r_2 \vee r_1 \vee r_0\right) \vee \mathcal{I}^{(j)}(v_3, v_2, v_1, v_0),$$

[d]With this identity, the Gate Equivalent complexity of $\pi'_{h,\mathcal{I}}$ can be reduced because `XOR` operations are more expensive than `OR` instructions.

where $j = 0, 1, 2, 3$ and $v_i = (l \otimes r)^{(i)}, i = 0, 1, 2, 3$ are the coordinate functions of the multiplication over $\mathbb{F}_{2^4}$. Let $t_1 \leftarrow \neg(r_3 \vee r_2 \vee r_1 \vee r_0)$, then we obtain the following relations

$$\mathcal{F}_1^{(j)} = h^{(j)} \wedge t_1 \vee \mathcal{I}^{(j)}.$$

Similarly, $\mathcal{F}_2(l\|r)$ can be expressed as

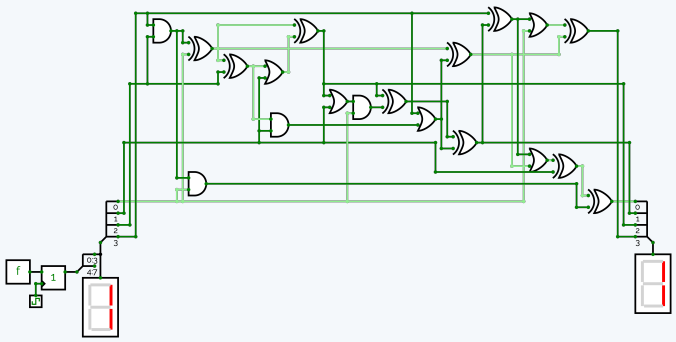
$$\mathcal{F}_2^{(j)}(l_1\|r) = h^{(j)}(r_3, r_2, r_1, r_0) \wedge \neg(l'_3 \vee l'_2 \vee l'_1 \vee l'_0) \vee \mathcal{I}^{(j)}(v_3, v_2, v_1, v_0),$$

where $j = 0, 1, 2, 3$, $v_i = \left((l')^{14} \otimes r\right)^{(i)}, i = 0, 1, 2, 3$. Let $t_2 \leftarrow \neg(l'_3 \vee l'_2 \vee l'_1 \vee l'_0)$, then we obtain the next relations

$$\mathcal{F}_2^{(j)} = h^{(j)} \wedge t_2 \vee \mathcal{I}^{(j)}.$$

Thus, for transformations $\mathcal{F}_i, i = 1, 2$, we have

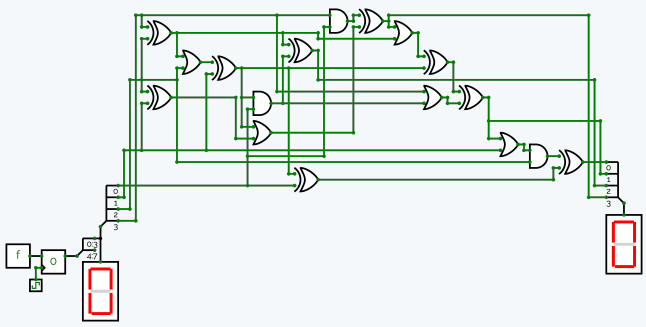
$$\texttt{BGC}(\mathcal{F}_1) = \texttt{BGC}(\mathcal{F}_2) = 12.$$



$h(x_1, x_2, x_3, x_4) = (z_1, z_2, z_3, z_4)$

| | |
|---|---|
| $t_4 \leftarrow x_3 \wedge x_4$ | $z_2 \leftarrow t_{11} \oplus t_{13}$ |
| $t_5 \leftarrow t_4 \oplus x_1$ | $t_{15} \leftarrow t_{13} \oplus t_5$ |
| $t_6 \leftarrow t_5 \oplus x_3$ | $t_{16} \leftarrow z_2 \oplus x_4$ |
| $t_7 \leftarrow t_6 \vee x_2$ | $t_{17} \leftarrow t_{16} \vee x_1$ |
| $z_3 \leftarrow t_5 \oplus t_7$ | $z_4 \leftarrow t_{15} \oplus t_{17}$ |
| $t_9 \leftarrow z_3 \vee x_2$ | $t_{19} \leftarrow t_{16} \vee t_{15}$ |
| $t_{10} \leftarrow t_9 \wedge x_1$ | $t_{20} \leftarrow t_{19} \oplus x_2$ |
| $t_{11} \leftarrow z_3 \oplus t_{10}$ | $t_{21} \leftarrow t_4 \wedge x_1$ |
| $t_{12} \leftarrow t_6 \wedge x_2$ | $z_1 \leftarrow t_{20} \oplus t_{21}$ |
| $t_{13} \leftarrow t_{12} \vee x_4$ | |

Figure 7: Bit-Slice representation and combinatorial circuit of $h$, where for the input value `1`, the corresponding output value is `1`.



$\mathcal{I}(x_1, x_2, x_3, x_4) = (z_1, z_2, z_3, z_4)$

| | |
|---|---|
| $t_4 \leftarrow x_3 \oplus x_4$ | $t_{13} \leftarrow z_4 \vee t_4$ |
| $t_5 \leftarrow x_3 \vee t_4$ | $t_{14} \leftarrow t_{13} \oplus t_6$ |
| $t_6 \leftarrow t_5 \oplus x_2$ | $t_{15} \leftarrow t_7 \vee x_4$ |
| $t_7 \leftarrow t_6 \wedge x_1$ | $z_2 \leftarrow t_{14} \oplus t_{15}$ |
| $z_3 \leftarrow t_4 \oplus t_7$ | $t_{17} \leftarrow x_1 \oplus t_6$ |
| $t_9 \leftarrow x_2 \oplus x_3$ | $t_{18} \leftarrow z_2 \vee x_2$ |
| $t_{10} \leftarrow t_9 \vee t_6$ | $t_{19} \leftarrow t_{18} \wedge x_3$ |
| $t_{11} \leftarrow x_4 \wedge x_1$ | $z_1 \leftarrow t_{17} \oplus t_{19}$ |
| $z_4 \leftarrow t_{10} \oplus t_{11}$ | |

Figure 8: Bit-Slice representation and combinatorial circuit of $\mathcal{I}$, where for the input value `0`, the corresponding output value is `0`.

**The 4-bit nonlinear transformations $h$ and $\mathcal{I}$:**
Using the open source software `sboxgates` [10] were found the Bit-Slice representations of $h$ and $\mathcal{I}$, which are given in Figures 7 and 8 respectively. From these representations we have that $\mathtt{BGC}(h) = 19$ and $\mathtt{BGC}(\mathcal{I}) = 17$ respectively.

**The resulting S-Box $\pi'_{h,\mathcal{I}}$:**
Finally, from the Equation (15) we obtain that

$$\mathtt{BGC}\left(\pi'_{h,\mathcal{I}}\right) = 1 + 2 \cdot 31 + 2 \cdot 12 + 2 \cdot 17 + 2 \cdot 19 + 0 = 159.$$

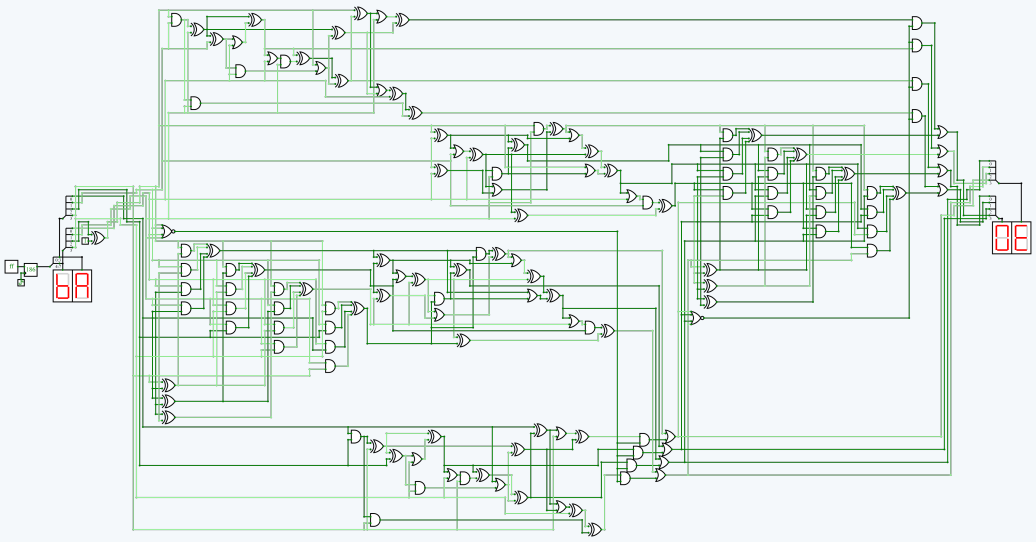



Figure 9: Combinatorial circuit of the S-Box $\pi'_{h,\mathcal{I}}$, where for the input value `BA`, the corresponding output value is `0E`.

### 4.2.1 Bit-Slice representation of $\dot{\pi}_{\lambda,\tau}$

Using the open source software `sboxgates` [10] we have found the Bit-Slice representation of the transformation $\tau$, which is showed in the Figure 10. From this representation we obtain that $\mathtt{BGC}(\tau) = 16$ and reutilizing the approaches described in Subsection 4.2 we derive that

$$\begin{aligned}\mathtt{BGC}\left(\dot{\pi}_{\lambda,\tau}\right) &= \mathtt{BGC}(\mathcal{A}) + 2 \cdot \mathtt{BGC}(\otimes) + \mathtt{BGC}(\mathcal{F}_1) + \mathtt{BGC}(\mathcal{F}_2) + \\ &+ 2 \cdot \mathtt{BGC}(\tau) + 2 \cdot \mathtt{BGC}(\lambda) + \mathtt{BGC}(\mathcal{L}).\end{aligned}$$

Now, considering equations (6), (7) and (8) we obtain

$$\mathtt{BGC}\left(\dot{\pi}_{\lambda,\tau}\right) = 1 + 2 \cdot 31 + 2 \cdot 12 + 2 \cdot 16 = 119.$$

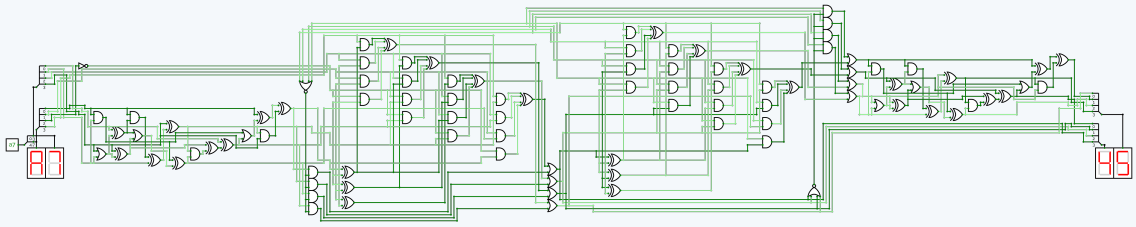


Figure 10: Combinatorial circuit of the S-Box $\dot{\pi}_{\lambda,\tau}$, where for the input value `A7`, the corresponding output value is `45`.

### 4.3 Bit-Slice representation of $\hat{\pi}_{\psi,\mathcal{I}}$

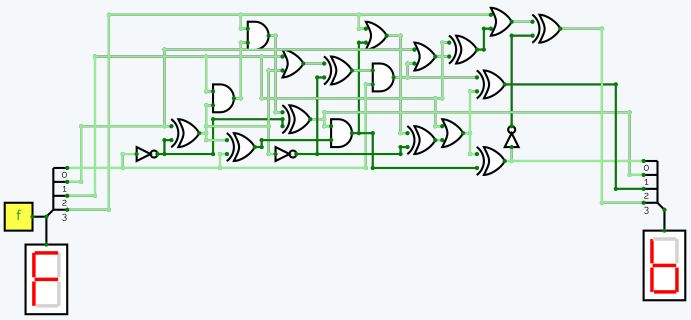
As was described in Subsection 3.2, from the high level of view of of the nonlinear bijective transformation $\hat{\pi}_{\psi,\mathcal{I}}$ we deduce that the Bit-Slice Gate Complexity of $\hat{\pi}_{\psi,\mathcal{I}}$ can be expressed by

$$\mathtt{BGC}\left(\hat{\pi}_{\psi,\mathcal{I}}\right) = \mathtt{BGC}\left(\mathcal{A}\right) + 3 \cdot \mathtt{BGC}\left(\otimes\right) + 2 \cdot \mathtt{BGC}\left(\mathcal{I}\right) + \mathtt{BGC}\left(\psi\right) + \mathtt{BGC}\left(\mathcal{L}\right). \quad (16)$$

**The affine and linear transformations $\mathcal{A}$ and $\mathcal{L}$:**
From equations (10) and (11) is evident that $\mathtt{BGC}\left(\mathcal{A}\right) = 1$ and $\mathtt{BGC}\left(\mathcal{L}\right) = 0$ respectively.

**The multiplication in the finite field $\mathbb{F}_{2^4} = \mathbb{F}_2[\xi]/\xi^4 \oplus \xi \oplus 1$:**
In this cases the multiplication in $\mathbb{F}_{2^4}$ coincide with the finite field multiplication given in Section 3.1. Therefore $\mathtt{BGC}\left(\otimes\right) = 31$.

**The 4-bit nonlinear transformations $\psi$ and $\mathcal{I}$:**
Using the open source software `sboxgates` [10] was found the Bit-Slice representation of $\psi$, which is showed in the Figure 11. From Figure 8 we have that $\mathtt{BGC}\left(\mathcal{I}\right) = 19$ and from Figure 11 we have $\mathtt{BGC}\left(\psi\right) = 21$.



$$\psi(x_3, x_2, x_1, x_0) = (z_3, z_2, z_1, z_0)$$

| | |
|---|---|
| $t_4 \leftarrow \neg x_0$ | $z_0 \leftarrow t_{10} \oplus t_{14}$ |
| $t_5 \leftarrow t_4 \oplus x_1$ | $t_{16} \leftarrow t_5 \vee x_2$ |
| $t_6 \leftarrow t_5 \wedge x_2$ | $t_{17} \leftarrow t_{16} \oplus t_{11}$ |
| $t_7 \leftarrow t_6 \wedge x_3$ | $t_{18} \leftarrow t_{17} \wedge x_0$ |
| $z_1 \leftarrow t_4 \oplus t_7$ | $z_2 \leftarrow t_{14} \oplus t_{18}$ |
| $t_9 \leftarrow x_0 \oplus t_5$ | $t_{20} \leftarrow \neg z_0$ |
| $t_{10} \leftarrow t_9 \wedge z_1$ | $t_{21} \leftarrow t_{18} \vee x_1$ |
| $t_{11} \leftarrow \neg t_5$ | $t_{22} \leftarrow t_{21} \oplus x_2$ |
| $t_{12} \leftarrow t_{10} \vee x_3$ | $t_{23} \leftarrow t_{22} \vee x_3$ |
| $t_{13} \leftarrow t_{11} \oplus t_{12}$ | $z_3 \leftarrow t_{20} \oplus t_{23}$ |
| $t_{14} \leftarrow t_{13} \vee x_2$ | |

Figure 11: Bit-Slice representation and combinatorial circuit of $\psi$, where for the input value `F`, the corresponding output value is `B`.

**The resulting S-Box $\hat{\pi}_{\psi,\mathcal{I}}$:**
Finally, from Equation (16) we obtain that

$$\texttt{BGC}\left(\hat{\pi}_{\psi,\mathcal{I}}\right) = 1 + 3 \cdot 31 + 2 \cdot 17 + 21 = 149.$$

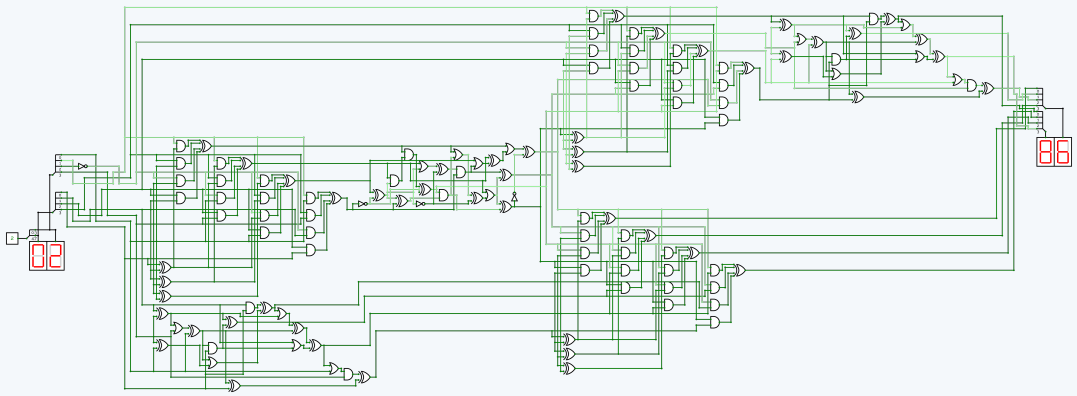



Figure 12: Combinatorial circuit of the S-Box $\hat{\pi}_{\psi,\mathcal{I}}$, where for the input value 02, the corresponding output value is 86.

## 4.4 A more compact Bit-Slice representation of $\tilde{\pi}_{\mathsf{Kuz}}$

The authors of the paper [1] by combining analytical and computer methods propose a Bit-Sliced representation of the S-Box $\tilde{\pi}_{\mathsf{Kuz}}$ which requires 226 Boolean (logical) operations, i.e., $\texttt{BGC}\left(\tilde{\pi}_{\mathsf{Kuz}}\right) = 226$. This value is significantly less than the obtained in 2016 by the method proposed in [6], requiring the latter 681 Boolean operations (254 `AND` and 436 `XOR`).

In the Summary Section of [1], based on our implementation metrics, authors obtain in Table 2, the following values $\texttt{BGC}\left(\varphi\right) = 32$ and $\texttt{BGC}\left(\sigma\right) = 33$ for the 4-bit nonlinear transformations $\varphi$ and $\sigma$ respectively[e]. In Section 4.8 of [1] was represented the coordinate functions of these transformations through their logical Boolean operations, from which it follows that $\texttt{BGC}\left(\varphi\right) = 33$ and $\texttt{BGC}\left(\sigma\right) = 31$ respectively. According to these values we obtain $\texttt{BGC}\left(\tilde{\pi}_{\mathsf{Kuz}}\right) = 225$ and not the 226 Boolean operations as the authors state.

In addition, it can be also checked that in the Section 3.3 of [1] was wrongly assumed that $\nu_1(0) = 2$, when actually from Table 8 we get that $\nu_1(0) = 7$. This mean that after branching elimination the complexity of their representation has increased by 15 operations (and not 13 operations as it was wrongly computed). So the actual number of Boolean operations `XOR`, `AND`, `OR`, `NOT` is 227 and not 226 as the authors claims.

[e]Under these values the authors of [1] get $\texttt{BGC}\left(\tilde{\pi}_{\mathsf{Kuz}}\right) = 226$.

Now we propose a little compact Bit-Slice representation of $\tilde{\pi}_{\mathsf{Kuz}}$. As was described in Subsection 3.3, from the definition of the nonlinear bijective transformation $\tilde{\pi}_{\mathsf{Kuz}}$ we obtain that its Bit-Slice Gate Complexity can be expressed as follows

$$\begin{aligned} \mathtt{BGC}\left(\tilde{\pi}_{\mathsf{Kuz}}\right) &= \mathtt{BGC}\left(\mathcal{L}_1\right) + 2 \cdot \mathtt{BGC}\left(\otimes\right) + \mathtt{BGC}\left(\mathcal{I}\right) + \mathtt{BGC}\left(\nu_0\right) + \mathtt{BGC}\left(\nu_1\right) + \\ &+ \mathtt{BGC}\left(\mathcal{F}\right) + \mathtt{BGC}\left(\varphi\right) + \mathtt{BGC}\left(\sigma\right) + \mathtt{BGC}\left(\mathcal{L}_2\right), \end{aligned} \tag{17}$$

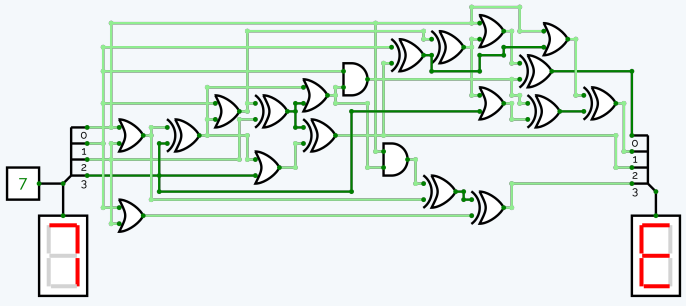
where by $\mathcal{F}$ is denoted the conditional *if* defined in the Equation (12).

**The linear transformations $\mathcal{L}_1$ and $\mathcal{L}_2$:**
From the Equations (13) and (14) is evident that $\mathtt{BGC}\left(\mathcal{L}_1\right) = 9$ and $\mathtt{BGC}\left(\mathcal{L}_2\right) = 5$, respectively.

**The multiplication in the finite field $\mathbb{F}_{2^4} = \mathbb{F}_2[\xi]/\xi^4 \oplus \xi^3 \oplus 1$:**
Similarly to the procedure described in Subsection 4.2 for the multiplication in $\mathbb{F}_{2^4} = \mathbb{F}_2[\xi]/\xi^4 \oplus \xi \oplus 1$ we can obtain that when $\mathbb{F}_{2^4} = \mathbb{F}_2[\xi]/\xi^4 \oplus \xi^3 \oplus 1$, the value of the Bit-Slice Gate Complexity is $\mathtt{BGC}\left(\otimes\right) = 31$.

**Eliminating the conditionals *if* inside the functions $\mathcal{F}$:**
Similarly to the procedure described in the subsection 4.2 for the conditionals *if* defined by the Equation (2) and taking into account approaches described in [1] we can obtain that $\mathtt{BGC}\left(\mathcal{F}\right) = 15$.

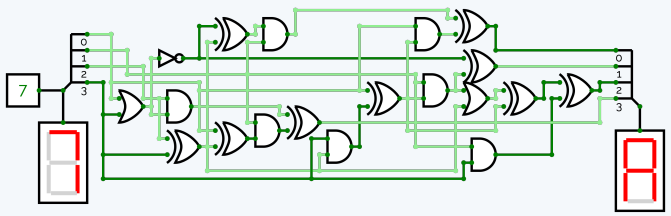
**The 4-bit nonlinear transformations $\mathcal{I}$, $\nu_0$, $\nu_1$, $\varphi$ and $\sigma$:**
Using the open source software `sboxgates` [10] was possible to find the Bit-Slice representation of the transformations $\mathcal{I}$, $\nu_0$, $\nu_1$, $\varphi$ and $\sigma$, which are given in the Figures 13, 14, 15, 16 and 17, respectively. From them is evident that $\mathtt{BGC}\left(\mathcal{I}\right) = 20$, $\mathtt{BGC}\left(\nu_0\right) = 19$, $\mathtt{BGC}\left(\nu_1\right) = 12$, $\mathtt{BGC}\left(\varphi\right) = 18$ and $\mathtt{BGC}\left(\sigma\right) = 19$.




$\mathcal{I}(x_3, x_2, x_1, x_0) = (z_3, z_2, z_1, z_0)$

| | |
|---|---|
| $t_4 \leftarrow x_2 \vee x_0$ | $t_{14} \leftarrow t_{13} \vee x_0$ |
| $t_5 \leftarrow t_4 \oplus x_3$ | $z_0 \leftarrow t_{11} \oplus t_{14}$ |
| $t_6 \leftarrow t_5 \vee x_1$ | $t_{16} \leftarrow t_{10} \wedge x_0$ |
| $t_7 \leftarrow x_2 \oplus t_6$ | $t_{17} \leftarrow t_{16} \oplus t_4$ |
| $t_8 \leftarrow t_5 \vee x_3$ | $t_{18} \leftarrow x_2 \vee x_1$ |
| $z_2 \leftarrow t_7 \oplus t_8$ | $z_3 \leftarrow t_{17} \oplus t_{18}$ |
| $t_{10} \leftarrow t_5 \vee t_7$ | $t_{20} \leftarrow t_{13} \vee x_3$ |
| $t_{11} \leftarrow t_{10} \wedge x_1$ | $t_{21} \leftarrow t_{20} \oplus t_{11}$ |
| $t_{12} \leftarrow z_2 \oplus x_1$ | $t_{22} \leftarrow t_{12} \vee x_0$ |
| $t_{13} \leftarrow t_{12} \oplus t_6$ | $z_1 \leftarrow t_{21} \oplus t_{22}$ |

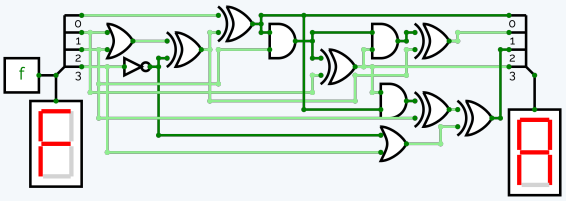
Figure 13: Bit-Slice representation and combinatorial circuit of $\mathcal{I}$, where for the input value 7, the corresponding output value is E.




$$\nu_0(x_3, x_2, x_1, x_0) = (z_3, z_2, z_1, z_0)$$

| | |
|---|---|
| $t_4 \leftarrow x_0 \vee x_3$ | $z_1 \leftarrow t_{10} \oplus t_{13}$ |
| $t_5 \leftarrow t_4 \wedge x_2$ | $t_{15} \leftarrow t_{10} \oplus t_6$ |
| $t_6 \leftarrow x_3 \oplus x_2$ | $t_{16} \leftarrow t_{15} \wedge x_1$ |
| $t_7 \leftarrow t_6 \oplus x_0$ | $t_{17} \leftarrow t_{12} \wedge x_0$ |
| $t_8 \leftarrow t_7 \wedge x_1$ | $z_0 \leftarrow t_{16} \oplus t_{17}$ |
| $z_3 \leftarrow t_5 \oplus t_8$ | $t_{19} \leftarrow t_6 \vee t_{13}$ |
| $t_{10} \leftarrow \neg t_4$ | $t_{20} \leftarrow t_{19} \oplus t_{12}$ |
| $t_{11} \leftarrow x_3 \wedge t_6$ | $t_{21} \leftarrow x_3 \wedge x_1$ |
| $t_{12} \leftarrow t_{11} \oplus x_0$ | $z_2 \leftarrow t_{20} \oplus t_{21}$ |
| $t_{13} \leftarrow t_{12} \wedge x_1$ | |

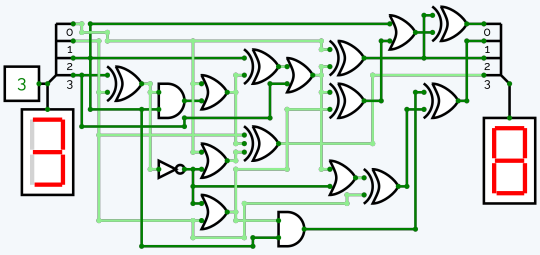
Figure 14: Bit-Slice representation and combinatorial circuit of $\nu_0$, where for the input value 7, the corresponding output value is A.




$$\nu_1(x_3, x_2, x_1, x_0) = (z_3, z_2, z_1, z_0)$$

| | |
|---|---|
| $t_4 \leftarrow \neg x_3$ | $t_{10} \leftarrow z_3 \wedge t_8$ |
| $t_5 \leftarrow x_2 \vee x_1$ | $z_1 \leftarrow t_{10} \oplus t_6$ |
| $t_6 \leftarrow t_4 \oplus t_5$ | $t_{12} \leftarrow z_0 \wedge z_3$ |
| $z_0 \leftarrow t_6 \oplus x_0$ | $t_{13} \leftarrow t_{12} \oplus x_2$ |
| $t_8 \leftarrow x_2 \wedge z_0$ | $t_{14} \leftarrow t_4 \vee x_3$ |
| $z_3 \leftarrow t_8 \oplus x_1$ | $z_2 \leftarrow t_{13} \oplus t_{14}$ |

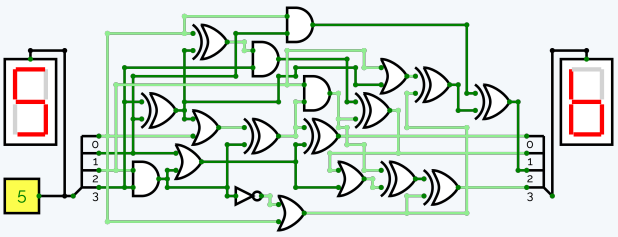
Figure 15: Bit-Slice representation and combinatorial circuit of $\nu_1$, where for the input value F, the corresponding output value is A.




$$\varphi(x_3, x_2, x_1, x_0) = (z_3, z_2, z_1, z_0)$$

| | |
|---|---|
| $t_4 \leftarrow x_1 \oplus x_3$ | $z_3 \leftarrow t_{12} \oplus x_1$ |
| $t_5 \leftarrow t_4 \wedge x_2$ | $t_{14} \leftarrow t_{10} \vee x_1$ |
| $t_6 \leftarrow t_5 \vee x_0$ | $t_{15} \leftarrow t_{14} \oplus t_8$ |
| $t_7 \leftarrow t_6 \oplus x_2$ | $t_{16} \leftarrow t_{15} \vee x_2$ |
| $t_8 \leftarrow t_7 \vee x_3$ | $z_0 \leftarrow z_2 \oplus t_{16}$ |
| $z_2 \leftarrow t_8 \oplus x_0$ | $t_{18} \leftarrow t_{10} \vee t_8$ |
| $t_{10} \leftarrow \neg t_4$ | $t_{19} \leftarrow t_{18} \oplus x_1$ |
| $t_{11} \leftarrow t_{10} \vee x_0$ | $t_{20} \leftarrow t_{14} \wedge x_2$ |
| $t_{12} \leftarrow t_6 \oplus t_{11}$ | $z_1 \leftarrow t_{19} \oplus t_{20}$ |

Figure 16: Bit-Slice representation and combinatorial circuit of $\varphi$, where for the input value 3, the corresponding output value is 8.




$$\sigma(x_3, x_2, x_1, x_0) = (z_3, z_2, z_1, z_0)$$

| | |
|---|---|
| $t_4 \leftarrow x_3 \oplus x_1$ | $t_{14} \leftarrow z_1 \vee t_8$ |
| $t_5 \leftarrow t_4 \vee x_0$ | $t_{15} \leftarrow t_{14} \oplus t_{12}$ |
| $t_6 \leftarrow x_3 \wedge x_2$ | $t_{16} \leftarrow \neg t_6$ |
| $t_7 \leftarrow t_5 \oplus t_6$ | $t_{17} \leftarrow t_{16} \vee x_0$ |
| $t_8 \leftarrow t_6 \vee x_1$ | $z_3 \leftarrow t_{15} \oplus t_{17}$ |
| $z_0 \leftarrow t_7 \oplus t_8$ | $t_{19} \leftarrow t_4 \vee x_2$ |
| $t_{10} \leftarrow t_4 \oplus x_0$ | $t_{20} \leftarrow t_{19} \oplus t_{17}$ |
| $t_{11} \leftarrow t_{10} \wedge x_3$ | $t_{21} \leftarrow x_0 \wedge x_1$ |
| $t_{12} \leftarrow t_7 \wedge x_2$ | $z_2 \leftarrow t_{20} \oplus t_{21}$ |
| $z_1 \leftarrow t_{11} \oplus t_{12}$ | |

Figure 17: Bit-Slice representation and combinatorial circuit of $\sigma$, where for the input value 5, the corresponding output value is B.

**The S-Box $\tilde{\pi}_{\mathsf{Kuz}}$:**
Finally, from the Equation (17) we obtain that

$$\mathtt{BGC}\left(\tilde{\pi}_{\mathsf{Kuz}}\right) = 9 + 2 \cdot 31 + 20 + 19 + 12 + 15 + 18 + 19 + 5 = 179.$$

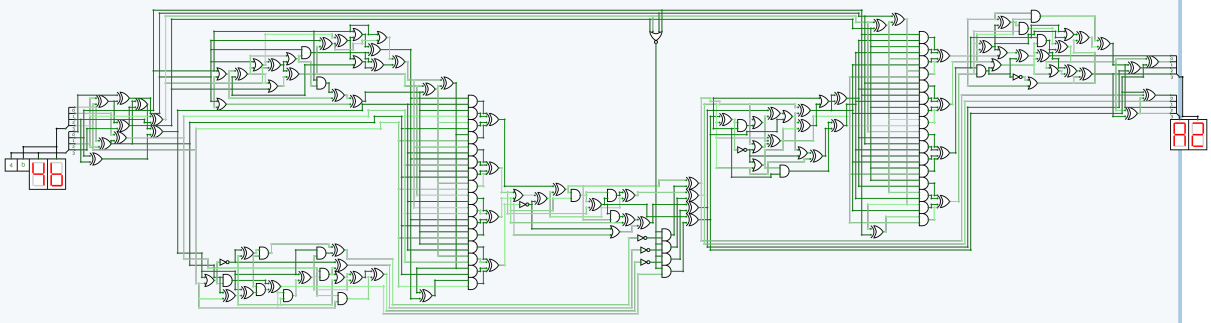



Figure 18: Combinatorial circuit of the S-Box $\tilde{\pi}_{\mathsf{Kuz}}$, where for the input value `4B`, the corresponding output value is `A2`.

# 5 A comparison of some 8-bit S-Boxes

In this section we compare our results with previous designs reported in the state of the art.

| S-Boxes | Logical Operations | | | | BGC $(\cdot)$ | GEC $(\cdot)$ | | Basic Cryptographic Parameters | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | XOR | AND | OR | NOT | | UMC/180nm | TSMC/65nm | NL | AD | DU | AI |
| $\pi_{\mathsf{Whp}}$ | 58 | 15 | 21 | 7 | 101 | 226.57 | 231.50 | 100 | 6 | 8 | 3(441) |
| $\pi_{\mathsf{AES}}$ | 83 | 32 | 0 | 0 | 115 | 291.56 | 297.00 | 112 | 7 | 4 | 2(39) |
| $\dot{\pi}_{\lambda,\tau}$ | 49 | 48 | 20 | 2 | 119 | 238.78 | 250.00 | 108 | 7 | 6 | 3(441) |
| $\hat{\pi}_{\psi,\mathcal{I}}$ | 73 | 58 | 15 | 3 | 149 | 318.10 | 330.00 | 104 | 7 | 6 | 3(441) |
| $\pi'_{h,\mathcal{I}}$ | 69 | 54 | 34 | 2 | 159 | 325.38 | 340.00 | 108 | 7 | 6 | 3(441) |
| $\tilde{\pi}_{\mathsf{Kuz}}$ | 94 | 54 | 26 | 5 | 179 | 391.75 | 404.50 | 100 | 7 | 8 | 3(441) |
| $\pi_{\mathsf{Kal}}$ | NR | | | | 361 | NR | | 104 | 7 | 8 | 3(441) |

Table 10: A comparison between Bit-Slice Gate/Gate Equivalent Complexities and the cryptographic parameters of some 8-bit S-Boxes (NR means "not reported"). The basic cryptographic parameters: nonlinearity, (algebraic) minimum degree, differential uniformity and (graph) algebraic immunity are denoted by `NL`, `AD`, `DU` and `AI`.

In Table 10 is represented the total number of each logical operations needed to implement the S-Boxes $\pi'_{h,\mathcal{I}}$, $\dot{\pi}_{\lambda,\tau}$, $\hat{\pi}_{\psi,\mathcal{I}}$, $\tilde{\pi}_{\mathsf{Kuz}}$ and the corresponding $\mathtt{GEC}\left(\cdot\right)$ value, when the technologies `UMC/180nm` and `TSMC/65nm` are employed. Moreover, these values are also calculated for the Bit-Slice representations of some 8-bit S-Boxes used in modern cryptosystems, such as: `Whirlpool` [2], `AES` [7] and `Kalyna` [18], denoted in this paper by $\pi_{\mathsf{Whp}}$, $\pi_{\mathsf{AES}}$ and $\pi_{\mathsf{Kal}}$. As can be seen, we reach:

– a better Bit-Slice Gate/Gate Equivalent Complexities, differential uniformity and nonlinearity than the S-Boxes $\tilde{\pi}_{\mathsf{Kuz}}$ and $\pi_{\mathsf{Kal}}$, when using the construction of $\pi'_{h_1,h_2,\mathcal{P}_d}$ to get almost optimal permutations;

– a better Bit-Slice Gate/Gate Equivalent Complexities and differential uniformity than permutations $\tilde{\pi}_{\mathsf{Kuz}}$ and $\pi_{\mathsf{Kal}}$ by employing the construction of $\hat{\pi}_{\psi,h}$ to synthesize almost optimal permutations;

– a better set of basic cryptographic parameters than those cryptographic parameters offered by the S-Box $\pi_{\mathsf{Whp}}$ for a number of extra operations `XOR`, `AND`, `OR` and `NOT`, when constructing some almost optimal permutations derived from $\pi'_{h_1,h_2,\mathcal{P}_d}$ and $\hat{\pi}_{\psi,h}$;

– a better (graph) algebraic immunity than the `AES` S-Box for a relatively small number of extra operations `XOR`, `AND`, `OR` and `NOT` and a slight deterioration of the nonlinearity and differential uniformity. However, there is an almost optimal permutation $\dot{\pi}_{\lambda,\tau}$ with the best value of Gate Equivalent Complexity among remaining S-Boxes displayed in this Table.

# 6 Conclusions

In this work, by combining analytical methods with a open source software provided in [10], we have considered the problem of finding low gate count logic circuit representations for some 8-bits instances belonging to certain classes of nonlinear bijective transformations proposed in [11, 12, 13] and the permutation used in the Russian cryptographic standard GOST R 34.12-2015 described in [14]. In particular:

1. For some 8-bit instances derived from these classes, having almost optimal cryptographic properties, we have found two Bit-Slice representations which requires 149 and 159 Boolean operations respectively. Specifically, from the first class of nonlinear bijective transformation we have found an almost optimal permutation $\dot{\pi}_{\lambda,\tau}$ with the best values of Bit-Slice Gate Complexity (119) and Gate Equivalent Complexities (238.78/250) among all S-Boxes studied in this work;

2. For the (almost optimal) S-Box of the block cipher "Kuznyechik", we obtain a (new) Bit-Slice representation consuming only 179 binary logical operations, 48 Boolean operations less than in previously known one given in [1]. Moreover, we have corrected the number of binary logical operations reported in the paper referenced above;

3. For all Bit-Slice representations we have determined its Bit-Slice Gate/Gate Equivalent Complexities;

4. For all 8-bit S-Boxes contemplated in this paper, we provide their Python implementations (see, Appendix Section) through the logical instructions `AND`, `XOR`, `OR` and `NOT`, allowing to verify the correspondence of the values given in the $\mathsf{Look-Up\ Tables}$ of these permutations with those values derived by the Bit-Slice representations;

5. The found Bit-Slice representations allows to insert the masking countermeasure by using the so-called ISW scheme given in [17].

It's should be pointed out that using appropriate binary linear transformations, for example,

$$\begin{aligned}\mathcal{L}_1(l_3, l_2, l_1, l_0, r_3, r_2, r_1, r_0) &= (r_3, l_1, r_0, l_0, l_2, r_1, l_3, r_2),\\ \mathcal{L}_2(l_3, l_2, l_1, l_0, r_3, r_2, r_1, r_0) &= (l_0, r_0, r_1, l_3, l_1, l_2, r_2, l_3),\end{aligned}$$

instead of those used in the TU-decomposition of Kuznyechik S-Box we can reduce the Bit-Slice Gate Complexity up to a value of 165 logical binary operations, keeping its cryptographic properties (nonlinearity, differential uniformity, algebraic degree, algebraic immunity, number of fixed points). All combinatorial circuits presented in this work have been designed and simulated with the open-source project CircuitVerse [9].

# References


[1] Avraamova O. D., Fomin D. B., Serov V. A., Smirnov A. V., Shokov V. N.,, "A compact Bit-Sliced representation of Kuznyechik S-Box", *Mat. Vopr. Kriptogr.*, **12**:2 (2021), 21–38.

[2] Barreto, P.S.L.M and Rijmen, V., "The Whirlpool hashing function", **13** (November, 2000), 14.

[3] Beierle C. et al., "The SKINNY Family of Block Ciphers and Its Low-Latency Variant MANTIS", *LNCS*, CRYPTO 2016., **9815**, ed. Robshaw M., Katz J., Springer, Berlin, Heidelberg, 2016, 123–153.

[4] Biham, Eli., "A fast new DES implementation in software", *International Workshop on Fast Software Encryption*, LNCS, **1267**, Springer, Berlin, Heidelberg, 1997, 260-272.

[5] Biryukov, A., Perrin L., and Udovenko A., "Reverse engineering the S-Box of streebog, kuznyechik and STRIBOBr1", *LNCS*, EUROCRYPT 2016., **9665**, ed. Marc Fischlin and Jean-Sébastien Coron, Springer, Berlin, Heidelberg, 2016, 372-402.

[6] Borisenko N. P., Vasinev D. A., Khoang Dyk Tkho,, "Method of forming S-blocks with minimum number of logic elements", *Abstract of Invention, RU 2572423 C2*, 2016 (in Russian).

[7] Boyar, Joan and René Peralta, "A new combinational logic minimization technique with applications to cryptology", *Springer*, **13** (2010), 178-189.

[8] Carlet C., *Boolean Functions for Cryptography and Coding Theory*, Cambridge, Cambridge University Press, 2021.

[9] *CircuitVerse simulator*, 2023, `https://circuitverse.org`.

[10] Dansarie, M., "sboxgates: A program for finding low gate count implementations of S-Boxes", *Journal of Open Source Software*, **6(62)** (2021), 2946.

[11] de la Cruz-Jiménez, R. A., "Generation of 8-bit S-Boxes having almost optimal cryptographic properties using smaller 4-bit S-Boxes and finite field multiplication", *International Conference on Cryptology and Information Security in Latin America*, Springer, 2017, 191–206.

[12] de la Cruz-Jiménez, R. A., "On some methods for constructing almost optimal s-boxes and their resilience against side-channel attacks", Cryptology ePrint Archive, Report 2018/618, 2018 `http://eprint.iacr.org/2016/493`.

[13] de la Cruz-Jiménez, R. A., "Constructing 8-bit permutations, 8-bit involutions and 8-bit orthomorphisms with almost optimal cryptographic parameters", *Матем. вопр. криптогр.*, **12**:3 (2021), 89–124.

[14] Dygin D. M., Lavrikov I. V., Marshalko G. B. , Rudskoy V. I., Trifonov D. I., Shishkin V. A., "On a new Russian Encryption Standard", *Mat. Vopr. Kriptogr.*, **6**:2 (2015), 29–34.

[15] Knuth, Donald E, "Fascicle 6: Satisfiability, volume 19 of the art of computer programming", Addison-Wesley, Reading, Mass, 2015.

[16] Kwan, M., "Reducing the Gate Count of Bitslice DES", Cryptology ePrint Archive, Report 2000/051., 2000 `https://eprint.iacr.org/2000/051`.

[17] Ishai, Y., Sahai, A., Wagner, D., "Private Circuits: Securing Hardware against Probing Attacks", *LNCS*, Advances in Cryptology - CRYPTO 2003. CRYPTO 2003, **2729**, ed. Boneh, D, Springer, Berlin, Heidelberg, 2003.

[18] Opirskyy, I., Sovyn, Y., Mykhailova, O., "Heuristic method of finding bitliced-description of derivative cryptographic S-Box", 2022, 104-109.

[19] Zhenzhen B. et al., "Peigen – a Platform for Evaluation, Implementation, and Generation of S-Boxes", **2019**:1 (2019), 330–394.

[20] ПНСТ 799-2022. Информационные технологии. Криптографическая защита информации. Термины и определения.

# A Appendix

Listing 1: Python implementation of $\pi'_{h,\mathcal{I}}$

```python
# ---------------------------------- #
# Python code of the BitSlice
# representation of the S-Box
# \pi'_{h,\mathcal{I}}
# ---------------------------------- #
def tobin(x, n):
    return [(x>>i)&1 for i in reversed(range(0, n))]

def frombin(v):
    y = 0
    for i in range(0, len(v)):
        y = (y << 1) | int(v[i])
    return y

def mult(x3,x2,x1,x0,y3,y2,y1,y0):
    t2, t1, t0 = (x3^x0),(x3^x2),(x2^x1)
    z3=y3&t2 ^ y2&x1 ^ y1&x2 ^ y0&x3
    z2=y3&t1 ^ y2&t2 ^ y1&x1 ^ y0&x2
    z1=y3&t0 ^ y2&t1 ^ y1&t2 ^ y0&x1
    z0=y3&x1 ^ y2&x2 ^ y1&x3 ^ y0&x0
    return z3,z2,z1,z0

def INV(x3,x2,x1,x0):
    t4 = x2 ^ x3;
    t5 = x2 | t4;
    t6 = t5 ^ x1;
    t7 = t6 & x0;
    z2 = t4 ^ t7;
    t9  = x1 ^ x2;
    t10 = t9 | t6;
    t11 = x3 & x0;
    z3  = t10 ^ t11;
    t13 = z3 | t4;
    t14 = t13 ^ t6;
    t15 = t7 | x3;
    z1  = t14 ^ t15;
    t17 = x0 ^ t6;
    t18 = z1 | x1;
    t19 = t18 & x2;
    z0  = t17 ^ t19;
    return z3, z2, z1, z0

def h(x3,x2,x1,x0):
    t4 = x2 & x3;
    t5 = t4 ^ x0;
    t6 = t5 ^ x2;
    t7 = t6 | x1;
    z2 = t5 ^ t7;
    t9  = z2 | x1;
    t10 = t9 & x0;
    t11 = z2 ^ t10;
    t12 = t6 & x1;
    t13 = t12 | x3;
    z1  = t11 ^ t13;
    t15 = t13 ^ t5;
    t16 = z1 ^ x3;
    t17 = t16 | x0;
    z3  = t15 ^ t17;
    t19 = t16 | t15;
    t20 = t19 ^ x1;
    t21 = t4 & x0;
    z0  = t20 ^ t21;
    return z3, z2, z1, z0

def pi(x):
    l3, l2, l1, l0,\
    r3, r2, r1, r0 =  tobin(x^64,8)
    l3, l2, l1, l0,\
    r3, r2, r1, r0 = r0, r2, l3, r3,\
    l2, r1,l0, l1
    t3, t2, t1, t0 = h(l3, l2, l1, l0);
    h1, h2, h3, h4 = mult(l3, l2, l1, l0,\
    r3, r2, r1, r0);
    h1, h2, h3, h4 = INV(h1,h2,h3,h4);
    tt = int(not(r3 | r2 | r1 | r0));
    l0 = (t0 & (tt)) | h4;
    l1 = (t1 & (tt)) | h3;
    l2 = (t2 & (tt)) | h2;
    l3 = (t3 & (tt)) | h1;
    t3, t2, t1, t0 = h(r3, r2, r1, r0);
    r3, r2, r1, r0 = INV(r3, r2, r1, r0);
    h1, h2, h3, h4 = mult(l3, l2, l1, l0,\
    r3, r2, r1, r0);
    tt = int(not(l3 | l2 | l1 | l0));
    r0 = t0 & (tt) | h4;
    r1 = t1 & (tt) | h3;
    r2 = t2 & (tt) | h2;
    r3 = t3 & (tt) | h1;
    return frombin([r3, l2, r0, r1,\
    l3, r2, l0, l1])

if __name__ == "__main__":
    print([pi(x) for x in range(1<<8)])
```

Listing 2: Python implementation of $\hat{\pi}_{\psi,\mathcal{I}}$

```python
# ---------------------------------- #
# Python code of the BitSlice
# representation of the S-Box
# \hat{\pi}_{\psi,\mathcal{I}}
# ---------------------------------- #

def tobin(x, n):
    return [(x>>i)&1 for i in reversed(range(0, n))]

def frombin(v):
    y = 0
    for i in range(0, len(v)):
        y = (y << 1) | int(v[i])
    return y

def mult(x3,x2,x1,x0,y3,y2,y1,y0):
    t2, t1, t0 = (x3^x0),(x3^x2),(x2^x1)
    z3=y3&t2 ^ y2&x1 ^ y1&x2 ^ y0&x3
    z2=y3&t1 ^ y2&t2 ^ y1&x1 ^ y0&x2
    z1=y3&t0 ^ y2&t1 ^ y1&t2 ^ y0&x1
    z0=y3&x1 ^ y2&x2 ^ y1&x3 ^ y0&x0
    return z3,z2,z1,z0

def INV(x3,x2,x1,x0):
    t4 = x2 ^ x3;
    t5 = x2 | t4;
    t6 = t5 ^ x1;
    t7 = t6 & x0;
    z2 = t4 ^ t7;
    t9  = x1 ^ x2;
    t10 = t9 | t6;
    t11 = x3 & x0;
    z3  = t10 ^ t11;
    t13 = z3 | t4;
    t14 = t13 ^ t6;
    t15 = t7 | x3;
    z1  = t14 ^ t15;
    t17 = x0 ^ t6;
    t18 = z1 | x1;
    t19 = t18 & x2;
    z0  = t17 ^ t19;
    return z3, z2, z1, z0

def PSI(x3,x2,x1,x0):
    t4 =  int(not(x0));
    t5 = t4 ^ x1;
    t6 = t5 & x2;
    t7 = t6 & x3;
    z1 = t4 ^ t7;
    t9 = x0 ^ t5;
    t10 = t9 & z1;
    t11 =  int(not(t5));
    t12 = t10 | x3;
    t13 = t11 ^ t12;
    t14 = t13 | x2;
    z0 = t10 ^ t14;
    t16 = t5 | x2;
    t17 = t16 ^ t11;
    t18 = t17 & x0;
    z2 = t14 ^ t18;
    t20 =  int(not(z0));
    t21 = x1 | t18;
    t22 = t21 ^ x2;
    t23 = t22 | x3;
    z3 = t20 ^ t23;
    return z3, z2, z1, z0

def pi(x):
    l3, l2, l1, l0,\
    r3, r2, r1, r0 = tobin(x, 8)
    l3, l2, l1, l0,\
    r3, r2, r1, r0 = l1, r3, r0, l0,\
    r1, l3, r2^1, l2
    h1, h2, h3, h4 = mult(l3, l2, l1, l0,\
    r3, r2, r1, r0)
    h1, h2, h3, h4 = PSI(h1, h2, h3, h4)
    l3, l2, l1, l0 = INV(l3, l2, l1, l0)
    l3, l2, l1, l0 = mult(l3, l2, l1, l0,\
    h1, h2, h3, h4 );
    r3, r2, r1, r0 = mult(r3, r2, r1, r0,\
    h1, h2, h3, h4)
    r3, r2, r1, r0 = INV(r3, r2, r1, r0);
    l3, l2, l1, l0,\
    r3, r2, r1, r0 = r1, l1, l0, r3,\
    l2, r0, r2, l3
    return  frombin([l3, l2, l1, l0,\
    r3, r2, r1, r0])




if __name__ == "__main__":
    print([pi(x) for x in range(1<<8)])
```

Listing 3: Python implementation of $\dot{\pi}_{\lambda,\tau}$

```python
# ---------------------------------- #
# Python code of the BitSlice
# representation of the S-Box
# \dot{\pi}_{\lambda,\tau}
# ---------------------------------- #
def tobin(x, n):
    return [(x>>i)&1 for i in reversed(range(0, n))]

def frombin(v):
    y = 0
    for i in range(0, len(v)):
        y = (y << 1) | int(v[i])
    return y

def mult(x3,x2,x1,x0,y3,y2,y1,y0):
    t2, t1, t0 = (x3^x0),(x3^x2),(x2^x1)
    z3=y3&t2 ^ y2&x1 ^ y1&x2 ^ y0&x3
    z2=y3&t1 ^ y2&t2 ^ y1&x1 ^ y0&x2
    z1=y3&t0 ^ y2&t1 ^ y1&t2 ^ y0&x1
    z0=y3&x1 ^ y2&x2 ^ y1&x3 ^ y0&x0
    return z3,z2,z1,z0

def tau(x3,x2,x1,x0):
    t4 = x2 | x3;
    t5 = t4 ^ x1;
    t6 = t5 & x0;
    t7 = x3 ^ t6;
    z1 = t7 ^ x2;
    t9 = t4 ^ x2;
    t10 = t9 | t5;
    t11 = x3 & x0;
    z3 = t10 ^ t11;
    t13 = z1 & z3;
    t14 = t13 ^ t9;
    z2 = t14 ^ x1;
    t16 = z2 | t11;
    t17 = t16 ^ x0;
    t18 = z1 & x2;
    z0 = t17 ^ t18;
    return z3, z2, z1, z0

def pi(x):
    l3, l2, l1, l0,\
    r3, r2, r1, r0 =  tobin(x^64,8)
    l3, l2, l1, l0,\
    r3, r2, r1, r0 = r1, r3, r0, l0,\
    l3, r2, l1, l2
    l3, l2, l1, l0 = tau(l3, l2, l1, l0)
    h1, h2, h3, h4 = l2, l3, l1, l0
    m1, m2, m3, m4 = mult(l3, l2, l1, l0,r3, r2, r1, r0);
    tt = int(not(r3 | r2 | r1 | r0));
    l0 = (h4 & (tt)) | m4;
    l1 = (h3 & (tt)) | m3;
    l2 = (h2 & (tt)) | m2;
    l3 = (h1 & (tt)) | m1;
    h1, h2, h3, h4 = r2, r3, r1, r0;
    m1, m2, m3, m4 = mult(l3, l2, l1, l0,\
    r3, r2, r1, r0);
    tt = int(not(l3 | l2 | l1 | l0));
    r0 = h4 & (tt) | m4;
    r1 = h3 & (tt) | m3;
    r2 = h2 & (tt) | m2;
    r3 = h1 & (tt) | m1;
    r3, r2, r1, r0 = tau(r3, r2, r1, r0);
    return frombin([l0, l1, l2, r2,\
                    r1, r3, r0, l3])

if __name__ == "__main__":
    print([pi(x) for x in range(1<<8)])
```

Listing 4: Python implementation of $\tilde{\pi}_{\mathsf{Kuz}}$

```python
# ---------------------------------------- #
# Python code of the BitSlice
# representation of the Kuznyechik S-Box
# \tilde{\pi}_\mathsf{Kuz}
# ---------------------------------------- #
def tobin(x, n):
    return [(x>>i)&1 for i in reversed(range(0,n))]

def frombin(v):
    y = 0
    for i in range(0, len(v)):
        y = (y << 1) | int(v[i])
    return y

def mult(x3, x2, x1, x0,y3, y2, y1, y0):
    t1 = (x3 ^ x2)
    t2 = (t1 ^ x1)
    z3 = y3&(t2^x0) ^ y2&t2 ^ y1&t1 ^ y0&x3
    z2 = y3&x3 ^ y2&x0 ^ y1&x1^y0&x2
    z1 = y3&t1 ^ y2&x3 ^ y1&x0 ^ y0&x1
    z0 = y3&t2 ^ y2&t1 ^ y1&x3 ^ y0&x0
    return z3, z2, z1, z0

def L1(x7, x6, x5, x4,x3, x2, x1, x0 ):
    z7 = x3
    z6 = x6 ^ x0
    z5 = z6 ^ x1
    t11= x3 ^ x1
    z3 = x7 ^ t11
    z4 = x5 ^ x2 ^ z6 ^ z3
    z2 = x6 ^ x2
    z1 = x4 ^ t11
    z0 = x5
    return z7, z6, z5, z4,z3, z2, z1, z0

def L2(x7, x6, x5, x4,x3, x2, x1, x0):
    z7 = x1 ^ x3
    z6 = x2
    z5 = x5
    z4 = z7 ^ x4 ^ x7
    z3 = x3
    z2 = x2 ^ x6
    z1 = x1 ^ x7
    z0 = x0
    return  z7, z6, z5,z4,z3, z2, z1, z0

def INV(x3, x2, x1, x0):
    t4  = x2 | x0
    t5  = t4 ^ x3
    t6  = t5 | x1
    t7  = x2 ^ t6
    t8  = t5 | x3
    z2  = t7 ^ t8
    t10 = t5  | t7
    t11 = t10 & x1
    t12 = z2  ^ x1
    t13 = t12 ^ t6
    t14 = t13 | x0
    z0  = t11 ^ t14
    t16 = t10 & x0
    t17 = t16 ^ t4
    t18 = x2  | x1
    z3  = t17 ^ t18
    t20 = t13 | x3
    t21 = t20 ^ t11
    t22 = t12 | x0
    z1  = t21 ^ t22
    return z3, z2, z1, z0

def NU_0(x3, x2, x1, x0):
    t4  = x0 | x3
    t5  = t4 & x2
    t6  = x3 ^ x2
    t7  = t6 ^ x0
    t8  = t7 & x1
    z3  = t5 ^ t8
    t10 = int(not(t4))
    t11 = x3 &  t6
    t12 = t11 ^ x0
    t13 = t12 & x1
    z1  = t10 ^ t13
    t15 = t10 ^ t6
    t16 = t15 & x1
    t17 = t12 & x0
    z0  = t16 ^ t17
    t19 = t6  | t13
    t20 = t19 ^ t12
    t21 = x3  & x1
    z2  = t20 ^ t21
    return z3, z2, z1, z0
```

```
91
92  def NU_1(x3, x2, x1, x0):
93      t4 = int(not(x3))
94      t5 = x2 | x1
95      t6 = t4 ^ t5
96      z0 = t6 ^ x0
97      t8 = x2 & z0
98      z3 = t8 ^ x1
99      t10 = z3 & t8
100     z1 = t10 ^ t6
101     t12 = z0 & z3
102     t13 = t12 ^ x2
103     t14 = t4 | x3
104     z2 = t13 ^ t14
105     return z3, z2, z1, z0
106
107 def SIGMA(x3, x2, x1, x0):
108     t4 = x3 ^ x1
109     t5 = t4 | x0
110     t6 = x3 & x2
111     t7 = t5 ^ t6
112     t8 = t6 | x1
113     z0 = t7 ^ t8
114     t10 = t4  ^ x0
115     t11 = t10 & x3
116     t12 = t7  & x2
117     z1 = t11  ^ t12
118     t14 = z1 | t8
119     t15 = t14 ^ t12
120     t16 = int(not(t6))
121     t17 = t16 | x0
122     z3 =  t15 ^ t17
123     t19 = x2  | t4
124     t20 = t19 ^ t17
125     t21 = x1  & x0
126     z2 = t20  ^ t21
127     return z3, z2, z1, z0
128
129 def PHI(x3, x2, x1, x0):
130     t4  = x1 ^ x3
131     t5  = t4 & x2
132     t6  = t5 | x0
133     t7  = x2 ^ t6
134     t8  = t7 | x3
135     z2  = x0 ^ t8
136     t10 = int(not(t4))
137     t11 = t10 | x0
138     t12 = t6 ^ t11
139     z3  = t12 ^ x1
140     t14 = x1  | t10
141     t15 = t14 ^ t8
142     t16 = t15 | x2
143     z0  = z2  ^ t16
144     t18 = t10 | t8
145     t19 = t18 ^ x1
146     t20 = t14 & x2
147     z1  = t19  ^ t20
148     return z3, z2, z1, z0
149
150 def pi(x):
151     l3, l2, l1, l0,\
152     r3, r2, r1, r0 = tobin(x, 8)
153     l3, l2, l1, l0,\
154     r3, r2, r1, r0 = L1(l3, l2, l1, l0,\
155     r3, r2, r1, r0)
156     t1, t2, t3, t4 = INV(r3, r2, r1, r0)
157     h1, h2, h3, h4 = mult(l3, l2, l1, l0,\
158     t1, t2, t3, t4)
159     h1, h2, h3, h4 = NU_1(h1, h2, h3, h4)
160     l3, l2, l1, l0 = NU_0(l3, l2, l1, l0)
161     ind = int(not(r3 | r2 | r1 | r0))
162     l3  = (ind &  l3)       ^ h1
163     l2  = (ind & (l2 ^ 1))  ^ h2
164     l1  = (ind & (l1 ^ 1))  ^ h3
165     l0  = (ind & (l0 ^ 1))  ^ h4
166     t1, t2, t3, t4 = PHI(l3, l2, l1, l0)
167     r3, r2, r1, r0 = mult(r3, r2, r1, r0,\
168     t1, t2, t3, t4)
169     r3, r2, r1, r0 = SIGMA(r3, r2, r1, r0)
170     l3, l2, l1, l0,\
171     r3, r2, r1, r0 = L2(l3, l2, l1, l0,\
172     r3, r2, r1, r0)
173     return frombin([l3, l2, l1, l0, \
174     r3, r2, r1, r0])
175
176
177
178
179
180
181 if __name__ == "__main__":
182     print([pi(x) for x in range(1<<8)])
```

# Computational work for some TU-based permutations

Denis Fomin[1] and Dmitry Trifonov[2]

[1]HSE University, Russia
[2]Academy of Cryptography of Russian Federation, Russia
dfomin@hse.ru, adevlampov@yandex.ru

**Abstract**

In this paper we consider the problem of estimating the computational complexity of some classes of permutations that have $TU$-representation. The combinatorial complexity and circuit depth of the function defining the permutation are used as a metric. To obtain these estimates the representation of field elements in different bases were considered: polynomial, normal, mixed as well as the representation of field elements in PRR and RRB bases.

**Keywords:** permutation, combinational complexity, circuit depth, butterfly, $TU$-decomposition

## Introduction

Computational complexity is one of the most important problems of modern computing science. With the increasing amount of data being processed, the widespread introduction of IoT and M2M technologies, the task of building highly efficient and secure systems is becoming more important than ever. Evaluating computational complexity is an important task from a practical and theoretical point of view.

At present, approaches to the construction of strong cryptographic algorithms have been formed, but the issues of complexity of their implementation, as a rule, remain unexamined. One of the main synthesis blocks of modern cryptographic algorithms is nonlinear bijective transformations. Due to the limitations of modern computing devices, in addition to the natural requirements related to their cryptographic characteristics, additional requirements related to the complexity of their computation are imposed. In this paper, the basic concepts of computational complexity will be given, a review of methods for constructing nonlinear bijective transformations with low complexity will also be given, as approaches to minimize the complexity of calculations of permutations of a special kind.

# 1 Ways to construct low-resource nonlinear bijective transformations with given cryptographic properties

The most commonly used approaches for evaluating the low resourceness of nonlinear transformations are — nonlinear complexity, gate- complexity, bitslice-complexity, combinational complexity. Currently, there are three main approaches to construct permutations with given performance characteristics: full search using graph traversal in depth and using the meet in the middle method ([1, 2, 3]), heuristic search ([4, 5, 6, 3]) and the use of monomial permutations (in particular, permutations of the conversion of non-zero field elements), [7]. However, practically all of the considered approaches allow us to estimate only the number of operations. In order to estimate the real physical implementation labor intensity it is proposed either to implement nodes on real physical devices [8] or to use the knowledge about the implementation labor intensity of each basis function for heuristic search of the optimal implementation [3].

Currently, the fundamental monograph in this area can be considered the work of John E. Savage [9]. The complexity metrics he introduced allow us to evaluate the efficiency of implementation on different platforms, while maintaining mathematical rigor.

Consider measures of complexity related to size and depth logic circuits. They serve as measures of the complexity of functions $f\colon \mathbb{F}_2^n \to \mathbb{F}_2^m$.

A logic circuit, which, according to [9], we will also call a combinational machine, is a connection of elements of some basis $\Omega$, each of which realizes some logical (Boolean) function of the specified basis. The logical scheme can be represented as an oriented graph, whereby the vertices that have a half-degree of approach equal to zero denote the arguments of the function $f$, and the vertices that have a half-degree of outcome equal to zero denote the value of the function $f$.

**Definition 1.** *The combinational complexity of a function $f$ in the basis $\Omega$, denoted by $C_\Omega(f)$, is the minimal number of elements of the basis $\Omega$ sufficient for realization of the function $f$ by a logic circuit.*

**Definition 2.** *The circuit depth (or just depth) of the function $f$ in the basis $\Omega$, denoted by $D_{\Omega(f)}$, s the number of logical elements located on the longest oriented path of the graph representing the logical scheme.*

It is often of interest to find a minimum size scheme for a given function $f$ i.e. the problem of calculating its combinatorial complexity. The Carnot map method and its generalization, the Kwine-McCluskey algorithm [10], are well

known for this purpose. Application of this method actually allows to minimize the size of circuits in case of realization of functions by formulas having a form of sum of products. This representation is called a representation in disjunctive normal form. O.B. Lupanov shows [11], that a simple addition function modulo 2 (function of parity is equal to one if there is an odd number of ones among its arguments $x_1, \ldots, x_n$, где $x_i \in \{0, 1\}$, $i = 1, \ldots, n$) is implemented under the above constraints by an exponential (with respect to $n$) scheme of size, whereas in the absence of constraints it is possible to realize schemes of linear size. Similar results are true for some other functions and for other kinds of normal forms.

**Remark 1.** *In this paper we will use the following basis:* $\Omega = \{\wedge, \vee, \oplus, \neg\}$.

## 2 Field elements representation

Let $\mathbb{F}_{2^n}$ be a finite field of $2^n$ elements. The field $\mathbb{F}_{2^n}$ has a subfield $\mathbb{F}_2$, чwhich allows us to consider the field $\mathbb{F}_{2^n}$ as a vector space over the field $\mathbb{F}_2$, having the basis $\alpha_1, \alpha_2, \ldots, \alpha_n$. Thus, for any field $\mathbb{F}_{2^n}$ by fixing in some way the basis each element of the field $\mathbb{F}_{2^n}$ $\mathbb{F}_{2^n}$ naturally can be represented as a vector of field elements $\mathbb{F}_2$ of length $n$, which allows one to define a bijective mapping $\sigma\colon \mathbb{F}_{2^n} \mapsto V_n$ from the set of field elements into the set of vectors.

For arbitrary $k\colon k|n$ in the field $\mathbb{F}_{2^n}$ there exists a subfield of $2^k$ elements, which we denote by $\mathbb{F}_{2^k}$. Thus, there exists an irreducible polynomial $f(x)$ of degree $m = n/k$ over $\mathbb{F}_{2^k}$ such that $\mathbb{F}_{2^n} \cong \mathbb{F}_{2^k}[x]/f(x)$ and $[x]_{f(x)}$ is the root of the polynomial $f(x)$ in the field $\mathbb{F}_{2^k}[x]/f(x)$.

**Definition 3.** *Let* $\alpha$ *be an element of the field* $\mathbb{F}_{2^n}$ *such that the set* $\left\{\alpha^i\right\}_{i=0}^{m-1}$ *is the basis of* $\mathbb{F}_{2^n}$ *over* $\mathbb{F}_{2^k}$*. In this case it is said that* $\left\{\alpha^i\right\}_{i=0}^{m-1}$ *is the polynomial basis of the field* $\mathbb{F}_{2^n}$ *over* $\mathbb{F}_{2^k}$ *and* $\alpha$ *is called the generator of the polynomial basis.*

Hereinafter a polynomial basis is denoted by Poly. Obviously, that $\left\{[x]^i_{f(x)}\right\}_{i=0}^{m-1}$ is a polynomial basis of $\mathbb{F}_{2^k}[x]/f(x)$ over $\mathbb{F}_{2^k}$. If $\alpha$ is a root of an irreducible polynomial over $\mathbb{F}_{2^k}$ then all the roots in the field $\mathbb{F}_{2^n}$ are elements of the following set: $\left\{\alpha^{2^i}\right\}_{i=0}^{m-1}$. The elements of the set are linearly independent over $\mathbb{F}_{2^k}$.

**Definition 4.** *Let* $\alpha$ *be an element of the field* $\mathbb{F}_{2^n}$ *such that the set* $\left\{\alpha^{2^i}\right\}_{i=0}^{m-1}$ *is the basis of* $\mathbb{F}_{2^n}$ *over* $\mathbb{F}_{2^k}$*. In this case it is said that* $\left\{\alpha^{2^i}\right\}_{i=0}^{m-1}$ *is the normal*

*basis of the field* $\mathbb{F}_{2^n}$ *over* $\mathbb{F}_{2^k}$ *and* $\alpha$ *is called the generator of the normal basis.*

Hereinafter a normal basis is denoted by Norm. Obviously, there exists at least one normal basis for the field $\mathbb{F}_{2^n}$ and the root $\alpha$ of an irreducible polynomial $f(x)$ in the field $\mathbb{F}_{2^n}$ is a generator of both polynomial and normal bases.

Speaking about a permutation on the set of field elements, we will imply that this permutation acts on the vector representation of the field elements.

It is known that polynomial and normal bases are effective for realization of multiplication and Frobenius endomorphism, respectively. Meanwhile, at present the most effective way to reduce combinatorial complexity and depth of functions implementing field operations is to use the field tower theorem and to implement operations in a subfield. For example, in [12] it is proposed that the elements of the field $\mathbb{F}_{2^8}$be represented as a vector $\mathbb{F}_{2^4}^2$,and the elements of the field $\mathbb{F}_{2^4}$ consider as a vector $\mathbb{F}_{2^2}^2$, etc. Thus, the elements of the field $\mathbb{F}_{2^8}$ are represented by vectors from the set $\left(\left(\mathbb{F}_2^2\right)^2\right)^2$.

However, in addition to the representations of field elements described above, there are other ways of representation that allow to achieve low values of combinational complexity and depth of the scheme of functional elements that implement operations in the field. In [13] it is proposed that set the elements of the field $\mathbb{F}_{2^n}$ using $m \geq n$ bits as follows. Let $f(x)$ be an irreducible polynomial of degree $n$ over the field $\mathbb{F}_2$. Let a polynomial $g(x)$ of degree $m-n$ over the field $\mathbb{F}_2$ such that $\mathrm{GCD}(f(x), g(x)) = 1$ and $g(0) \neq 0$. Then consider a polynomial $p(x) = g(x) \cdot f(x)$ of degree $m$.

We'll be set elements of the field $\mathbb{F}_2^n$ by polynomials of degree not exceeding $m$ so that they form a subspace of dimension $n$ in a vector space of dimension $m$. In fact, a CRC $(m, n)$-code is set. Elements of the field $\mathbb{F}_{2^n}$ are uniquely defined by elements of the factor ring of polynomials modulo $p(x)$ which are divided by the polynomial $g(x)$ without remainder. Note that the multiplication operation of such polynomials is correctly defined and defined through the multiplication modulo of the polynomial $p(x)$. In literature such a representation is called Polynomial Ring Representation or PRR, [13].

According to [14], the complexity of operations in the field is usually the smaller the smaller the coefficients in the record of the irreducible polynomial defining it. Thus, in the case where $p(x) = x^{n+1} + 1$, the complexity operations is potentially reduced. Obviously the polynomial $x^{n+1}+1$ can not be irreducible, which makes it impossible to specify with the such a polynomial. However, it can be used to implement PRR-representation. In [15] the

following case is proposed:

$$x^{n+1}+1=(x+1)\cdot\left(x^n+x^{n-1}+\ldots+1\right),$$

where polynomial $g(x)=x+1$, $f(x)=x^n+x^{n-1}+\ldots+1$. Using an irreducible polynomial $f(x)$ of this form allows us to efficiently implement the multiplication operation in the field, and the operation of raising to an arbitrary even degree, which is simply a permutation of the coefficients of the basis vectors [13].

Note that in the PRR-representation each element of the field is uniquely represented by one element of the factor ring $\mathbb{F}_2[x]/p(x)$. However, using all elements of the factor ring $\mathbb{F}_2[x]/p(x)$, one can define elements of the ring. In this case, one field element can be represented by several elements of the specified factor ring. Indeed, if $t_1,t_2\in\mathbb{F}_2[x]/p(x)$, $t_1(x)=t_2(x) \mod f(x)$ then $t_1$ and $t_2$ represent the same field element $\mathbb{F}_{2^n}=\mathbb{F}_2[x]/f(x)$. When $p(x)=x^{n+1}+1=(x+1)\left(x^n+x^{n-1}+\ldots+1\right)$, this representation is called the RRB representation, [16]. Using such a representation reduces the depth of the multiplication operation of field elements [15].

**Remark 2.** *In this paper we will evaluate the combinatorial complexity and depth of functions realizing some functions over the field. Since the type of function directly depends on the basis and the field over which the transformation is considered, let us introduce additional notations:* $C_\Omega\left(f;\mathbb{FF};\text{Basis}\right)$ *and* $D_\Omega\left(f;\mathbb{FF};\text{Basis}\right)$ *will denote the combinational complexity and depth of the function* $f$*, defined over the field* $\mathbb{FF}$ *in the basis* Basis.

## 3 Some classes of permutations

In [17] new classes of permutations of the space $\mathbb{F}_2^8$ were introduced. They have $TU$-representation and rather good cryptographic characteristics. Later, in the papers [18, 19, 20] these permutations were generalized and parametric families of nonlinear bijective transformations were proposed, and their cryptographic characteristics were estimated.

**Definition 5.** *Let* $x_1,x_2\in\mathbb{F}_2^m$, $\pi_i,\widehat{\pi}_i\in S\left(\mathbb{F}_2^m\right)$, $\pi_i(0)=0$, $\widehat{\pi}_i(0)=0$, $i\in\overline{1,2}$*, then a permutation* $F_A$*, defined as follows*

$$y_1=\begin{cases}\pi_1\left(x_1\right)\cdot x_2, & x_2\neq 0\\ \widehat{\pi}_1\left(x_1\right), & x_2=0\end{cases},$$

$$y_2=\begin{cases}\pi_2\left(\left(x_2\right)^2\cdot\pi_1\left(x_1\right)\right), & x_1\neq 0\\ \widehat{\pi}_2\left(x_2\right), & x_1=0\end{cases}.$$

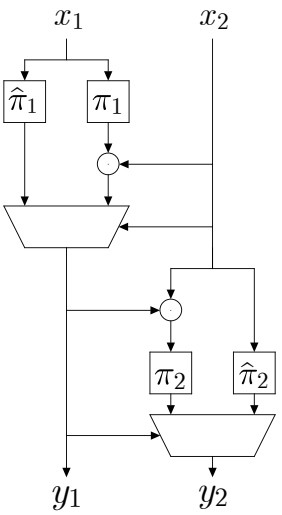



Figure 1: Type "A" permutation

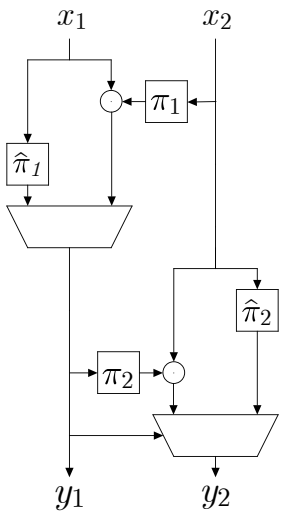



Figure 2: Type "B" permutation

*is called a permutation from a parametric family of type "A" or simply type "A" permutation (fig. 1).*

**Definition 6.** *Let* $x_1, x_2 \in \mathbb{F}_2^m$, $\pi_i, \widehat{\pi}_i \in S\left(\mathbb{F}_2^m\right)$, $\pi_i(0) = 0$, $\widehat{\pi}_i(0) = 0$, $i \in \overline{1,2}$, *then a permutation* $F_B$, *defined as follows*

$$y_1 = \begin{cases} x_1 \cdot \pi_1\left(x_2\right), & x_2 \neq 0 \\ \widehat{\pi}_1\left(x_1\right), & x_2 = 0 \end{cases},$$

$$y_2 = \begin{cases} x_2 \cdot \pi_2\left(x_1 \cdot \pi_1\left(x_2\right)\right), & x_1 \neq 0 \\ \widehat{\pi}_2\left(x_2\right), & x_1 = 0 \end{cases},$$

*is called a permutation from a parametric family of type "B" or simply type "B" permutation (fig. 2).*

Consider a family of permutations, the parameters of which are four degrees: $(\alpha, \beta, \gamma, \delta)$ and permutations $\widehat{\pi}_i$, $i \in \overline{1,2}$:

$$\begin{aligned} G_1\left(x_1, x_2\right) = y_1 &= \begin{cases} x_1^{\alpha} \cdot x_2^{\beta}, & x_2 \neq 0 \\ \widehat{\pi}_1\left(x_1\right), & x_2 = 0 \end{cases}, \\ G_2\left(x_1, x_2\right) = y_2 &= \begin{cases} x_1^{\gamma} \cdot x_2^{\delta}, & x_1 \neq 0 \\ \widehat{\pi}_2\left(x_2\right), & x_1 = 0 \end{cases}. \end{aligned} \tag{1}$$

For equation (1) to specify a bijective transformation, it is sufficient that the equation

$$\begin{cases} G_1\left(x_1, x_2\right) = a_1 \\ G_2\left(x_1, x_2\right) = a_2 \end{cases}$$

has a solution for arbitrary $a_1, a_2 \in \mathbb{F}_2^m$. Such a family of permutations is called a permutation from a parametric family of type "G" or simply type "G" permutation. Obviously, in the case when monomial permutations are chosen as parameters in parametric families of type "A" and type "B", they can be represented by permutations from the parametric family of type "G". It was shown in [20] that such a permutation also has a $TU$–representation. Such a representation o type "A" and type "B" permutations potentially reduces the depth of the function implementing the permutation. Moreover, among all known permutations from the proposed families, the following permutation $G(x_1, x_2) = (y_1, y_2)$ from the parametric family «G» has the minimum number of nonlinear transformations used and is set as follows:

1. $x' = x_1^{-1}$
2. $y' = x_2^{-1}$
3. $x'' = x_1 \cdot y'$
4. $y'' = x' \cdot y'$
5. `if` $x = 0$ `then` $y_1 = y'$ `else` $y_1 = y''$
6. `if` $x = y$ `then` $y_2 = x'$ `else` $y_2 = x''$

The results of [21] show the efficiency of the implementation of the above defined nonlinear bijective transformations on hardware platforms. There is a question about the combinatorial complexity and depth of function for the implementation of permutations from the presented families.

The following functions are used to specify the permutations: multiplication in the field; multiplexer (conditional selection); permutations.

Thus, for parametric families of type "A" and type "B" in [17, 19] monomial permutations $\pi_1, \pi_2$ are considered as parametric permutations. In this work we also consider monomial permutations as the specified parameters.

Permutations $\widehat{\pi}_1, \widehat{\pi}_2$ in the parametric families of types "A", "B" and "G" in [22] is proposed to choose using a heuristic search.

**Remark 3.** *In this paper an experimental study of affine equivalence classes of $\widehat{\pi}_1, \widehat{\pi}_2$ for the considered parametric families was carried out in the case of construction of the permutation of space $\mathbb{F}_2^8$. This can be done since there is a complete classification of permutations [23] for the space $\mathbb{F}_2^4$. Experimental results have shown that in the vast majority of cases the mentioned permutations belong to only two families of $\mathbb{F}_2^4$ permutations with representatives $x^{14}$ and $x^7 + x^4 + x$. Thus, it is of interest to find the complexity of the mentioned permutations.*

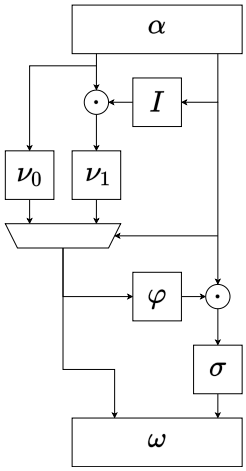



Figure 3: Kuznyechik permutation representation, [24]

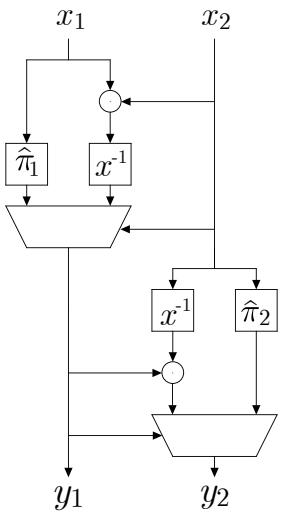



Figure 4: Permutation, defined in [25]

**Remark 4.** *In addition to the permutation considered in this paper, the approach outlined below is also applicable to a number of permutations, such as for Kuznyechik permutation, [24] (fig. 3), and the permutation from the paper [25] (fig. 4).*

# 4 Computational work estimation for some permutations

## 4.1 Computational work fore some functions over the field $\mathbb{F}_{2^2}$ in the normal basis

The field $\mathbb{F}_{2^2}$ is given by a single irreducible polynomial of degree 2: $x^2+x+1$ over the field $\mathbb{F}_2$. In order to construct the field $\mathbb{F}_{(2^2)^2}$ it is necessary to choose an irreducible polynomial of degree 2 over the field $\mathbb{F}_{2^2}$.

It is known that the polynomial $x^2+x+\epsilon$ is irreducible over the field $\mathbb{F}_{2^n}$ if and only if $\mathrm{tr}(\epsilon) = 1$. Also note that an arbitrary polynomial $ax^2+bx+c$ can be reduced to the above form by considering the following transformation: $(a/b^2)f(bx/a)$. Thus, hereinafter we consider only polynomials of this kind.

The elements of the field $\mathbb{F}_2$ will be denoted by the symbols $a, b, c$, the elements of the field $\mathbb{F}_{2^2}$ — $\mathrm{a}, \mathrm{b}, \mathrm{c}$, the basis vectors — by small Greek letters $\alpha, \beta, \gamma$, the elements of the field $\mathbb{F}_{(2^2)^2}$ will be denoted by $\mathbf{a}, \mathbf{b}, \mathbf{c}$, the basis vectors by $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}$.

In [26] it is proposed to consider the following way of realization of arithmetic operations in the field $\mathbb{F}_{2^2}$. Let $e(x) = x^2 + x + 1$, the root of which is the element $\alpha$. Then $\alpha^2 + \alpha = 1$ and $\alpha^3 = 1$, $\alpha^3 = \alpha^2 + \alpha$. Consider the

normal basis $\{\alpha, \alpha^2\}$. The following results directly follow from [26].

**Proposition 1.** *Let «·» — be the multiplication operation in the field* $\mathbb{F}_{2^2}$, *then for* $x, y \in \mathbb{F}_{2^2}$ $C_\Omega(x \cdot y; \mathbb{F}_{2^2}; \mathrm{Norm}) \leq 7$, $D_\Omega(x \cdot y; \mathbb{F}_{2^2}; \mathrm{Norm}) \leq 3$.

Indeed, according to [26]: let $\mathrm{a} = a_0\alpha + a_1\alpha^2$, $\mathrm{b} = b_0\alpha + b_1\alpha^2$, $\mathrm{c} = c_0\alpha + c_1\alpha^2$, $\mathrm{a} \cdot \mathrm{b} = \mathrm{c}$, then

$$\mathrm{a} \cdot \mathrm{b} = \left(a_0\alpha + a_1\alpha^2\right)\left(b_0\alpha + b_1\alpha^2\right) =$$
$$= \left((a_0 + a_1)(b_0 + b_1) + a_0b_0\right)\alpha + \left((a_0 + a_1)(b_0 + b_1) + a_1b_1\right)\alpha^2 = c_0\alpha + c_1\alpha^2.$$

Similarly, it is shown that

**Proposition 2.** *Let* $\alpha, \alpha^2$ *elements of the field* $\mathbb{F}_{2^2}$, *defining its normal basis, then for* $x \in \mathbb{F}_{2^2}$ $C_\Omega(x \cdot \alpha; \mathbb{F}_{2^2}; \mathrm{Norm}) = C_\Omega(x \cdot \alpha^2; \mathbb{F}_{2^2}; \mathrm{Norm}) = 1$, $D_\Omega(x \cdot \alpha; \mathbb{F}_{2^2}; \mathrm{Norm}) = D_\Omega(x \cdot \alpha^2; \mathbb{F}_{2^2}; \mathrm{Norm}) = 1$.

The proof obviously follows from [26]:

$$\alpha\mathrm{a} = a_1\alpha + (a_0 + a_1)\alpha^2, \ \alpha^2\mathrm{a} = (a_0 + a_1)\alpha + a_0\alpha^2.$$

**Proposition 3.** *For* $x \in \mathbb{F}_{2^2}$ $C_\Omega(x^2; \mathbb{F}_{2^2}; \mathrm{Norm}) = 0$, $D_\Omega(x^2; \mathbb{F}_{2^2}; \mathrm{Norm}) = 0$.

The correctness of the latter proposition follows from the fact that $x^2$ is a Frobenius endomorphism.

### 4.2 Computational work fore some functions over the field $\mathbb{F}_{(2^2)^2}$ in the polynomial basis

In [27] the field $\mathbb{F}_{(2^2)^2}$ пis proposed to be considered in the polynomial basis $\{1, \boldsymbol{\beta}\}$, which is denoted by Poly. The field $\mathbb{F}_{(2^2)^2}$ is constructed using the irreducible polynomial $g(x) = x^2 + x + \alpha$, where $\alpha$ is defined above and is the basis element of the field $\mathbb{F}_{2^2}$ in the normal basis. Let us give some of the results that follow from this work.

Let $\mathbf{a} = \mathrm{a}_0 + \mathrm{a}_1\boldsymbol{\beta}$, $\mathbf{b} = \mathrm{b}_0 + \mathrm{b}_1\boldsymbol{\beta}$, $\mathrm{a}_i, \mathrm{b}_i \in \mathbb{F}_{2^2}$, $i = \{1, 2\}$.

**Proposition 4.** *Let «·» — be the multiplication operation in the field* $\mathbb{F}_{(2^2)^2}$, *defined by an irreducible polynomial* $g(x) = x^2 + x + \alpha$ *in the polynomial basis* $\{1, \boldsymbol{\beta}\}$. *Then for* $x, y \in \mathbb{F}_{(2^2)^2}$ $C_\Omega\left(x \cdot y; \mathbb{F}_{(2^2)^2}; \mathrm{Poly}\right) \leq 30$, $D_\Omega\left(x \cdot y; \mathbb{F}_{(2^2)^2}; \mathrm{Poly}\right) \leq 5$.

For the proof it is necessary to consider the implementation of the multiplication operation:

$$\mathbf{a} \cdot \mathbf{b} = \left(\mathrm{a}_0 + \mathrm{a}_1\boldsymbol{\beta}\right)\left(\mathrm{b}_0 + \mathrm{b}_1\boldsymbol{\beta}\right) =$$
$$= \left(\mathrm{a}_0\mathrm{b}_0 + \mathrm{a}_1\mathrm{b}_1\alpha\right) + \left(\left(\mathrm{a}_0 + \mathrm{a}_1\right)\left(\mathrm{b}_0 + \mathrm{b}_1\right) + \mathrm{a}_0\mathrm{b}_0\right)\boldsymbol{\beta} = \mathrm{c}_0 + \mathrm{c}_1\boldsymbol{\beta} = \mathbf{c}.$$

**Proposition 5.** *Consider the field* $\mathbb{F}_{(2^2)^2}$*, defined by an irreducible polynomial* $g(x) = x^2 + x + \alpha$ *in the polynomial basis* $\{1, \boldsymbol{\beta}\}$*. Then for* $x \in \mathbb{F}_{(2^2)^2}$ $C_\Omega\left(x^4; \mathbb{F}_{(2^2)^2}; \mathrm{Poly}\right) \leq 2$*,* $D_\Omega\left(x^4; \mathbb{F}_{(2^2)^2}; \mathrm{Poly}\right) = 1$*.*

Indeed, it is a Frobenius endomorphism and is evaluated as follows:

$$\mathbf{a}^4 = (\mathrm{a}_0 + \mathrm{a}_1\boldsymbol{\beta})^4 = \mathrm{a}_0 + \mathrm{a}_1\boldsymbol{\beta}^4 = \mathrm{a}_0 + \mathrm{a}_1(\boldsymbol{\beta} + \alpha^2 + \alpha) = \\ = \mathrm{a}_0 + \mathrm{a}_1(\boldsymbol{\beta} + 1) = (\mathrm{a}_0 + \mathrm{a}_1) + \mathrm{a}_1\boldsymbol{\beta}.$$

Squaring is considering similarly, from which it follows

**Proposition 6.** *Consider the field* $\mathbb{F}_{(2^2)^2}$*, defined by an irreducible polynomial* $g(x) = x^2 + x + \alpha$ *in the polynomial basis* $\{1, \boldsymbol{\beta}\}$*. Then for* $x \in \mathbb{F}_{(2^2)^2}$ $C_\Omega\left(x^2; \mathbb{F}_{(2^2)^2}; \mathrm{Poly}\right) \leq 3$*,* $D_\Omega\left(x^2; \mathbb{F}_{(2^2)^2}; \mathrm{Poly}\right) \leq 2$*.*

The inverse element in the field can be calculated using the algorithm proposed by Itoh-Tsujii in [28]:

$$\mathbf{a}^{-1} = \left(\mathbf{a}\mathbf{a}^4\right)^{-1}\mathbf{a}^4 = \left((\mathrm{a}_0 + \mathrm{a}_1\boldsymbol{\beta})\left(\mathrm{a}_0 + \mathrm{a}_1\boldsymbol{\beta}^4\right)\right)^{-1}\left(\mathrm{a}_0 + \mathrm{a}_1\boldsymbol{\beta}^4\right) = \\ = \left(\mathrm{a}_0(\mathrm{a}_0 + \mathrm{a}_1) + \mathrm{a}_1^2\alpha\right)^{-1}((\mathrm{a}_0 + \mathrm{a}_1) + \mathrm{a}_1\boldsymbol{\beta}) = d_0 + d_1\boldsymbol{\beta} = D.$$

Hence follows:

**Proposition 7.** *Consider the field* $\mathbb{F}_{(2^2)^2}$*, defined by an irreducible polynomial* $g(x) = x^2 + x + \alpha$ *in the polynomial basis* $\{1, \boldsymbol{\beta}\}$*. Then for* $x \in \mathbb{F}_{(2^2)^2}$ $C_\Omega\left(x^{-1}; \mathbb{F}_{(2^2)^2}; \mathrm{Poly}\right) \leq 26$*,* $D_\Omega\left(x^{-1}; \mathbb{F}_{(2^2)^2}; \mathrm{Poly}\right) \leq 8$*.*

Since we limit ourselves to considering only the choice of monomial parameters in the parametric families of type "A" and "B" for constructing the permutations of the space$\mathbb{F}_2^8$, then we find the realization difficulties of all permutations of the form $x^i$, $i = 1, \ldots, 15$. All such permutations fall into two classes: linear $i = \{1, 2, 4, 8\}$ and nonlinear $i = \{7, 11, 13, 14\}$.

First, consider the permutation $x^8$.

**Proposition 8.** *Consider the field* $\mathbb{F}_{(2^2)^2}$*, defined by an irreducible polynomial* $g(x) = x^2 + x + \alpha$ *in the polynomial basis* $\{1, \boldsymbol{\beta}\}$*. Then for* $x \in \mathbb{F}_{(2^2)^2}$ $C_\Omega\left(x^8; \mathbb{F}_{(2^2)^2}; \mathrm{Poly}\right) \leq 3$*,* $D_\Omega\left(x^8; \mathbb{F}_{(2^2)^2}; \mathrm{Poly}\right) \leq 2$*.*

The proof follows from the following equations:

$$\mathbf{a}^8 = \left(\mathbf{a}^4\right)^2 = ((\mathrm{a}_0 + \mathrm{a}_1) + \mathrm{a}_1\boldsymbol{\beta})^2 = (\mathrm{a}_0^2 + \mathrm{a}_1^2) + \mathrm{a}_1^2\boldsymbol{\beta}^2 = \\ = (\mathrm{a}_0^2 + \mathrm{a}_1^2) + \mathrm{a}_1^2(\boldsymbol{\beta} + \alpha) = (\mathrm{a}_0^2 + \mathrm{a}_1^2\alpha^2) + \mathrm{a}_1^2\boldsymbol{\beta}.$$

The permutations $x^{13}$, $x^{11}$ and $x^7$ belong to the same cyclomatic class, whence follows:

**Proposition 9.** *Consider the field $\mathbb{F}_{(2^2)^2}$, defined by an irreducible polynomial $g(x) = x^2 + x + \alpha$ in the polynomial basis $\{1, \boldsymbol{\beta}\}$. Then for $x \in \mathbb{F}_{(2^2)^2}$:*

- $C_\Omega\left(x^{13}; \mathbb{F}_{(2^2)^2}; \mathrm{Poly}\right) \le 29$, $D_\Omega\left(x^{13}; \mathbb{F}_{(2^2)^2}; \mathrm{Poly}\right) \le 8$;
- $C_\Omega\left(x^{11}; \mathbb{F}_{(2^2)^2}; \mathrm{Poly}\right) \le 26$, $D_\Omega\left(x^{11}; \mathbb{F}_{(2^2)^2}; \mathrm{Poly}\right) \le 8$;
- $C_\Omega\left(x^{7}; \mathbb{F}_{(2^2)^2}; \mathrm{Poly}\right) \le 29$, $D_\Omega\left(x^{7}; \mathbb{F}_{(2^2)^2}; \mathrm{Poly}\right) \le 8$.

The proof is done similarly for the three permutations. As an example, consider $x^{13}$:

$$\begin{aligned}\mathbf{a}^{13} = \left(\mathbf{a}^{14}\right)^2 &= \left[\left(\mathrm{a}_0(\mathrm{a}_0 + \mathrm{a}_1) + \mathrm{a}_1^2\alpha\right)^{-1}\left((\mathrm{a}_0 + \mathrm{a}_1) + \mathrm{a}_1\right)\boldsymbol{\beta}\right]^2 = \\ &= \left(\mathrm{a}_0^2(\mathrm{a}_0^2 + \mathrm{a}_1^2) + \mathrm{a}_1\alpha^2\right)^{-1}\left((\mathrm{a}_0^2 + \mathrm{a}_1^2) + \mathrm{a}_1^2\boldsymbol{\beta}^2\right) = \\ &= \left(\mathrm{a}_0^2(\mathrm{a}_0^2 + \mathrm{a}_1^2) + \mathrm{a}_1\alpha^2\right)^{-1}\left((\mathrm{a}_0^2 + \mathrm{a}_1^2) + \mathrm{a}_1^2\left(\boldsymbol{\beta} + \alpha\right)\right) = \\ &= \left(\mathrm{a}_0^2(\mathrm{a}_0^2 + \mathrm{a}_1^2) + \mathrm{a}_1\alpha^2\right)^{-1}\left(\mathrm{a}_0^2 + \mathrm{a}_1^2\alpha^2 + \mathrm{a}_1^2\boldsymbol{\beta}\right) = d_0 + d_1\boldsymbol{\beta} = D.\end{aligned}$$

As stated earlier, we are also interested in the function $x^7 + x^4 + x$. Consider first the function $x^4 + x$:

$$\mathbf{a}^4 + \mathbf{a} = (\mathrm{a}_0 + \mathrm{a}_1) + \mathrm{a}_1\boldsymbol{\beta} + \mathrm{a}_0 + \mathrm{a}_1\boldsymbol{\beta} = \mathrm{a}_1.$$

Then the following is obvious:

**Proposition 10.** *Consider the field $\mathbb{F}_{(2^2)^2}$, defined by an irreducible polynomial $g(x) = x^2 + x + \alpha$ in the polynomial basis $\{1, \boldsymbol{\beta}\}$. Then for $x \in \mathbb{F}_{(2^2)^2}$ $C_\Omega\left(x^7 + x^4 + x; \mathbb{F}_{(2^2)^2}; \mathrm{Poly}\right) \le 31$, $D_\Omega\left(x^7 + x^4 + x; \mathbb{F}_{(2^2)^2}; \mathrm{Poly}\right) \le 9$.*

### 4.3 Computational work fore some functions over the field $\mathbb{F}_{(2^2)^2}$ in the normal and mixed bases

Let the elements of the field $\mathbb{F}_{(2^2)^2}$ be defined in a normal basis. It is known that using the normal basis the Frobenius endomorphism is realized much more efficiently. This allows us to propose the implementation of the non-zero element circulation operation $x^{-1}$, which has much less complexity.

As in [12] consider the normal basis $\{\boldsymbol{\beta}, \boldsymbol{\beta}^4\}$, where $\boldsymbol{\beta}$ is the root of the irreducible over $\mathbb{F}_{2^2}$ polynomial $x^2 + x + \alpha$. Let $\mathbf{a} = \mathrm{a}_0\boldsymbol{\beta} + \mathrm{a}_1\boldsymbol{\beta}^4$ is an

arbitrary nonzero element of the field $\mathbb{F}_{(2^2)^2}$. The inverse element in the field is evaluated as follows:

$$\mathbf{a}^{-1} = \left(\mathbf{a}\mathbf{a}^4\right)^{-1}\mathbf{a}^4 = \left(\left(\mathrm{a}_0\boldsymbol{\beta} + \mathrm{a}_1\boldsymbol{\beta}^4\right)\left(\mathrm{a}_1\boldsymbol{\beta} + \mathrm{a}_0\boldsymbol{\beta}^4\right)\right)^{-1}\left(\mathrm{a}_1\boldsymbol{\beta} + \mathrm{a}_0\boldsymbol{\beta}^4\right) = \\ = \left(\mathrm{a}_0\mathrm{a}_1 + (\mathrm{a}_0 + \mathrm{a}_1)^2\alpha\right)^{-1}\left(\mathrm{a}_1\boldsymbol{\beta} + \mathrm{a}_0\boldsymbol{\beta}^4\right) = d_0\boldsymbol{\beta} + d_1\boldsymbol{\beta}^4 = D.$$

**Proposition 11.** *Consider the field $\mathbb{F}_{(2^2)^2}$, defined by an irreducible polynomial $g(x) = x^2 + x + \alpha$, in the normal basis$\{\boldsymbol{\beta}, \boldsymbol{\beta}^4\}$. Then for $x \in \mathbb{F}_{(2^2)^2}$ $C_\Omega\left(x^{-1}; \mathbb{F}_{(2^2)^2}; \mathrm{Norm}\right) \leq 26$, $D_\Omega\left(x^{-1}; \mathbb{F}_{(2^2)^2}; \mathrm{Norm}\right) \leq 7$.*

Thus, using the normal basis allows you to reduce the depth of the function that implements the calculation of the inverse element in the field by 1.

In [15] it is proposed to consider mixed bases for realization of operations in the field. Using different bases for different operations can lead to reducing the depth of the scheme implementing the permutation. For example, the following formula describes a way to implement the permutation $x^{-1}$ of nonzero elements given in the normal basis so that the result is represented in the polynomial basis:

$$\mathbf{a}^{-1} = \left(\mathbf{a}\mathbf{a}^4\right)^{-1}\mathbf{a}^4 = \left(\left(\mathrm{a}_0\boldsymbol{\beta} + \mathrm{a}_1\boldsymbol{\beta}^4\right)\left(\mathrm{a}_1\boldsymbol{\beta} + \mathrm{a}_0\boldsymbol{\beta}^4\right)\right)^{-1}\left(\mathrm{a}_1\boldsymbol{\beta} + \mathrm{a}_0\boldsymbol{\beta}^4\right) = \\ = \left(\mathrm{a}_0\mathrm{a}_1 + (\mathrm{a}_0 + \mathrm{a}_1)^2\alpha\right)^{-1}\left((\mathrm{a}_1 + \mathrm{a}_0) + \mathrm{a}_0\boldsymbol{\beta}\right) = d_0 + d_1\boldsymbol{\beta} = D.$$

This representation does not increase the complexity of the formula. For mixed bases we will use the notation PtN (Polynomial to Normal) and NtP (Normal to Polynomial) for functions given in one basis, the result of which is represented in another basis. As a polynomial basis everywhere we will consider the basis $\{1, \boldsymbol{\beta}\}$, and as a normal one we consider the basis $\{\boldsymbol{\beta}, \boldsymbol{\beta}^4\}$. Then:

**Proposition 12.** *For $x \in \mathbb{F}_{(2^2)^2}$ $C_\Omega\left(x^{-1}; \mathbb{F}_{(2^2)^2}; \mathrm{NtP}\right) \leq 26$, $D_\Omega\left(x^{-1}; \mathbb{F}_{(2^2)^2}; \mathrm{NtP}\right) \leq 7$.*

Let us find the complexity of all monomial permutations. The increase in the 4th degree is calculated as follows:

$$\mathbf{a}^4 = \left(\mathrm{a}_0\boldsymbol{\beta} + \mathrm{a}_1\boldsymbol{\beta}^4\right)^4 = \mathrm{a}_1\boldsymbol{\beta} + \mathrm{a}_0\boldsymbol{\beta}^4.$$

If one need the result to be represented in a polynomial basis:

$$\mathbf{a}^4 = \left(\mathrm{a}_0\boldsymbol{\beta} + \mathrm{a}_1\boldsymbol{\beta}^4\right)^4 = \mathrm{a}_1\boldsymbol{\beta} + \mathrm{a}_0\boldsymbol{\beta}^4 = \mathrm{a}_1\boldsymbol{\beta} + \mathrm{a}_0(\boldsymbol{\beta} + 1) = \mathrm{a}_0 + (\mathrm{a}_0 + \mathrm{a}_1)\boldsymbol{\beta}.$$

Hence follows:

**Proposition 13.** *For* $x \in \mathbb{F}_{(2^2)^2}$

– $C_\Omega\left(x^4; \mathbb{F}_{(2^2)^2}; \mathrm{Norm}\right) = 0$, $D_\Omega\left(x^4; \mathbb{F}_{(2^2)^2}; \mathrm{Norm}\right) = 0$;

– $C_\Omega\left(x^4; \mathbb{F}_{(2^2)^2}; \mathrm{NtP}\right) \leq 2$, $D_\Omega\left(x^4; \mathbb{F}_{(2^2)^2}; \mathrm{NtP}\right) \leq 1$.

Squaring and the 8th degree are calculated similarly:

**Proposition 14.** *For* $x \in \mathbb{F}_{(2^2)^2}$ $C_\Omega\left(x^2; \mathbb{F}_{(2^2)^2}; \mathrm{Norm}\right) = C_\Omega\left(x^8; \mathbb{F}_{(2^2)^2}; \mathrm{Norm}\right) \leq 4$, $D_\Omega\left(x^2; \mathbb{F}_{(2^2)^2}; \mathrm{Norm}\right) = D_\Omega\left(x^8; \mathbb{F}_{(2^2)^2}; \mathrm{Norm}\right) \leq 2$.

*Proof.* For the proof, note that the calculation $(\mathrm{a}_0^2\alpha^2 + \mathrm{a}_1^2\alpha)$ can be performed in 3 (instead of 4) operations, since the value of the specified function is depend on the intersecting values of the variables. After that the value of $(\mathrm{a}_0^2\alpha + \mathrm{a}_1^2\alpha^2)$ is calculated by multiplying $(\mathrm{a}_0^2\alpha^2 + \mathrm{a}_1^2\alpha)$ by $\alpha^2$, which is possible to produce in 1 operation.

To obtain an estimate of depth, it is necessary to consider the graph in which $(\mathrm{a}_0^2\alpha^2 + \mathrm{a}_1^2\alpha)$ and $(\mathrm{a}_0^2\alpha + \mathrm{a}_1^2\alpha^2)$ are two independent paths. □

Similarly for the mixed basis:

**Proposition 15.** *For* $x \in \mathbb{F}_{(2^2)^2}$ $C_\Omega\left(x^2; \mathbb{F}_{(2^2)^2}; \mathrm{NtP}\right) = C_\Omega\left(x^8; \mathbb{F}_{(2^2)^2}; \mathrm{NtP}\right) \leq 4$, $D_\Omega\left(x^2; \mathbb{F}_{(2^2)^2}; \mathrm{NtP}\right) = D_\Omega\left(x^8; \mathbb{F}_{(2^2)^2}; \mathrm{NtP}\right) \leq 2$.

Thus, using the normal basis to implement linear permutations does not increase the efficiency compared to using the polynomial basis.

Let us estimate the complexity of the implementation of permutations $x^7$, $x^{11}$, $x^{13}$. Consider first, the simplest case:

$$\mathbf{a}^{11} = \left(\mathbf{a}^{-1}\right)^4 = \left(\mathrm{a}_0\mathrm{a}_1 + (\mathrm{a}_0 + \mathrm{a}_1)^2\alpha\right)^{-1}\left(\mathrm{a}_0\boldsymbol{\beta} + \mathrm{a}_1\boldsymbol{\beta}^4\right) = d_0\boldsymbol{\beta} + d_1\boldsymbol{\beta}^4 = D.$$

In the case where the permutation values are represented in a polynomial basis:

$$\begin{aligned}\mathbf{a}^{11} = \left(\mathbf{a}^{-1}\right)^4 &= \left(\mathrm{a}_0\mathrm{a}_1 + (\mathrm{a}_0 + \mathrm{a}_1)^2\alpha\right)^{-1}\left((\mathrm{a}_0 + \mathrm{a}_1) + \mathrm{a}_0(\boldsymbol{\beta} + 1)\right) = \\ &= \left(\mathrm{a}_0\mathrm{a}_1 + (\mathrm{a}_0 + \mathrm{a}_1)^2\alpha\right)^{-1}\left(\mathrm{a}_1 + (\mathrm{a}_0 + \mathrm{a}_1)\boldsymbol{\beta}\right) = d_0 + d_1\boldsymbol{\beta} = D.\end{aligned}$$

**Remark 5.** *The combinational complexity and depth of the function implementing the permutation* $x^{11}$ *in the normal and mixed bases are equal to the corresponding values of the permutation* $x^{14}$.

Consider the permutation $x^{13}$ in the normal and mixed bases.

$$\mathbf{a}^{13} = \left(\mathbf{a}^{-1}\right)^2 = \left(\mathrm{a}_0^2\mathrm{a}_1^2 + (\mathrm{a}_0 + \mathrm{a}_1)\alpha^2\right)^{-1} \left(\mathrm{a}_1^2\left(\alpha^2\boldsymbol{\beta} + \alpha\boldsymbol{\beta}^4\right) + \mathrm{a}_0^2\left(\alpha\boldsymbol{\beta} + \alpha^2\boldsymbol{\beta}^4\right)\right) =$$
$$= \left(\mathrm{a}_0^2\mathrm{a}_1^2 + (\mathrm{a}_0 + \mathrm{a}_1)\alpha^2\right)^{-1} \left(\left(\mathrm{a}_0^2\alpha + \mathrm{a}_1^2\alpha^2\right)\boldsymbol{\beta} + \left(\mathrm{a}_0^2\alpha^2 + \mathrm{a}_1^2\alpha\right)\boldsymbol{\beta}^4\right) =$$
$$= d_0 + d_1\boldsymbol{\beta} = D.$$

In case $x^7 = (x^{13})^4$ the equation is the same up to rearrangement of the coefficient values at the basis vectors.

**Proposition 16.** *For* $x \in \mathbb{F}_{(2^2)^2}$ $C_\Omega\left(x^{13}; \mathbb{F}_{(2^2)^2}; \mathrm{Norm}\right) = C_\Omega\left(x^7; \mathbb{F}_{(2^2)^2}; \mathrm{Norm}\right) \leq 29$, $D_\Omega\left(x^{13}; \mathbb{F}_{(2^2)^2}; \mathrm{Norm}\right) = D_\Omega\left(x^7; \mathbb{F}_{(2^2)^2}; \mathrm{Norm}\right) \leq 7$.

*Proof.* For the proof it is enough to show that the values of $(\mathrm{a}_0 + \mathrm{a}_1)\alpha^2$, $\left(\mathrm{a}_0^2\alpha + \mathrm{a}_1^2\alpha^2\right)$, $\left(\mathrm{a}_0^2\alpha^2 + \mathrm{a}_1^2\alpha\right)$ can be calculated in 6 operations in the basis $\Omega$. □

Consider the case where the value of the permutation is represented in a polynomial basis:

$$\mathbf{a}^{13} = \left(\mathrm{a}_0^2\mathrm{a}_1^2 + (\mathrm{a}_0 + \mathrm{a}_1)\alpha^2\right)^{-1} \left(\left(\mathrm{a}_0^2\alpha + \mathrm{a}_1^2\alpha^2\right)\boldsymbol{\beta} + \left(\mathrm{a}_0^2\alpha^2 + \mathrm{a}_1^2\alpha\right)(\boldsymbol{\beta} + 1)\right) =$$
$$= \left(\mathrm{a}_0^2\mathrm{a}_1^2 + (\mathrm{a}_0 + \mathrm{a}_1)\alpha^2\right)^{-1} \left(\left(\mathrm{a}_0^2\alpha^2 + \mathrm{a}_1^2\alpha\right) + \left(\mathrm{a}_0^2 + \mathrm{a}_1^2\right)\boldsymbol{\beta}\right) =$$
$$= d_0 + d_1\boldsymbol{\beta} = D.$$

**Remark 6.** *he combinational complexity and depth of the function implementing the permutations* $x^{13}$ *and* $x^7$ *n the normal basis are equal to the corresponding values in the mixed basis.*

Here is an equation from [15] that allows us to obtain a value in the normal basis for the multiplication operation in the field of elements represented in the polynomial basis:

$$\mathbf{a} \cdot \mathbf{b} = (\mathrm{a}_0 + \mathrm{a}_1\boldsymbol{\beta})(\mathrm{b}_0 + \mathrm{b}_1\boldsymbol{\beta}) =$$
$$= \left[(\mathrm{a}_0 + \mathrm{a}_1)(\mathrm{b}_0 + \mathrm{b}_1) + \mathrm{a}_1\mathrm{b}_1\alpha\right]\boldsymbol{\beta} + (\mathrm{a}_0\mathrm{b}_0 + \mathrm{a}_1\mathrm{b}_1\alpha)\,\boldsymbol{\beta}^4 = \mathrm{c}_0 + \mathrm{c}_1\boldsymbol{\beta} = \mathbf{c}.$$

**Proposition 17.** *For* $x, y \in \mathbb{F}_{(2^2)^2}$ $C_\Omega\left(x \cdot y; \mathbb{F}_{(2^2)^2}; \mathrm{PtN}\right) \leq 30$, $D_\Omega\left(x \cdot y; \mathbb{F}_{(2^2)^2}; \mathrm{PtN}\right) \leq 5$.

### 4.4 Computational work fore some functions over the field $\mathbb{F}_{(2^2)^2}$ using PRR and RRB representations

Let $f(x) = x^4+x^3+x^2+x+1$ be an irreducible polynomial over the field $\mathbb{F}_2$ and $\beta$ — be its root in the minimal expansion field. Then $\{\boldsymbol{\beta}^0, \boldsymbol{\beta}^1, \boldsymbol{\beta}^2, \boldsymbol{\beta}^3\}$ is a polynomial basis. Using the RRB-representation the field elements are represented as a linear combination of elements of the set $\{\boldsymbol{\beta}^0, \boldsymbol{\beta}^1, \boldsymbol{\beta}^2, \boldsymbol{\beta}^3, \boldsymbol{\beta}^4\}$. Obviously, the field elements are not represented in the only one way.

For this representation the multiplication operation is implemented as follows. Let $\mathbf{a} = a_0 + a_1\boldsymbol{\beta} + \ldots + a_4\boldsymbol{\beta}^4$, $\mathbf{b} = b_0 + b_1\boldsymbol{\beta} + \ldots + b_4\boldsymbol{\beta}^4$. Then $\mathbf{d} = \mathbf{a} \cdot \mathbf{b}$, $\boldsymbol{d} = d_0 + d_1\boldsymbol{\beta} + \ldots + d_4\boldsymbol{\beta}^4$ are calculated by the following (see [15]):

$$
\begin{aligned}
d_0 &= (a_1 + a_3)(b_1 + b_4) + (a_2 + a_3)(b_2 + b_3),\\
d_1 &= (a_0 + a_1)(b_0 + b_1) + (a_2 + a_4)(b_2 + b_4),\\
d_2 &= (a_0 + a_2)(b_0 + b_2) + (a_4 + a_4)(b_3 + b_4),\\
d_3 &= (a_0 + a_3)(b_0 + b_3) + (a_2 + a_2)(b_1 + b_2),\\
d_4 &= (a_0 + a_4)(b_0 + b_4) + (a_3 + a_3)(b_1 + b_3).
\end{aligned}
$$

The combinational complexity of this representation is 35, which is greater than similar values considered for other bases. At the same time, the depth of the function defining such a representation is 3, which is the smallest known value [15].

Using PRR-representation allows to implement some operations more efficiently than in polynomial, normal or mixed bases. For example, in [15] the method of calculating the inverse element in the field is given. Let $\mathbf{a} = a_0 + a_1\boldsymbol{\beta} + \ldots + a_4\boldsymbol{\beta}^4$, $\mathbf{b} = b_0 + b_1\boldsymbol{\beta} + \ldots + b_4\boldsymbol{\beta}^4$, $\mathbf{a}^{-1} = \mathbf{b}$. Then:

$$
\begin{aligned}
b_0 &= (a_1 \vee a_4)(a_2 \vee a_3),\\
b_1 &= ((a_4 + 1)(a_1 + a_2)) \vee (a_0a_4(a_2 \vee a_3)),\\
b_2 &= ((a_3 + 1)(a_2 + a_4)) \vee (a_0a_3(a_1 \vee a_4)),\\
b_3 &= ((a_2 + 1)(a_1 + a_3)) \vee (a_0a_2(a_1 \vee a_4)),\\
b_4 &= ((a_1 + 1)(a_3 + a_4)) \vee (a_0a_1(a_2 \vee a_3)).
\end{aligned}
$$

Such a representation is possible because in PRR representation the arguments of the Boolean function that defines the operation of inversion of non-zero field elements are only vectors that have zero binary weight. This allows you to define other values arbitrarily so as to minimize computational complexity. Moreover, since the value given in the PRR representation is also an

RRB representation, it is possible (as the authors of [15] did for the function above) to abandon the requirement that the permutation value have a PRR representation. For example, if the unit field element is inverted using a PRR representation by the equations above, one get a 1 instead of the polynomial that sets the unit in the PRR representation. It is possible to do this if after the permutation value is evaluated, there is a multiplication in the field.

The combinational complexity of the operation of inversion of non-zero field elements is 31, which is greater than in the case of normal, polynomial, and mixed bases, but the depth of the function is 3.

Note that the expansion of elements of degree 2, 4, 8 has combinational complexity and depth equal to 0. Hence, in particular, it follows that calculating the monomial permutation has complexity and depth equal to similar values of the permutation $x^{14}$.

### 4.5 Computational work for MUX

Let us consider the complexity of the multiplexer implementation, similarly to [6]. According to the definition of the parametric families of types "A", "B" and "G" a calculation similar to the following occurs:
«`if` $x_1 = 0$ `then` $y = \widehat{\pi}(x_0)$ `else` $y = \pi_2(\pi_0(x_0) \cdot \pi_1(x_1))$»,
where $\pi_0, \pi_1, \pi_2, \widehat{\pi}$ are bijective permutations of $\mathbb{F}_2^4$.

Consider the indicator function that takes a value of 1 at the point $x_1 = 0$ and zero at all other points:

$$\mathrm{Ind}_0(x_1) = \overline{x_1^{(1)}} \cdot \overline{x_1^{(2)}} \cdot \overline{x_1^{(3)}} \cdot \overline{x_1^{(4)}} = \overline{x_1^{(1)} \vee x_1^{(2)} \vee x_1^{(3)} \vee x_1^{(4)}},$$

where $x^{(j)}$, $j = \{1, \ldots, 4\}$, is the value standing in the $j$-th cell of the vector $x_1 \in \mathbb{F}_2^4$. The combinational complexity of the indicator function is 4, the depth is 3 (due to the computation of the negation).

Value considered function can be calculated as follows:

$$\mathrm{Ind}_0(x_1) \cdot \widehat{\pi}(x_0) + \overline{\mathrm{Ind}_0(x_1)} \cdot \pi_2(\pi_0(x_0) \cdot \pi_1(x_1)).$$

The calculation of this function can be simplified:

$$\mathrm{Ind}_0(x_1) \cdot (\widehat{\pi}(x_0) + \pi_2(\pi_0(x_0) \cdot \pi_1(0))) + \pi_2(\pi_0(x_0) \cdot \pi_1(x_1)).$$

In the case $\pi_1(0) = 0$, the last expression is simplified:

$$\mathrm{Ind}_0(x_1) \cdot (\widehat{\pi}(x_0) + \pi_2(0)) + \pi_2(\pi_0(x_0) \cdot \pi_1(x_1)).$$

In the case $\pi_2(0) = 0$, the expression takes the following form:

$$\mathrm{Ind}_0(x_1) \cdot \widehat{\pi}(x_0) + \pi_2(\pi_0(x_0) \cdot \pi_1(x_1)).$$

Then:

**Proposition 18.** *The combinational complexity of* $F(x_0, x_1) = \mathrm{Ind}_0(x_1) \cdot \widehat{\pi}(x_0) + \pi_2(\pi_0(x_0) \cdot \pi_1(x_1))$ *is estimated by:*

$$C_\Omega\left(\widehat{\pi}\right) + C_\Omega\left(\pi_2(\pi_0(x_0) \cdot \pi_1(x_1))\right) + 12.$$

*The depth of the function that implements the above formula is equal:*

$$\max\left\{4, D_\Omega\left(\widehat{\pi}\right) + 2, D_\Omega\left(\pi_2(\pi_0(x_0) \cdot \pi_1(x_1))\right) + 1\right\}.$$

**Remark 7.** *When using the PRR representation, the complexity and depth of the formula calculating* $\mathrm{Ind}_0(x_1)$ *will not change, since the zero value in this representation is given by a vector of 5 zeros (see e.g. [15]), while none of the other vectors* $\mathbb{F}_2^5$*, defining elements of the field* $\mathbb{F}_2^4$ *does not have 4 zeros in its entry on any positions.*

*If RRB representation is used, the combinational complexity of calculating* $\mathrm{Ind}_0(x_1)$ *will increase by 1, and the depth will not change. Moreover, when using the permutation preserving 0 (monomial permutations keep 0), it is possible to calculate the* $\mathrm{Ind}_0(x_1)$ *from the input values, which even when using the PRP representation will not change the combinational complexity (relative to the value obtained in the last proposition), and will reduce the depth of the whole scheme.*

*Thus, for the normal, polynomial basis the combinational complexity of calculating y is 12 more than the combinational complexity of calculating the remaining functions, in the case of PRR or RRB representations this value will be 14.*

### 4.6 Computational work for $\widehat{\pi}_i$

As indicated earlier, as a result of experimental studies of the algorithm from [22], the permutations $\widehat{\pi}_i$, $i \in \{1, 2\}$, are overwhelmingly affine-equivalent to permutations with representatives: $x^{14}$, $x^7 + x^4 + x$, whose implementation complexity was estimated earlier.

When implementing affine-equivalent permutations, in addition to implementing formulas implementing $x^{14}$ and $x^7 + x^4 + x$ it is necessary to compute no more than two multiplications on reversible matrices $\mathrm{GL}(4, 2)$ and no more than two additions with vectors of length 4.

Obviously, the depth of the matrix multiplication depends on the maximum weight of the matrix row and is equal to $\lceil \log_2 \mathrm{w}_{\max} \rceil$. This value is obviously always less than or equal to 2.

The combinational complexity can be estimated using the number of 1's in the whole matrix $\mathrm{w}_+$. It can be easily shown that the combinational complexity of multiplication by an arbitrary reversible matrix does not exceed 9.

Thus, the combinational complexity of the calculation of $\widehat{\pi}_i$, $i \in \{1, 2\}$, does not exceed the following value: $C_\Omega(\pi') + 26$, where $\pi' \in \left\{x^{14}, x^7 + x^4 + x\right\}$. The depth of the formula specifying $\widehat{\pi}_i$, $i \in \{1, 2\}$, does not exceed the following value: $D_\Omega(\pi') + 6$, where $\pi' \in \left\{x^{14}, x^7 + x^4 + x\right\}$.

**Remark 8.** *For one permutation $\widehat{\pi}_i$ there can exist more than one pair of matrices and vectors such that*

$$\widehat{\pi}_i(x) = a'_1 + L'_1\left(\pi'\left(L'_2(x) + a'_2\right)\right) = a''_1 + L'_1\left(\pi'\left(L'_2(x) + a''_2\right)\right),$$

*where $\pi' \in \{x^{-1}, x^7 + x^4 + x\}$, which allows one to choose a representation that has less combinational complexity or depth. In this case one representation can minimize the combinational complexity, the other one can minimize the depth.*

## 4.7 Computational work estimation for some parametric families of permutations

Let us consider the complexity of realization of permutations from the parametric family of type "G", which also generalize the parametric families of types "A" and "B" in the case of monomial choice of parameters $\pi_1, \pi_2$.

Let us assume that the input vectors are specified in the basis we need, since multiplication of each of the coordinates $x_1$, $x_2$ by a reversible matrix does not change its equivalence class, and the representation of operations is not specified when specifying them. Let us perform the same reasoning as in [27]. For this purpose, the calculation of all permutations and multiplication operations should be performed in mixed bases, and the calculation of $\widehat{\pi}_i$, $i = 1, 2$ in the normal basis.

To implement a permutation from the parametric family of type "G" it is necessary to implement two functions, each of which consists of three permutations (two of them monomial), an operation of multiplication and a multiplexer. The combinational complexity of such a permutation is estimated by the following value:

$$C_\Omega(x^\alpha)+C_\Omega(x^\beta)+C_\Omega(x^\gamma)+C_\Omega(x^\delta)+2C_\Omega(\cdot)+C_\Omega\left(\widehat{\pi}_1\right)+C_\Omega\left(\widehat{\pi}_2\right)+2C_\Omega(\mathrm{MUX}) \leq$$
$$\leq 4 \cdot C_\Omega(x^7) + 2C_\Omega(\cdot) + 2C_\Omega(x^7 + x^4 + x) + 2 \cdot 30 + 2 \cdot 12 = 322.$$

Moreover, each of the pairs $(x^\alpha, x^\gamma)$, $\left(x^\beta, x^\delta\right)$, contains either one linear substitution, and the pair implementation does not exceed $29 + 3$ operations, or contains two nonlinear permutations, whose formulas largely coincide, and their combinational complexity also does not exceed $29 + 3$ (first we calculate

the nonlinear one, then we power up $2^i$ at some $i$, and again we obtain the other nonlinear one). This fact reduces the maximum value of combinatorial complexity to 270.

This value can be greatly overestimated. Consider the following permutation $S(x_1, x_2) = (y_1, y_2)$ (из [19, 20]):

$$y_1 = \begin{cases} x_1 \cdot x_2^2, & x_2 \neq 0 \\ x_1^{-1}, & x_2 = 0 \end{cases},$$

$$y_2 = \begin{cases} x_1^{-1} \cdot x_2^{-1}, & x_1 \neq 0 \\ x_2^{-1}, & x_1 = 0 \end{cases}.$$

Its combinational complexity in the considered basis obviously does not exceed 147.

Let us estimate the depth of the formula specifying a permutation from a parametric family of type "G":

$$\max\left\{4, D_\Omega\left(\widehat{\pi}_1\right) + 2, D_\Omega\left(\widehat{\pi}_2\right) + 2, D_\Omega\left(x_1^\alpha \cdot x_2^\beta\right) + 1, D_\Omega\left(x_1^\gamma \cdot x_2^\delta\right) + 1\right\} \leq$$
$$\leq \max\left\{4, 8 + 6 + 2, \max\left\{D_\Omega\left(x_1^\alpha\right), D_\Omega\left(x_2^\beta\right), D_\Omega\left(x_1^\gamma\right), D_\Omega\left(x_2^\delta\right)\right\} + 6\right\} \leq$$
$$\leq \max\{4, 8 + 6 + 2, 8 + 6\} \leq 16.$$

The depth of the formula defining the last permutation obviously does not exceed 14. In particular, it follows from this that the choice of $\widehat{\pi}_i$, $i = \{1, 2\}$, is essential for the implementation of permutations on hardware-software platforms. At the same depth, the substitution $G$ defined earlier has combinational complexity equal to 144.

If all $i \in \{\alpha, \beta, \gamma, \delta\}$ defining monomial permutations belong to the set $\{4, 7, 11, 13, 14\}$, then one can use the normal basis representation of permutations to reduce the corresponding chain depth by 1. In this case, the combinational complexity of the realization of specific substitutions can either remain unchanged (1, 7, 11, 13, 14), decrease (4), or increase (2, 8).

It turns out that the use of mixed bases potentially reduces both the combinational complexity and the depth of the formula specifying the permutation.

The use of PRR and RRB representations is justified only when it is necessary to minimize the depth of the formula specifying the permutation. However, they can be useful when implementing a large number of linear monomial permutations. In this case, additionally it is necessary to take into account the labor intensity of transformation from polynomial/normal bases to the specified representations and back.

# 5 Conclusions

In this paper the estimations of combinatorial complexity and depth of functions realizing permutations from the parametric family of type "G", which also generalize the parametric types "A" and "B" in the case of monomial choice of parameters $\pi_1, \pi_2$ are considered.

The results obtained in this work can be used in the implementation of the mentioned permutations on various software and hardware platforms.

# Список литературы


[1] Markus Ullrich, Christophe De Cannière, Sebastiaan Indesteege, Özgül Küçük, Nicky Mouha, and Bart Preneel. Finding optimal bitsliced implementations of 4 x 4-bit s-boxes. Ecrypt II, 2011.

[2] Chichaeva Anastasia. Searching for efficiently implemented permutations with optimal cryptographic characteristics (in Russian). In *Ruscrypto 2021*, 2021.

[3] Jérémy Jean, Thomas Peyrin, Siang Meng Sim, and Jade Tourteaux. Optimizing implementations of lightweight building blocks. *IACR Trans. Symmetric Cryptol.*, 2017(4):130–168, 2017.

[4] R.L. Rudell. Multiple-Valued Logic Minimization for PLA Synthesis. In *Technical report, EECS Department, University of California, Berkeley*, 1986.

[5] Fičer P. Hlavička, J. A Heuristic Boolean Minimizer. In *ICCAD'01*, 2001.

[6] O. D. Avraamova, D. B. Fomin, V. A. Serov, A. V. Smirnov, V. N. Shokov A compact bitsliced representation of Kuznyechik S-box *Matematicheskie Voprosy Kriptografii,*, 12(2):21–38, 2021.

[7] Yasuyuki Nogami, Kenta Nekado, Tetsumi Toyota, Naoto Hongo, and Yoshitaka Morikawa. Mixed bases for efficient inversion in $\mathbb{F}((2^2)^2)2$ and conversion matrices of subbytes of aes. pages 234–247, 08 2010.

[8] National Institute of Standards and Technology. Interagency or internal report 8369. *NIST*, 2021. `https://doi.org/10.6028/NIST.IR.8369`.

[9] John E. Savage *The Complexity of Computing* John Wiley & Sons; First Edition, 1976.

[10] T.L. Boot. Digital networks and computer systems. *New York: Wiley*, 1971.

[11] Lupanov O.B. On the realization of logic algebra functions by formulas from finite classes (formulas limited depth) in the basis & ,∨, ¯. (in Russian) *Problems of Cybernetics. Issue. 6. M.: Fizmatgiz*, pages C. 5–14, 1961.

[12] Yasuyuki Nogami, Kenta Nekado, Tetsumi Toyota, Naoto Hongo, and Yoshitaka Morikawa. Mixed bases for efficient inversion in $gf((2^2)^2)^2$ and conversion matrices of subbytes of aes. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, 94-A(6):1318–1327, 2011.

[13] Germain Drolet. A New Representation of Elements of Finite Fields GF(2m) Yielding Small Complexity Arithmetic Circuits. *IEEE Trans. Computers*, 47(9):938–946, 1998.

[14] Huapeng Wu. Low complexity bit-parallel finite field arithmetic using polynomial basis. In Çetin Kaya Koç and Christof Paar, editors, *CHES*, volume 1717 of *Lecture Notes in Computer Science*, pages 280–291. Springer, 1999.

[15] Rei Ueno, Naofumi Homma, Yasuyuki Nogami, and Takafumi Aoki. Highly efficient gf(28) inversion circuit based on hybrid gf representations. *J. Cryptographic Engineering*, 9(2):101–113, 2019.

[16] Kenta Nekado, Yasuyuki Nogami, and Kengo Iokibe. Very Short Critical Path Implementation of AES with Direct Logic Gates. In Goichiro Hanaoka and Toshihiro Yamauchi, editors, *IWSEC*, volume 7631 of *Lecture Notes in Computer Science*, pages 51–68. Springer, 2012.

[17] D.B. Fomin. New classes of 8-bit permutations based on a butterfly structure. *Matematicheskie Voprosy Kriptografii*, 10(2):169–180, 2019.

[18] D.B. Fomin. Construction of permutations on the space $V_{2m}$ by means of $(2m, m)$-functions. *Matematicheskie Voprosy Kriptografii*, 11(3):C. 121–138, 2020.

[19] D.B. Fomin. On the algebraic degree and differential uniformity of permutations on the space $V_{2m}$, constructed via $(2m, m)$-functions. *Matematicheskie Voprosy Kriptografii*, 11(4):C. 133–149, 2020.

[20] M.A. Kovrizhnykh and D.B Fomin. On differential uniformity of permutations derived using a generalized construction. *Matematicheskie Voprosy Kriptografii*, 13(2):C. 37–52, 2022.

[21] D.I. Trifonov and D.B. Fomin. About the hardware implementation of one byte permutation class. *Applied Discrete Mathematics. Applications*, 12:P. 134–137, 2019.

[22] Kovrizhnykh M. A. and Fomin D. B. Heuristic algorithm for obtaining permutations with given cryptographic properties using a generalized construction *Applied Discrete Mathematics.*, 57:P. 5–21, 2022.

[23] Markku-Juhani O. Saarinen. Cryptographic Analysis of All 4 x 4 - Bit S-Boxes. In *IACR Cryptology ePrint Archive*, 2011.

[24] Alex Biryukov, Léo Perrin, and Aleksei Udovenko. Reverse-Engineering the S-Box of Streebog, Kuznyechik and STRIBOBr1. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT (1)*, volume 9665 of *Lecture Notes in Computer Science*, pages 372–402. Springer, 2016.

[25] Reynier Antonio de la Cruz Jiménez. Generation of 8-bit s-boxes having almost optimal cryptographic properties using smaller 4-bit s-boxes and finite field multiplication. `www.cs.haifa.ac.il/~orrd/LC17/paper60.pdf`.

[26] David Canright. A Very Compact S-Box for AES. In Josyula R. Rao and Berk Sunar, editors, *CHES*, volume 3659 of *Lecture Notes in Computer Science*, pages Pp. 441–455. Springer, 2005.

[27] Sumio Morioka and Akashi Satoh. An optimized s-box circuit architecture for low power aes design. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *CHES, ed. by Jr., Burton S. Kaliski and Koç, Çetin Kaya and Paar, Christof*, volume 2523 of *Lecture Notes in Computer Science*, pages Pp. 172–186. Springer, 2002.

[28] Toshiya Itoh and Shigeo Tsujii. A fast algorithm for computing multiplicative inverses in $GF(2^m)$ using normal bases. *Inf. Comput.*, 78(3):171–177, September 1988.

# On one way of constructing unbalanced TU-based permutations

Denis Fomin

HSE University, Moscow, Russia
dfomin@hse.ru

**Abstract**

This article presents a new approach of constructing bijective nonlinear transformations. Some known results are generalized using the proposed approach and classes of differentially 4 and 6-uniform permutations of the space $\mathbb{F}_2^n$ for arbitrary even $n \geq 6$ are presented.

**Keywords:** permutation, TU decomposition, differential uniformity, S-Box

## Introduction

S-Boxes is a core element of modern cryptographic algorithms. In this work we study bijective S-Boxes (or permutations). Their choice is conditioned by the need to guarantee the ability of the cryptographic algorithm to resist known methods of cryptographic analysis. The efficiency of linear [1, 2, 3], differential [4, 3] and some types of algebraic methods of cryptographic analysis [5, 6, 7, 8] directly depends on the cryptographic characteristics of the substitution algorithm used [9]:

– nonlinearity;

– differential $\delta$-uniformity;

– algebraic degree;

– graph algebraic immunity.

The value of cryptographic characteristics of low-dimensional permutations is well studied, but they are far from similar values even for random permutations of large dimensions. Thus, their use in the synthesis of promising cryptographic algorithms, requires the implementation of more substitutions while maintaining the same level of security. Thus, there are many reasons for constructing higher-dimensional permutations using functions defined over lower-dimensional spaces:

– it is possible to implement it using T-tables,

– it is possible to implement it with using small number of logical operators,

– it is possible to implement it on FPGA,

– effective hardware masking is available [10, 11].

There are a large number of ways to build such nonlinear transformations: based on the Feistel network [12, 13, 14], using the Misty network [15, 12, 16], SPN-network [17, 18, 19] or other constructions [20]. At the same time, for the nonlinear bijective transformations listed above, nonlinearity are usually not higher than those obtained by random search.

Thus, a rather interesting task is to construct nonlinear bijective transformations, represented using functions from arguments of lower dimensional, whose nonlinearity indices will be better than similar ones obtained by random search. One such method is based on the use of a "butterfly" type permutation, which was proposed during the study [21] of the possibility to decompose the only known 6-bit differential 2-uniform permutation [22] and the method of decomposition construction for the nonlinear transformation of Russian cryptographic standards [23].

# 1 Definitions

Let $\mathbb{F}_2 = \{0, 1\}$ be a finite field with two elements, $(\mathbb{F}_2^n, +) = \left\{(a_0, a_1, \ldots, a_{n-1}), a_i \in \mathbb{F}_2, i \in \overline{0, n-1}\right\}$ — vector space of dimension $n$, 0 — is a neutral element of the vector space.

If we consider the additive group of vector space $\mathbb{F}_2^n$ and specify the multiplication operation in a special way, we can construct a field, which we will denote by $\mathbb{F}_{2^n}$.

**Definition 1.** *The vector Boolean function (or $(n, m)$-function) $S$ is the mapping $\mathbb{F}_2^n \to \mathbb{F}_2^m$, $n, m \in \mathbb{N}$.*

**Definition 2.** *Walsh-Hadamard transform $W_S(a, b)$ of $(n, m)$-function $S$ for $a \in \mathbb{F}_2^n$, $b \in \mathbb{F}_2^m$ is defined as follows:*

$$W_S^{a,b} = \sum_{x \in \mathbb{F}_2^n} (-1)^{\langle a,x \rangle + \langle b,S(x) \rangle}.$$

**Definition 3.** *The nonlinearity of $(n, m)$-function $S$ is denote as $N_S$ and defined as follows:*

$$N_S = 2^{n-1} - \frac{1}{2} \max_{\substack{a \in \mathbb{F}_2^n, \\ b \in \mathbb{F}_2^m \setminus 0}} \left| W_S^{a,b} \right|.$$

The nonlinearity of $(n, m)$-function characterizes its distance from the set of linear functions of the same dimension [9]. This characteristic allows us to evaluate the applicability of the linear attack [24, 1, 2, 3].

An arbitrary $(n, m)$-function $F$ can be represented by its coordinate functions: $F(\overline{x}) = (f_1(\overline{x}), f_2(\overline{x}), \ldots, f_m(\overline{x}))$, where $\overline{x} = (x_1, x_2 \ldots, x_n) \in \mathbb{F}_2^n$, $f_i(\overline{x})$ — a Boolean function, $i \in \overline{1, m}$.

**Definition 4.** *The algebraic degree (minimum degree)* $\deg(S)$ *of $(n, m)$-function $S$ is the minimum degree among all the component functions of $S$: $\langle a, S(x) \rangle$, $a \in \mathbb{F}_2^m \setminus 0$:*

$$\deg(S) = \min_{a \in \mathbb{F}_2^m \setminus 0} \deg\left(\langle a, S(x) \rangle\right).$$

**Definition 5.** *The maximum degree of $(n, m)$-function $S$ is the maximum degree among all the component functions of $S$: $\langle a, S(x) \rangle$, $a \in \mathbb{F}_2^m \setminus 0$:*

$$\deg_m(S) = \max_{a \in \mathbb{F}_2^m \setminus 0} \deg\left(\langle a, S(x) \rangle\right).$$

The minimum degree allows us to evaluate the applicability of the interpolation od some other algebraic attacks, [5, 6, 7]. It's know that for $n \geq 3$ a balanced Boolean function has the algebraic degree up to $n - 1$, [9].

**Definition 6.** *For $a \in \mathbb{F}_2^n \setminus 0, b \in \mathbb{F}_2^m$ let*

$$\delta_S^{a,b} = \left| \{ x \in \mathbb{F}_{2^n} | S(x + a) + S(x) = b \} \right|.$$

*An $(n, m)$-function $S$ is called differentially $\delta_S$-uniform if*

$$\delta_S = \max_{\substack{a \in \mathbb{F}_2^n \setminus 0, \\ b \in \mathbb{F}_2^m}} \delta_S^{a,b}.$$

The differential uniformity of $(n, m)$-function allows us to evaluate the applicability of the differential attack [24, 4, 3]. The smallest possible value for this characteristic is 2, however, the only one differential 2-uniform permutation is known for space $\mathbb{F}_{2^m}$ for even $m$ (up to CCZ-equivalent class), [21].

Consider the set $\mathcal{G}_k$ of $(n+m, 1)$-functions $G(x_1, \ldots, x_n, y_1, \ldots, y_m)$, such that $\deg(G) \leq k$, $k \in \mathbb{N}$ and for each $\overline{x} \in \mathbb{F}_2^n$ if we substitute in place of each

variable $y_i$, $i \in \overline{1,m}$, the value of the corresponding Boolean function $f_i(\overline{x})$, then the value of the function $G(x_1,\ldots,x_n,f_1(\overline{x}),\ldots,f_m(\overline{x}))$ is equals to 0:

$$\mathcal{G}_k = \{G(x_1,\ldots,x_n,y_1,\ldots,y_m)\colon G(x_1,\ldots,x_n,f_1(\overline{x}),\ldots,f_m(\overline{x})) = 0\ \forall \overline{x} \in \mathbb{F}_2^n\}.$$

The set $\mathcal{G}_k$ is a subgroup of the ring of polynomials of degree non above $k$. Let's denote $r_F^k$ — the basis size of $\mathcal{G}_k$.

**Definition 7.** *A minimum number $k$ such that $r_F^k \neq 0$, is called graph algebraic immunity of $F$ and denoted by $AI_{gr}(F)$, [9].*

In [8] an attack using low graph algebraic immunity of nonlinear transformation is proposed. Thus, the use of functions with high value of $AI_{gr}(F)$ could increase the resistance of the symmetric algorithm to some algebraic attacks.

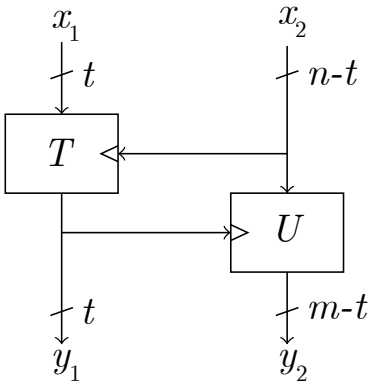



Figure 1: $TU$-representation of an $(n,m)$-function, [25]

Let's recall the definition of the $TU$-representation according to [25]:

**Definition 8** ([25])**.** *Let $F$ be an $(n,m)$-function, $1 \leq t \leq \min(n,m)$, $x_1$, $y_1 \in \mathbb{F}_2^t$, $x_2 \in \mathbb{F}_2^{n-t}$, $y_2 \in \mathbb{F}_2^{m-t}$, $x \in \mathbb{F}_2^n$, $x = x_1\|x_2$, $y = y_1\|y_2$, $T(x_1,x_2)$ is a $(n,t)$ function such that if we fix value $x_2$ by any value from $\mathbb{F}_{2^{n-t}}$ then the function $T$ is a bijection for value $x_1$, $U$ is any $(n,m-t)$-function. Then if the function $F$ has the following representation (see pic. 1):*

$$F(x) = F(x_1\|x_2) = (T(x_1,x_2), U(x_2,T(x_1,x_2))), \tag{1}$$

*then such representation of $F$ in the form (1) is called the $TU$-representation.*

S-Box of Russian block cipher and hash-function standards, permutation, that is CCZ-equivalent to only known differentially 2-uniform permutation of $\mathbb{F}_2^{2m}$, and some other permutations with good cryptographic characteristics has $TU$-representation, [26, 27, 21, 28, 29, 30, 31, 32].

## 2 On the way of constructing differentially $\delta$-uniform permutations

Consider the method of constructing permutations proposed in [33]. Let $\mathbb{F}_2^n$, $n \geq 6$ is a vector space with elements $v = (v_1, v_2, \ldots, v_n)$. For each element $v \in \mathbb{F}_2^n$ we put the match the pair $(v', v_n)$, where $v' \in \mathbb{F}_{2^{n-1}}$, $v' = v_{n-1}x^{n-2} + \cdots + v_1$, $\mathbb{F}_{2^{n-1}} = \mathbb{F}_2[x]/f(x)$, $\deg(f) = n-1$. This correspondence specifies bijective mapping of the set $\mathbb{F}_2^n$ to $\mathbb{F}_{2^{n-1}} \times \mathbb{F}_2$.

Let $\mathrm{tr}(x)$ be a trace function from the field $\mathbb{F}_{2^{n-1}}$ to $\mathbb{F}_2$. For any $c \in \mathbb{F}_{2^n-1} \backslash \{0, 1\}$ such that $\mathrm{tr}(c) = \mathrm{tr}\left(c^{-1}\right)$, and arbitrary Boolean function $g$ of $n-1$ variables in [33] the function $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$ us defined as follows:

$$F(v_1, v_2, \ldots, v_{n-1}, v_n) = \begin{cases} \left(v'^{-1}, g(v')\right), & v_n = 0 \\ \left(c \cdot v'^{-1}, g\left(v' \cdot c^{-1}\right) + 1\right), & v_n = 1 \end{cases}, \quad (2)$$

where $v' \in \mathbb{F}_{2^{n-1}}$, $v'$ is defined by the vector $(v_1, v_2, \ldots, v_{n-1}) \in \mathbb{F}_2^{n-1}$, $0^{-1} = 0$. It's shown, [33], that the function $F$ is differentially 4-uniform permutation, that has the maximal algebraic degree equals to $n-1$, and the nonlinearity less or equals to $2^{n-1} - 2\left\lfloor 2^{(n+1)/2} \right\rfloor - 4$.

Let's prove that permutation $F$ that is defined by (2) has $TU$-representation.

**Proposition 1.** *Let $x_1 \in \mathbb{F}_2^{n-1}$, $x_2 \in \mathbb{F}_2$,*

- $T\colon \mathbb{F}_2^{n-1} \times \mathbb{F}_2 \to \mathbb{F}_2^{n-1}$, $T(x_1, x_2) = x_1^{-1} \cdot c^{x_2}$,
- $U\colon \mathbb{F}_2^1 \times \mathbb{F}_2^{n-1} \to \mathbb{F}_2$, $U(x_2, x_1) = g\left(x_1^{-1}\right) + x_2$.

*Then*

1. *if we fix $x_2$ by an arbitrary value from $\mathbb{F}_2$ then the function $T$ is a bijection on the varible $x_1$,*
2. *if we fix $x_1$ by an arbitrary value from $\mathbb{F}_2^{n-1}$ then the function $U$ is a bijection on the variable $x_2$,*
3. *functions $T$ and $U$ define a $TU$-representation of permutation defined by (2).*

*Proof.* Proof is obvious and can be made by direct verification. □

Let's prove a more general statement which allows us to guarantee $\delta$-uniformity of the function $F$, given by a $TU$-representation of special form. For the function $F$ that has a $TU$-representation given by equation (1), denote

– for $a \in \mathbb{F}_2^t$ the value $\delta_{T,a}$ is equal to $\delta$ if permutation $T$ with fixed $x_2 = a$ is differentially $\delta$-uniform,

– for $a \in \mathbb{F}_2^t$, $\alpha_1$, $\beta_1 \in \mathbb{F}_2^{n-t}$, $\alpha_2 \in \mathbb{F}_2^t \backslash \theta$, the value $\Delta_{T,a}^{\alpha_1,\alpha_2,\beta_1}$ is the number of solutions to the equation:

$$T(x_1, a) + T(x_1 + \alpha_1, a + \alpha_2) = \beta_1.$$

**Theorem 1.** *Let $n, t \in \mathbb{N}$, $1 \leq t \leq n-1$, $x_1 \in \mathbb{F}_2^{n-t}$, $x_2 \in \mathbb{F}_2^t$,*

– *function $T\colon \mathbb{F}_2^{n-t} \times \mathbb{F}_2^t \to \mathbb{F}_2^{n-1}$ such that fixation $x_2$ by arbitrary value from $\mathbb{F}_2^t$ the function $T(x_1, x_2)$ is the permutation on the variable $x_1$,*

– *function $U\colon \mathbb{F}_2^t \times \mathbb{F}_2^{n-t} \to \mathbb{F}_2^t$ such that fixation $x_1$ by arbitrary value from $\mathbb{F}_2^{n-t}$ the function $U(x_2, x_1)$ is the permutation on the variable $x_2$.*

*Then the permutation $F$, defined by (1) is differentially $\delta$-uniform, where*

$$\delta \leq 2^t \cdot \max \left\{ \max_{a \in \mathbb{F}_2^t} (\delta_{T,a}), \max_{\substack{\alpha_1, \beta_1 \in \mathbb{F}_2^{n-t}, \\ a \in \mathbb{F}_2^t,\ \alpha_2 \in \mathbb{F}_2^t \backslash \theta}} \left( \Delta_{T,a}^{\alpha_1,\alpha_2,\beta_1} \right) \right\}. \tag{3}$$

*Proof.* As we can see from expression (3) the value $d\delta$ is defined only by the the properties of the function $T$. The description of the function $U$ is necessary only for the correct definition of the bijective transformation of the space $\mathbb{F}_2^n$.

Let $\alpha_1$, $\beta_1 \in \mathbb{F}_2^{n-t}$, $\alpha_2$, $\beta_2 \in \mathbb{F}_2^t$, then from $\alpha_i = \theta$ follows that $\alpha_j \neq \theta$, $i, j \in \{0, 1\}$, $i \neq j$. To determine the value of $\delta$ of permutation $F$, it is necessary to know the number of solutions of the following equation:

$$F(x_1, x_2) + F(x_1 + \alpha_1, x_2 + \alpha_2) = \beta_1 \| \beta_2. \tag{4}$$

Consider the case $\alpha_2 = \theta$. In this case $\alpha_1 \neq \theta$ and (4) can be rewritten as follows(see. eq. (1)):

$$\begin{cases} T(x_1, x_2) + T(x_1 + \alpha_1, x_2) = \beta_1, \\ U(x_2, T(x_1, x_2)) + U(x_2, T(x_1 + \alpha_1, x_2)) = \beta_2. \end{cases}$$

The number of solutions of the last system is not greater than the number of solutions of the equation:

$$T(x_1, x_2) + T(x_1 + \alpha_1, x_2) = \beta_1.$$

Let's fix the value $x_2 = a$. In this case, the number of solutions of the last equation is not greater than $\delta_{T,a}$. Since the value of $x_2$ can be fixed in $2^t$ different ways, the number of solutions to equation (4) with $\alpha_2 \neq 0$ is no greater than $2^t \cdot \max\limits_{a \in \mathbb{F}_2^t} (\delta_{T,a})$.

Let $\alpha_2 \neq 0$. In this case the equation (4) has the following representation (see eq. (1)):

$$\begin{cases} T(x_1, x_2) + T(x_1 + \alpha_1, x_2 + \alpha_2) = \beta_1, \\ U(x_2, T(x_1, x_2)) + U(x_2, T(x_1 + \alpha_1, x_2 + \alpha_2)) = \beta_2. \end{cases}$$

As in the previous case, the number of solutions of the last system is not greater than the number of solutions of the equation

$$T(x_1, x_2) + T(x_1 + \alpha_1, x_2 + \alpha_2) = \beta_1.$$

There are $2^t$ ways to solve the last equation fixing $x_2 = a$, and the number of solutions equal to $\Delta_{T,a}^{\alpha_1,\alpha_2,\beta_1}$, which proves the theorem. □

Let's show that theorem 3 allows us to construct differentially $\delta$-uniform permutations.

**Corollary 1.** *In the conditions of theorem 1 let* $t = 1$, $\delta_{T,a} \leq \delta$, $a \in \mathbb{F}_2$, *then the permutation* $F$, *defined by* (1) *is differentially* $2\delta$*-uniform* $\max\limits_{\alpha_1, \beta_1 \in \mathbb{F}_2^{n-1}} \Delta_{T,0}^{\alpha_1,1,\beta_1} \leq \delta$.

According to the corollary 1 in order to construct a differentialy 4-uniform permutation $F$ one must take two differential 2-uniform permutations $\pi_1$ and $\pi_2$ of the space $\mathbb{F}_2^{n-1}$. And if $T(x_1, 0) = \pi_1$ and $T(x_1, 1) = \pi_2$, then it remains to check that the number of solutions of following equations:

$$\pi_1(x) + \pi_2(x + \alpha_1) = \beta_1$$

for all possible values of $\alpha_1, \beta_1 \in \mathbb{F}_2^{n-1}$ are not greater than 2.

**Proposition 2.** *Let* $x_1 \in \mathbb{F}_2^{n-1}$, $n$ *be an even number,* $x_2 \in \mathbb{F}_2$, $f$ *be an arbitrary Boolean function of* $n-1$ *variables,* $c \in \mathbb{F}_{2^{n-1}} \setminus \{\theta, 1\}$,

– $T\colon \mathbb{F}_2^{n-1} \times \mathbb{F}_2 \to \mathbb{F}_2^{n-1}$, $T(x_1, x_2) = x_1^{-1} \cdot c^{x_2}$,

– $U\colon \mathbb{F}_2^1 \times \mathbb{F}_2^{n-1} \to \mathbb{F}_2$, $U(x_2, x_1) = f(x_1) + x_2$.

*Then equation* (1) *defines the permutation* $F$, *and at the same time*

1. *if* $\mathrm{tr}(c) = \mathrm{tr}\left(c^{-1}\right) = 1$, *then* $\delta_F = 4$,

*2. otherwise* — $\delta_F = 6$.

*Proof.* The equation (2) obviously defines a permutation and for any $c \neq \theta$ function $c \cdot x_1^{-1}$ is a differentially 2-uniform permutation (since $n-1$ is an odd number). It remains to find the maximum number of solutions to the equations:

$$x_1^{-1} + c \cdot (x_1 + \alpha_1)^{-1} = \beta_1 \tag{5}$$

for all $\alpha_1$, $\beta_1 \in \mathbb{F}_2^{n-1}$.

According to [34], the equation $Ax^2 + Bx + C \in \mathbb{F}_{2^n}[x]$, $A \in \mathbb{F}_{2^n}^*$, $B$, $C \in \mathbb{F}_{2^n}$, has exactly 2 solutions if and only if $\mathrm{tr}_1^n\left(B^{-2} \cdot A \cdot C\right) = 0$ and has 0 solutions otherwise.

Let $x_1 = 0$ be the solution to equation (5), then

$$c = \alpha_1 \cdot \beta_1.$$

If $x_1 = \alpha_1$ is a solution to the equation, then

$$\alpha_1^{-1} + c \cdot \theta = \beta_1 \Rightarrow \alpha_1 \cdot \beta_1 = 1.$$

Let's multiply equation (5) by $x_1 \cdot (x_1 + \alpha_1)$:

$$x_1 + \alpha_1 + c \cdot x_1 = \beta_1 \cdot x_1 \cdot (x_1 + \alpha_1) \Rightarrow$$
$$\Rightarrow \beta_1 \cdot x_1^2 + x_1 \cdot (\beta_1 \cdot \alpha_1 + c + 1) + \alpha_1 = 0. \tag{6}$$

Consider the case $\beta_1 = \theta$. Then by the condition of the proposition and equality (2) $x_1 = 0$ is not a solution of equation (5). In this case the last equation has one solution: $x_1 = \alpha_1 \cdot (c+1)^{-1}$ (the last expression is correct since $c \neq 1$). From here we can also easily show that if $c = 1$, then equation (5) has $2^n$ solutions.

If $\beta \neq \theta$, then equation (6) has no more than two solutions. Then at most equation (5) has no more than 3 solutions since if $x_1 = 0$ and $x_1 = \alpha_1$ are simultaneously solutions of (5), then $c = 1$, which contradicts the proposition condition. Thus $\delta_F \leq 6$.

For the equality $\delta_F = 4$ to hold, it is necessary that in the case where $x_1 = 0$ or $x_1 = \alpha_1$ equation (6) has no solutions, that is equivalent to the fact that

$$\mathrm{tr}\left(\alpha_1 \cdot \beta_1 \cdot (\beta_1 \cdot \alpha_1 + c + 1)^{-2}\right) = 1.$$

Let $x_1 = 0$ be the solution. Then

$$\mathrm{tr}\left(\alpha_1 \cdot \beta_1 \cdot (\beta_1 \cdot \alpha_1 + c + 1)^{-2}\right) = \mathrm{tr}\,(c)\,.$$

If $x_1 = 0$ is a solution, then

$$\operatorname{tr}\left(\alpha_1 \cdot \beta_1 \cdot (\beta_1 \cdot \alpha_1 + c + 1)^{-2}\right) = \operatorname{tr}\left(c^{-1}\right).$$

The last two equalities prove the proposition. □

**Remark 1.** *The proof of point 1 of the previous proposition was previously proved in [33].*

The following proposition is proved in a similar way:

**Proposition 3.** *Let $x_1 \in \mathbb{F}_2^{n-1}$, $n$ be an even number, $x_2 \in \mathbb{F}_2$, $f$ be an arbitrary Boolean function of $n-1$ variables, $c \in \mathbb{F}_{2^{n-1}} \setminus \{\theta, 1\}$,*

- $T\colon \mathbb{F}_2^{n-1} \times \mathbb{F}_2 \to \mathbb{F}_2^{n-1}$, $T(x_1, x_2) = x_1^3 \cdot c^{x_2}$,
- $U\colon \mathbb{F}_2^1 \times \mathbb{F}_2^{n-1} \to \mathbb{F}_2$, $U(x_2, x_1) = f(x_1) + x_2$.

*Then equation* (1) *defines the permutation $F$, and $\delta_F = 6$.*

*Proof.* In the case of even $n$ the function $c \cdot x_1^3$ is a differentially 2-uniform permutation for any $c \in \mathbb{F}_{2^{n-1}}^*$.

Then it must be shown that

$$c \cdot x_1^3 + (x_1 + \alpha_1)^3 = \beta_1 \tag{7}$$

has no more than 3 solutions, and there are such $\alpha_1$ and $\beta_1$, at which this equation has exactly 3 solutions.

Since this is an equation of degree 3, it can have no more than 3 solutions. Further proof will be done as follows. Let $y$ be a solution of equation (7). Fixing an arbitrary solution uniquely specifies $\beta_1$. Then dividing equation (7) by $(x_1 - y)$ we obtain a quadratic equation and show that it has 2 different solutions (that are not equal to $y$), which will show that the total number of solutions is 3.

Let $y$ be the solution to equation (7), then

$$\beta_1 = c \cdot y^3 + (y + \alpha_1)^3.$$

Let's divide $c \cdot x_1^3 + (x_1 + \alpha_1)^3 = \beta_1$ by $x_1 + y$:

$$\begin{aligned}(7) = (x_1 + y) \cdot \\ \cdot \left[x_1^2(1+c) + x_1(\alpha_1 + y(1+c)) + \right. \\ \left. + \alpha_1^2 + y \cdot \alpha_1 + y^2(1+c)\right]. \end{aligned} \tag{8}$$

Let $y = 0$. First, $y$ is not the root of the equation

$$x_1^2(1+c) + x_1\alpha_1 + \alpha_1^2, \tag{9}$$

and second, this equation has 2 solutions if $\mathrm{tr}\left((1+c)\alpha_1^2\alpha_1^{-2}\right) = 0$ or if $\mathrm{tr}\,(c) = 1$.

It remains to show that if $\mathrm{tr}(c) = 0$, , then equation (7) also has 3 solutions. The equation

$$x_1^2(1+c) + x_1\left(\alpha_1 + y(1+c)\right) + \alpha_1^2 + y\cdot\alpha_1 + y^2(1+c) = 0$$

has 2 solutions if

1. $\alpha \neq y(1+c)$,
2. $$\mathrm{tr}\left(\frac{(1+c)\left(\alpha_1^2 + y\cdot\alpha_1 + y^2(1+c)\right)}{\alpha_1^2 + y^2(1+c^2)}\right) = 0.$$

Let

$$\frac{(1+c)\left(\alpha_1^2 + y\cdot\alpha_1 + y^2(1+c)\right)}{\alpha_1^2 + y^2(1+c^2)} = v.$$

Let us find the conditions on $v$, for which the last expression will give the correct equality:

$$\begin{aligned}(1+c)\alpha_1^2 + (1+c)y\alpha_1 + y^2(1+c^2) = v\alpha_1^2 + y^2v(1+c^2) \Rightarrow \\ \Rightarrow \alpha_1^2(1+c+v) + \alpha_1\cdot(1+c)y + \\ + \left(y^2(1+c^2) + y^2v(1+c^2)\right) = 0. \end{aligned} \tag{10}$$

Consider equation (10) as an equation on the variable $y$. Then it has two solutions if

$$\mathrm{tr}\left(\frac{(1+c+v)y^2(v+1)(1+c^2)}{y^2\cdot(1+c^2)}\right) = 0,$$

which is equivalent to saying that

$$\mathrm{tr}\left(1+c+cv+v^2\right) = 0.$$

In the case $\mathrm{tr}(c) = 0$ equation (10) will be solvable for the variable $y$, if for arbitrary $c \in \mathbb{F}_{2^{n-1}}\backslash\{0,1\}$, $\mathrm{tr}(c) = 0$ the equality is satisfied:

$$\mathrm{tr}(cv + v^2) = 1.$$

Let us show that there are always such $v$. Suppose the contrary: let $\mathrm{tr}(cv + v^2) = 0$. Then consider $v' = v + 1$:

$$\mathrm{tr}\left(c(v+1) + (v+1)^2\right) = \mathrm{tr}\left(cv + c + v + 1\right) = \\ = \mathrm{tr}(cv + v^2) + 1 = 1. \quad (11)$$

That is, there exist such $v$, that equation (10) is solvable for the variable $y$, and $\mathrm{tr}(cv + v^2) = 1$. Denote by $V$ is a set of $v$ such that

$$V = \left\{v \,\middle|\, \mathrm{tr}\left(cv + v^2\right) = 1\right\}.$$

Let's show, that this set there is $v$ such that, $\mathrm{tr}(v) = 0$. Let's suppose the contrary:

$$V = \left\{v \,\middle|\, \mathrm{tr}\left(cv + v^2\right) = 1\right\} = \left\{v \,|\mathrm{tr}\,(v) = 1\right\}.$$

Then let's consider the following set

$$\overline{V} = \left\{v \,\middle|\, \mathrm{tr}\left(cv + v^2\right) = 0\right\} = \left\{v \,|\mathrm{tr}\,(v) = 0\right\}.$$

The sets $\{v \,|\mathrm{tr}\,(v) = 0\}$ and $\{v \,|\mathrm{tr}\,(v) = 1\}$ are not equal. At the same time since $\mathrm{tr}(v) = \mathrm{tr}\left(v^2\right)$, then sets $V$ and $\overline{V}$are equal, which leads to a contradiction.

Let's summarize. In the case $\mathrm{tr}(c) = 0$ , there exists $v$ such that $\mathrm{tr}(v) = 0$ and equation (10) has 2 different roots other than $y$, which completes the proof. □

The proof of statement 3 uses the fact that an equation of the third degree cannot have more than 3 solutions, that allows us to construct differential 6-uniform permutations. That's why the following hypothesis was proposed. Let $x_1$, $a$, $b \in \mathbb{F}_{2^{n-1}}$,

- $T(x_1, 0) = x_1^3$,
- $T(x_1, 1) = x_1^3 + a \cdot x_1^2 + b \cdot x_1$.

Both functions $T(x_1, 0)$ and $T(x_1, 1)$ are differentially 2-uniform as extended affine-equivalent to the differentially 2-uniform permutations. Then if we find such $a$, $b \in \mathbb{F}_{2^{n-1}}$ that $T(x_1, 1)$ is a permutation , then the equation for variable $x_1$

$$T\left(x_1 + \alpha_1, 0\right) + T(x_1, 1) = \beta_1$$

will be an equation with a degree of nonlinearity no greater than 2, which allows us to assume that such a substitution defined by equation (1) will be differentially 2-uniform permutation. However, the following proposition shows that for any $a$, $b \in \mathbb{F}_{2^{n-1}}$ the constructed permutation will not be differentially 2-uniform.

**Proposition 4.** *Let $x_1 \in \mathbb{F}_2^{n-1}$, $n$ be an even number, $x_2 \in \mathbb{F}_2$, $a$, $b \in \mathbb{F}_{2^{n-1}}$, $T\colon \mathbb{F}_2^{n-1} \times \mathbb{F}_2 \to \mathbb{F}_2^{n-1}$*

- $T(x_1, 0) = x_1^3$,
- $T(x_1, 1) = x_1^3 + a \cdot x_1^2 + b \cdot x_1$.

*Then either $\alpha_1 \in \mathbb{F}_{2^{n-1}}$ and $\beta_1 \in \mathbb{F}_{2^{n-1}}$ exist such that the number of solutions to the equation*

$$T(x_1 + \alpha_1, 0) + T(x_1, 1) = \beta_1$$

*will equal to $2^{n-1}$ or $T(x_1, 1)$ is not a permutation.*

*Proof.* For each $\alpha_1 \in \mathbb{F}_{2^{n-1}}$ and $\beta_1 \in \mathbb{F}_{2^{n-1}}$ the equation

$$T(x_1 + \alpha_1, 0) + T(x_1, 1) = \beta_1$$

can be represented as follows:

$$x_1^2(\alpha_1 + a) + x_1\left(b + \alpha_1^2\right) + \alpha_1^3 + \beta_1 = 0.$$

Then if $a = \alpha_1$ and $b = \alpha_1$, then taking $\beta_1 = \alpha_1^3$ we obtain the equation not depending on variable $x_1$ and having $2^{n-1}$ solutions.

Let us show that if $b \neq a^2$, then the function

$$x_1^3 + a \cdot x_1^2 + b \cdot x_1$$

is not a permutation. Consider the equation

$$x_1^3 + a \cdot x_1^2 + b \cdot x_1 = c.$$

It must be shown that for at least one $c \in \mathbb{F}_{2^{n-1}}$ this equation has exactly one solution. The equation

$$x_1^3 + a \cdot x_1^2 + b \cdot x_1 = c$$

will be a permutation if for arbitrary $c \in \mathbb{F}_{2^{n-1}}$ there is no multiple roots. This is equivalent that this equation with

$$x^2 + b = \left(x + b^{1/2}\right)^2.$$

is have greatest common division equals to 1 Note that if we fix an arbitrary solution $y \in \mathbb{F}_{2^{n-1}}$, the value $c$ is expressed by the equation:

$$c = y^3 + ay^2 + by.$$

It is worth noting that in the case $a = b^2$ the function $T(x_1, 1)$ has the form

$$(x_1 + a)^3 + const$$

and always has one solution (although formally a multiple) and sets up a permutation. □

Let us omit the proof of the following proposition, which is carried out similarly to the proof of the previous propositions and is a purely technical task.

**Proposition 5.** *Let* $x_1 \in \mathbb{F}_2^{n-1}$, $n$ *e an even number,* $x_2 \in \mathbb{F}_2$, $f$ *be an arbitrary Booleann function of* $n-1$ *variables,* $c \in \mathbb{F}_{2^{n-1}} \setminus \{\theta, 1\}$,

- $T\colon \mathbb{F}_2^{n-1} \times \mathbb{F}_2 \to \mathbb{F}_2^{n-1}$, $T(x_1, 0) = x_1^3$, $T(x_1, 0) = x_1^{-1}$ ,
- $U\colon \mathbb{F}_2^1 \times \mathbb{F}_2^{n-1} \to \mathbb{F}_2$, $U(x_2, x_1) = f(x_1) + x_2$.

*Then the equation* (1) *specifies differentially 8-uniform permutation.*

Here is another interesting example of the application of theorem 1.

**Proposition 6.** *Let* $t = 2$, $x_1 \in \mathbb{F}_2^{n-t}$, $x_2 \in \mathbb{F}_2^t$,

- $T\colon \mathbb{F}_2^{n-t} \times \mathbb{F}_2^t \to \mathbb{F}_2^{n-1}$, *when fixing an arbitrary* $x_2$ *function* $T(x_1, x_2) = x_1^{-1} \cdot c^{x_2}$,
- $U\colon \mathbb{F}_2^t \times \mathbb{F}_2^{n-t} \to \mathbb{F}_2^t$, *when fixing an arbitrary* $x_1$ *function* $U(x_2, x_1)$ *is a permutation on the variable* $x_2$.

*Then there exist such* $c_y$, $y \in \mathbb{F}_{2^2}$, $c_{y_1} \neq c_{y_2}$ *if* $y_1 \neq y_2$, *that the permutation* $F$, *given by equation* (1) *is a differentially 8-uniform permutation.*

The proof of this statement basically repeats the reasoning done in the proof of proposition 2 and will not be given here.

## 3 Cryptographic properties of some constructed differentially 4 and 5-uniform permutations

In [33] only the maximum algebraic degree was estimated. Let us show that the permutation proposed in this paper and considered in [33] can have an algebraic degree equal to 1, which potentially leads to the possibility of using algebraic methods of cryptographic analysis.

**Proposition 7.** *Let* $x_1 \in \mathbb{F}_2^{n-1}$, $n \in \mathbb{N}$ *be an even number,* $i \in \mathbb{N}$, $i \leq 2^{n-1} - 2$, $x_2 \in \mathbb{F}_2$, $c \in \mathbb{F}_{2^{n-1}} \setminus \{\theta, 1\}$, $T\colon \mathbb{F}_2^{n-1} \times \mathbb{F}_2 \to \mathbb{F}_2^{n-1}$, $T(x_1, x_2) = x_1^i \cdot c^{x_2}$, *then* $\deg T = |i| + 1$.

The proof is obvious. Note that in the case $i = 2^{n-2}$ we obtain a function $T$ having the maximal algebraic degree.

**Remark 2.** *If in statements of propositions 2 and 3 the function $f$ has an algebraic degree equal to 1, then the entire permutation $F$ will also have an algebraic degree equal to 1.*

**Proposition 8.** *Under the conditions of propositions 2 and 3, the permutation $F$ will have the graph algebraic immunity equals to 2.*

The proof obviously follows from the truth of the following quadratic relation:

$$y \cdot x_1 + x_2 \cdot (c + 1) + 1 = 0,$$

where $y = T(x_1, x_2)$.

# 4 Conclusions

New approaches to the construction of nonlinear bijective transformations are considered. We present classes differentially 4 and 6-uniform permutations of the space $\mathbb{F}_2^n$ for an arbitrary even $n \geq 6$.

# References


[1] Mitsuru Matsui., "The First Experimental Cryptanalysis of the Data Encryption Standard. In Yvo Desmedt, editor, *CRYPTO*, volume 839 of *Lecture Notes in Computer Science*, pages 1–11. Springer, 1994.".

[2] Lars R. Knudsen and M. J. B. Robshaw., "Non-linear approximations in Linear Cryptanalysis. In Ueli Maurer, editor, *Advances in Cryptology — EUROCRYPT '96*, pages 224–236, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.".

[3] Howard M. Heys., "A Tutorial on Linear and Differential Cryptanalysis. *Cryptologia*, 26(3):189–221, 2002.".

[4] Eli Biham and Adi Shamir., "Differential Cryptanalysis of DES-like Cryptosystems. *J. Cryptology*, 4(1):3–72, 1991.".

[5] Bing Sun, Longjiang Qu, and Chao Li., "New Cryptanalysis of Block Ciphers with Low Algebraic Degree. In Orr Dunkelman, editor, *Fast Software Encryption*, pages 180–192, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.".

[6] Thomas Jakobsen and Lars R. Knudsen., "Attacks on Block Ciphers of Low Algebraic Degree. *Journal of Cryptology*, 14(3):197–210, Jun 2001.".

[7] Håvard Raddum., "Algebraic Analysis of the Simon Block Cipher Family. volume 9230, pages 157–169, 08 2015.".

[8] Nicolas Courtois and Josef Pieprzyk., "Cryptanalysis of Block Ciphers with Overdefined Systems of Equations. Cryptology ePrint Archive, Report 2002/044, 2002. `https://eprint.iacr.org/2002/044`.".

[9] Claude Carlet., "Vectorial Boolean Functions for Cryptography. *Boolean Models and Methods in Mathematics, Computer Science, and Engineering*, 12 2006.".

[10] Erik Boss, Vincent Grosso, Tim Güneysu, Gregor Leander, Amir Moradi, and Tobias Schneider., "Strong 8-bit Sboxes with efficient masking in hardware extended version. *J. Cryptographic Engineering*, 7(2):149–165, 2017.".

[11] Sebastian Kutzner, Phuong Ha Nguyen, and Axel Poschmann., "Enabling 3-Share Threshold Implementations for all 4-Bit S-Boxes. In Hyang-Sook Lee and Dong-Guk Han, editors, *ICISC*, volume 8565 of *Lecture Notes in Computer Science*, pages 91–108. Springer, 2013.".

[12] Anne Canteaut, Sébastien Duval, and Gaëtan Leurent., "Construction of lightweight s-boxes using Feistel and MISTY structures (full version). *IACR Cryptology ePrint Archive*, 2015:711, 2015. `http://eprint.iacr.org/2015/711`.".

[13] Chae Hoon Lim., "CRYPTON: A new 128-bit block cipher - specification and analysis, 1998.".

[14] Benoît Gérard, Vincent Grosso, María Naya-Plasencia, and François-Xavier Standaert., "Block Ciphers That Are Easier to Mask: How Far Can We Go? In Guido Bertoni and Jean-Sébastien Coron, editors, *CHES*, volume 8086 of *Lecture Notes in Computer Science*, pages 383–399. Springer, 2013.".

[15] Mitsuru Matsui., "New block encryption algorithm MISTY. In Eli Biham, editor, *FSE*, volume 1267 of *Lecture Notes in Computer Science*, pages 54–68. Springer, 1997.".

[16] Vincent Grosso, Gaëtan Leurent, François-Xavier Standaert, and Kerem Varici., "Ls-designs: Bitslice encryption for efficient masked software implementations. In Carlos Cid and Christian Rechberger, editors, *FSE*, volume 8540 of *Lecture Notes in Computer Science*, pages 18–37. Springer, 2014.".

[17] François-Xavier Standaert, Gilles Piret, Gaël Rouvroy, Jean-Jacques Quisquater, and Jean-Didier Legat., "ICEBERG : An involutional cipher efficient for block encryption in reconfigurable hardware. In Bimal K. Roy and Willi Meier, editors, *FSE*, volume 3017 of *Lecture Notes in Computer Science*, pages 279–299. Springer, 2004.".

[18] Vincent Rijmen and Paulo Barreto., "The KHAZAD block cipher. 2000.".

[19] Chae Hoon Lim., "A revised version of Crypton - Crypton v1.0. In Lars R. Knudsen, editor, *FSE*, volume 1636 of *Lecture Notes in Computer Science*, pages 31–45. Springer, 1999.".

[20] William Stallings., "The Whirlpool secure hash function. *Cryptologia*, 30(1):55–67, 2006.".

[21] Léo Perrin, Aleksei Udovenko, and Alex Biryukov., "Cryptanalysis of a theorem: Decomposing the only known solution to the big APN problem (full version). *IACR Cryptology ePrint Archive*, 2016:539, 2016. `http://eprint.iacr.org/2016/539`.".

[22] K.A. Browning, J.F. Dillon, M.T. McQuistan, and A.J. Wolfe., "An APN permutation in dimension six. 518, 01 2010.".

[23] Alex Biryukov, Léo Perrin, and Aleksei Udovenko., "Reverse-engineering the S-Box of Streebog, Kuznyechik and STRIBOBr1., 2016. `http://eprint.iacr.org/2016/071`.".

[24] F.M/ Malyshev, A.E. Trishin., "Linear and differential cryptoanalysis: Another viewpoint. pages 42–45, 2018.".

[25] Anne Canteaut and Léo Perrin., "On CCZ-equivalence, extended-affine equivalence, and function twisting. Cryptology ePrint Archive, Report 2018/713, 2018. `https://eprint.iacr.org/2018/713`.".

[26] Alex Biryukov, Léo Perrin, and Aleksei Udovenko., "Reverse-Engineering the S-Box of Streebog, Kuznyechik and STRIBOBr1. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT (1)*, volume 9665 of *Lecture Notes in Computer Science*, pages 372–402. Springer, 2016.".

[27] Léo Perrin., "*Cryptanalysis, Reverse-Engineering and Design of Symmetric Cryptographic Algorithms.* PhD thesis, University of Luxembourg, 2017.".

[28] Denis Fomin., "New classes of 8-bit permutations based on a butterfly structure. *Matematicheskie vopr. kriptogr.*, 10(4), 2019.".

[29] Reynier Antonio de la Cruz Jiménez., "Generation of 8-bit s-boxes having almost optimal cryptographic properties using smaller 4-bit s-boxes and finite field multiplication. `www.cs.haifa.ac.il/~orrd/LC17/paper60.pdf`.".

[30] Reynier Antonio de la Cruz Jiménez., "On some methods for constructing almost optimal s-boxes and their resilience against side-channel attacks. Cryptology ePrint Archive, Report 2018/618, 2018. `https://eprint.iacr.org/2018/618`.".

[31] Denis Fomin., "On the algebraic degree and differential uniformity of permutations o nthe space $V_{2m}$ constructed via $(2m, m)$-functions *Matematicheskie vopr. kriptogr.*, 11(4):133–149, 2020.".

[32] Alejandro Freyre-Echevarria., "On the generation of cryptographically strong substitution boxes from small ones and heuristic search. *10th Workshop on Current Trends in Cryptology (CTCrypt 2021). Pre-proceedings*, pages 112–128, 2021.".

[33] Claude Carlet, Deng Tang, Xiaohu Tang, and Qunying Liao., "New construction of differentially 4-uniform bijections. In Dongdai Lin, Shouhuai Xu, and Moti Yung, editors, *Information Security and Cryptology*, pages 22–38, Cham, 2014. Springer International Publishing.".

[34] R. Lidl, H. Niederreiter, P.M. Cohn, G.C. Rota, B. Doran, P. Flajolet, M. Ismail, T.Y. Lam, and E. Lutwak., "*Finite Fields*. Number vol. 20, part 1 in EBL-Schweitzer. Cambridge University Press, 1997.".

# Symmetric Cryptography Analysis

# Fast correlation attack for GRAIN-128AEAD with fault

Sergei Katishev and Maxime Malov

Federal State Budget Educational Institution of Higher Education
«MIREA – Russian Technological University», Russia
katyshev_s@mirea.ru ,maxy008@mail.ru

**Abstract**

In this paper we consider fast correlation attack (FCA) on stream cipher GRAIN-128AEAD assuming that there is one fault in algorithm. We use version of FCA described by Yosuke Todo, Takanori Isobe, Willi Meier, Kazumaro Aoki and Bin Zhang. As a result, we construct linear relations required for attack on GRAIN-128AEAD with fault and apply successful attack.

**Keywords:** Stream cipher, GRAIN-128AEAD, correlation attack

## 1 Introduction

GRAIN-128AEAD is a cipher of the GRAIN family of stream ciphers. Its specification is closely based on Grain-128a introduced in 2011, which has been analysed in the literature for several years (for example, [1, 2, 3, 4, 5] etc.). In 2015, NIST started a project to standardize lightweight cryptography. As part of this project, NIST launched a competition to select the most suitable stream cipher. The GRAIN-128AEAD cipher was developed to participate in the competition.

Grain-128AEAD consists of two main parts: a block responsible for generating a pseudo-random sequence (generation block), a block responsible for transmitted messages authentication (authentication block). Grain-128AEAD supports AEAD (Authenticated Encryption with Associated Data) encryption mode. First, we describe the structure of the generation block and the authentication block, and then the algorithms for initializing the initial state, generating a pseudo-random sequence, encryption and authentication.

**Generation block.** The generation block consists of a binary nonlinear feedback shift register (NFSR), a binary linear feedback shift register (LFSR)

and a combining function. The length of each register is 128 bits. Denote by

$$W^{(t)} = (w_0^{(t)}, \ldots, w_{127}^{(t)}),$$
$$U^{(t)} = (u_0^{(t)}, \ldots, u_{127}^{(t)})$$

states of NFSR and LFSR at time $t \geq 0$. Thus, the state of the generation block at time $t$ is a pair $(W^{(t)}, U^{(t)})$, which corresponds to a binary vector of 256 bits in length.

Recurrence relation for LFSR is:

$$u_{127}^{(t+1)} = u_0^{(t)} \oplus u_7^{(t)} \oplus u_{38}^{(t)} \oplus u_{70}^{(t)} \oplus u_{81}^{(t)} \oplus u_{96}^{(t)} = \Sigma_u(U^{(t)}).$$

Recurrence relation for NFSR is:

$$\begin{aligned} w_{127}^{(t+1)} = u_0^{(t)} \oplus w_0^{(t)} \oplus w_{26}^{(t)} \oplus w_{56}^{(t)} \oplus w_{91}^{(t)} \oplus w_{96}^{(t)} \oplus w_3^{(t)} w_{67}^{(t)} \oplus w_{11}^{(t)} w_{13}^{(t)} \\ \oplus w_{17}^{(t)} w_{18}^{(t)} \oplus w_{27}^{(t)} w_{59}^{(t)} \oplus w_{40}^{(t)} w_{48}^{(t)} \oplus w_{61}^{(t)} w_{65}^{(t)} \oplus w_{68}^{(t)} w_{84}^{(t)} \\ \oplus w_{22}^{(t)} w_{24}^{(t)} w_{25}^{(t)} \oplus w_{70}^{(t)} w_{78}^{(t)} w_{82}^{(t)} \oplus w_{88}^{(t)} w_{92}^{(t)} w_{93}^{(t)} w_{95}^{(t)} = u_0^{(t)} \oplus \Sigma_w(W^{(t)}). \end{aligned}$$

The output sequence $\vec{y}^{(t)}$ is formed by:

$$\begin{aligned} y^{(t)} &= F(W^{(t)}, U^{(t)}) = \\ &= h(w_{12}^{(t)}, u_8^{(t)}, u_{13}^{(t)}, u_{20}^{(t)}, w_{95}^{(t)}, u_{42}^{(t)}, u_{60}^{(t)}, u_{79}^{(t)}, u_{94}^{(t)}) \oplus u_{93}^{(t)} \oplus \sum_{j \in A} w_j^{(t)} = \\ &= w_{12}^{(t)} u_8^{(t)} \oplus u_{13}^{(t)} u_{20}^{(t)} \oplus w_{95}^{(t)} u_{42}^{(t)} \oplus u_{60}^{(t)} u_{79}^{(t)} u_{94}^{(t)} \oplus u_{93}^{(t)} \oplus \sum_{j \in A} w_j^{(t)}, \end{aligned}$$

where $A = \{2, 15, 36, 45, 64, 73, 89\}$, $t = 0, 1, \ldots$.

**Authentication block** consists of two binary registers: a memory register (MR), a shift register (SR). The length of both registers is 64 bits. The state of the authentication block is set by initializing registers. The binary vector $(x_1^{(t)}, x_2^{(t)})$ is fed to the input of the authentication block. Let's introduce the notation

$$A^{(t)} = (a_0^{(t)}, \ldots, a_{63}^{(t)}),$$
$$R^{(t)} = (r_0^{(t)}, \ldots, r_{63}^{(t)})$$

— the MR and SR states at time $t \geq 0$. The new state is generated according to the following rule:

$$H((A^{(t)}, R^{(t)}), (x_1^{(t)}, x_2^{(t)})) = ((a_0^{(t)} \oplus x_2^{(t)} r_0^{(t)}, \ldots, a_{63}^{(t)} \oplus x_2^{(t)} r_{63}^{(t)}), (r_1^{(t)}, \ldots, r_{63}^{(t)}, x_1^{(t)})).$$

**Initialization** of the initial state is carried out as follows. The key $k = (k^{(0)}, \ldots, k^{(127)})$ and initialization vector $IV = (iv_0, ..., iv_{95})$ are loaded into a linear register:

$$(w_0^{(0)}, \ldots, w_{127}^{(0)}) = (k^{(0)}, \ldots, k^{(127)}),$$
$$(u_0^{(0)}, \ldots, u_{127}^{(0)}) = (iv_0, ..., iv_{95}, 1, \ldots, 1, 0).$$

Then the algorithm runs for 384 cycles. During the first 256 cycles the output $y^{(t)}$ is fed to the input to the LFSR and NFSR:

$$u_{127}^{(t+1)} = \Sigma_u(U^{(t)}) \oplus y^{(t)},$$
$$w_{127}^{(t+1)} = u_0^{(t)} \oplus \Sigma_w(W^{(t)}) \oplus y^{(t)}.$$

During the remaining 128 clock cycles the NFSR works unchanged, and the key bits are fed to the input to the LFSR while the output $y^{(t)}$ is written to the MR and SR:

$$u_{127}^{(t+1)} = \Sigma_u(U^{(t)}) \oplus k^{(t-256)}, 256 \leq t \leq 383 \quad (1)$$
$$(a_0^{(0)}, \ldots, a_{63}^{(0)}) = (y_{256}, \ldots, y_{319}),$$
$$(r_0^{(0)}, \ldots, r_{63}^{(0)}) = (y_{320}, \ldots, y_{383}).$$

**Encryption** is performed by bitwise addition of the plaintext $m^{(t)}||1$ with the sign of the generated keystream $z^{(t)}||z^{(t+1)}$. One round of encryption corresponds to two cycles of the algorithm in which two bits $y^{(2t)}$ and $y^{(2t+1)}$ are generated. Each even bit is used as $z^{(t)}$, and the pair $(y^{(2t+1)}, m^{(t)})$ is applied as input to the authentication block. After encrypting the last bit the content of the MR is used in MAC.

## 2 Fast Correlation Attack

There are many approaches for analysing the strength of encryption algorithms. One such approach is to analyse a simplified cipher scheme. Usually in such cases, several standard scenarios are corresponding to:

– reduced number of rounds at the initialization stage;

– reduced set of initial states or a set of key choices;

– the presence of faults in the algorithm.

Since the main area of application of the Grain-128AEAD algorithm is so-called "Internet of Things", which implies the integration of the cipher into

electronic devices, we will start from the latter scenario. In this paper we will use the main ideas from the attacks presented in the paper [4] considering one fault in the algorithm.

## 2.1 (Fast) Correlation method of cryptographic analysis

*Correlation method of cryptographic analysis (correlation attack)* is usually associated with studies of so-called combining generators of pseudorandom sequences built on the shift registers and Boolean functions.

The method can be applied if it is possible to construct a linear relation between the bits of one of the linear shift registers (for example, first one with initial state $\vec{s}^{(0)} = (s_0^{(0)}, ..., s_{n-1}^{(0)})$ and outputs $\vec{y}^{(0)} = (y^{(0)}, ..., y^{(k)})$) performed with a probability other than $\frac{1}{2}$:

$$P\{< \vec{s}^{(0)}, L_s >=< \vec{y}^{(0)}, L_y >\} = q \neq \frac{1}{2},$$

where $L_s$, $L_y$ – some binary vectors.

Obviously, such a relation can be continued for the entire length of the known output sequence:

$$P\{< \vec{s}^{(0)} \cdot S^t(f), L_s >=< \vec{y}^{(t)}, L_y >\} = q \neq \frac{1}{2},$$

where $S(f)$ – the matrix for the characteristic polynomial $f$ and $\vec{y}^{(t)} = (y^{(t)}, ..., y^{(t+k)})$.

Using these relations (assuming that the output sequence does not depend on the initial state of the register) it is possible to construct a statistical criterion for rejecting the considering initial states of the first register. The criterion (when $q > \frac{1}{2}$) can be represented as

$$\sum_{t=0}^{N-1} (-1)^{<\vec{s}^{(0)} \cdot S^t(f), L_s> \oplus <\vec{y}^{(t)}, L_y>} > C,$$

for some constant $C$. A rough estimation of the amount of output needed in this case is $N \approx \frac{1}{(1/2-q)^2}$. Thus, with the complexity of $N2^n$, we will find the true initial state.

## 2.2 The main idea of FCA

It is easy to see that the statistic used in the criteria is a function of the initial state of the register:

$$\nu(\vec{s}^{(0)}) = \sum_{t=0}^{N-1} (-1)^{<\vec{s}^{(0)} \cdot S^t(f), L_s> \oplus <\vec{y}^{(t)}, L_y>}.$$

Let $\{0,1\}^n = \{(x_1, \cdots, x_n) | x_i \in \{0,1\}\}$. In [4] authors show that it can be represented as:

$$\nu(\vec{s}^{(0)}) = \sum_{t=0}^{N-1} (-1)^{<\vec{s}^{(0)}, L_s \cdot (S^t(f))^T> \oplus <\vec{y}^{(t)}, L_y>} =$$

$$= \sum_{\vec{x} \in \{0,1\}^n} \left( \sum_{t \in \{0,\ldots,N-1 | L_s \cdot (S^t(f))^T = \vec{x}\}} (-1)^{<\vec{s}^{(0)}, \vec{x}> \oplus <\vec{y}^{(t)}, L_y>} \right) =$$

$$= \sum_{\vec{x} \in \{0,1\}^n} \left( \sum_{t \in \{0,\ldots,N-1 | L_s \cdot (S^t(f))^T = \vec{x}\}} (-1)^{<\vec{y}^{(t)}, L_y>} \right) (-1)^{<\vec{s}^{(0)}, \vec{x}>}.$$

the values

$$\omega(\vec{x}) = \sum_{t \in \{0,\ldots,N-1 | L_s \cdot (S^t(f))^T = \vec{x}\}} (-1)^{<\vec{y}^{(t)}, L_y>} \tag{2}$$

are Walsh-Hadamard coefficients [4] of the function $\nu(\vec{s}^{(0)})$. All Walsh-Hadamard coefficients $\omega(\vec{x})$, $\vec{x} \in \{0,1\}^n$ can be found by $O(N)$ operations. Using a fast Walsh-Hadamard transform, one can find the values of $\nu(\vec{s}^{(0)})$ simultaneously for all initial fillings for $n2^n$ operations. So, it remains only to select the values that meet the criterion.

The authors of [4] propose a method for further reducing the complexity of the attack provided that $m$ relations are known. Then it is possible to fix a part of the $\beta$ bit of the vector $\vec{s}^{(0)}$ (for example, with zeros) and consider the function $\nu'(\vec{s}^{(0)})$ depending on $n - \beta$ variables. Then we will choose those $m$ relations in which the fixed bits do not participate. The complexity of the attack will decrease to $(n - \beta)2^{n-\beta}$ operations. To preserve the reliability of the attack it is necessary that $\beta \ll \log m$ (for a more accurate estimate see for example [4]).

Before outlining the basic algorithm let's consider property that we use for successfully caring out the correlation attack described in [4]. Consider the expression $L_s \cdot S^t(f)$. It is clear that the matrix $(S^t(f))^T$ is a matrix representation of element $\theta^t$ for all $t \in \mathbb{N}_0$ where $\theta$ is primitive element

of the field $GF(2^n)$. In this case the first row of this matrix is a vector representation of $\theta$. Denote by $\vec{\Theta}^{(t)}$ the first row of the matrix $(S^t(f))^T$. It is easy to see that $\vec{\Theta}^{(t)}$ is a vector representation of the element $\theta^t$. The vector $L_s$ can also be considered as an element of the $\gamma$ field $GF(2^n)$. Now using the matrix representation of the field elements $\gamma$ is represented as a matrix $M_\gamma^T = M_{L_s}^T = (S^t(f))^T$ where $d$ is some natural number. Then, taking into account the commutativity of the elements in the field we can obtain the following equality:

$$L_s \cdot (S^t(f))^T = \vec{\Theta}^{(t)} \cdot M_{L_s}^T.$$

So, instead of statistics

$$\sum_{t=0}^{N-1} (-1)^{<\vec{s}^{(0)}, L_s \cdot (S^t(f))^T> \oplus <\vec{y}^{(t)}, L_y>},$$

depending on the initial states we can consider statistics equal to

$$\nu(\vec{x}) = \sum_{t=0}^{N-1} (-1)^{<\vec{s}, \vec{\Theta}^{(t)}> \oplus <\vec{y}^{(t)}, L_y>} = \sum_{t=0}^{N-1} (-1)^{\hat{e}^{(t)}},$$

where $\vec{s} = \vec{s}^{(0)} \cdot M_{L_s}$. This allows us instead of counting statistics for each value of $L_s$ to calculate statistics once. Thus, the main tasks to implement the attack are:

- finding a suitable linear relation of $L_y$ elements of the output sequence;
- finding the optimal set of values $\mathbb{L}$ consisting of linear relations $L_s$ with a large correlation value.

There is no general method for solving these problems but the necessary constructions for the Grain-128AEAD cipher will be discussed in the next paragraph. Next, we will assume that we managed to build $L_y$ and find the set $\mathbb{L}$.

We introduce the basic assumption for the correlation attack. Let $\mathbb{L}$ consist of $D$ different vectors for which the correlation value between random binary variables $< \vec{s}, \vec{\Theta}^{(t)} >$ and $< \vec{y}^{(t)}, L_y >$ is large. Then we will observe that statistic 0 statistics $\sum_{t=0}^{N-1} (-1)^{\hat{e}^{(t)}}$ deviates significantly from 0 in the case when $\vec{s} = \vec{s}^{(0)} \cdot L_s, L_s \in \mathbb{L}$. For all other vectors $L_s$ we will assume that its value is close to 0. Denote by $\omega_{a_1,\ldots,a_k}^{i_1,\ldots,i_k}(\vec{x})$ Walsh-Hadamard coefficient $\omega(\vec{x})$ of the vector $\vec{x}$ which has the values $a_1, \ldots, a_k$ in the places with the numbers $i_1, \ldots, i_k$ respectively. Similarly by $\nu_{a_1,\ldots,a_k}^{i_1,\ldots,i_k}(\vec{s'})$ we denote the subfunction of the function $\nu(\vec{s})$.

**The main stages of the fast correlation attack.**

1. Fix $\beta$ coordinates of the vector $\vec{x}$ with numbers $i_1, \ldots, i_\beta$ (for example, $a_1 = 0, \ldots, a_\beta = 0$) and calculate the values of the Walsh-Hadamard coefficients for the subfunction $\nu^{i_1,\ldots,i_\beta}_{a_1,\ldots,a_\beta}(\vec{s'})$

$$\omega(\vec{x'}) = \frac{1}{2^\beta} \sum_{a_1,\ldots,a_\beta \in \{0,1\}} (-1)^{a_1+\cdots+a_\beta} \omega^{i_1,\ldots,i_\beta}_{a_1,\ldots,a_\beta}(\overrightarrow{x}).$$

   for each element $\overrightarrow{x}' \in \{0,1\}^{n-\beta}$. Since the value of $\omega(\overrightarrow{x}')$ can be found immediately without calculating the value of the coefficients $\omega(\overrightarrow{x})$ from (2) the complexity of this stage is $O(N)$ and it will require $O(N)$ bits of memory.

2. Using the fast Walsh-Hadamard algorithm (FWH) we find the values of the subfunction $\nu^{i_1,\ldots,i_k}_{a_1,\ldots,a_k}(\overrightarrow{s}')$ and compose a set $\mathbb{S}$ consisting of those binary vectors $\overrightarrow{s}'$ for which the inequality is fulfilled

$$\left| \frac{\nu^{i_1,\ldots,i_k}_{a_1,\ldots,a_k}(\overrightarrow{s}')}{N} \right| \geq C_1, C_1 \in \mathbb{R}.$$

   The overall complexity estimation of the algorithm is $O((n-\beta)2^{(n-\beta)})$.

3. Making a multiset $\hat{\mathbb{S}} = \{\overrightarrow{s} \cdot M^{-1}_{L_s} | \overrightarrow{s} \in \mathbb{S}, L_s \in \mathbb{L}_s\}$ and for each $\overrightarrow{\hat{s}} \in \hat{\mathbb{S}}$ we calculate the frequency of its occurrence $\hat{\nu}(\overrightarrow{\hat{s}})$ in $\hat{\mathbb{S}}$. To find the true initial filling we check the feasibility of the second criterion:

$$\hat{\nu}(\overrightarrow{\hat{u}}) \geq C_2, C_2 \in \mathbb{R}.$$

   Note that the distribution of the random variable $\hat{\nu}(\overrightarrow{\hat{u}})$ tends to a Poisson distribution.

To get the success estimation of the attack we will use the result of [4]

**Theorem 1.** *[4]. Let $n$ be the size of the LFSR in an LFSR-based stream cipher. We assume that there are $D$ linear masks whose absolute value of correlation is greater than $\delta$. When the size of bypassed bits is $\beta$, we can recover the initial state of the LFSR with time complexity $O(3(n-\beta)2^{(n-\beta)})$ and the required number of parity-check equations is $N = (n-\beta)2^{(n-\beta)}$, where the success probability is $\sum_{k=C_2}^{\infty} \frac{\lambda_2^k e^{-\lambda_2}}{k!}$, where $C_2$ is the minimum value satisfying*

$$\sum_{k=C_2}^{\infty} \frac{N2^{-n}e^{-(N2^{-n})}}{k!}.$$

*and*

$$\lambda_2 = \frac{D2^{-\beta}}{\sqrt{2\pi N}} \int\limits_{C_2}^{\infty} \exp\left(-\frac{(x - N\delta)^2}{2N}\right) dx,$$

$$C_2 = \sqrt{2N} \cdot erfc^{-1}\left(\frac{2(n-\beta)}{D}\right).$$

**Application of correlation attack to algorithms of the Grain family.** The structure of the combining algorithm generator of the Grain family turns out to be suitable for the implementation of the correlation method. Let's consider the features of using the method from [4] applied to the Grain-128a algorithm.

As already noted, the main idea is to construct some linear relationship between the initial state of the linear shift register and the output bits. The authors noticed that $< \vec{y}^{(t)}, L_y >$ is a linear combination of bits of the output sequence $y^{(t)}$ with numbers 0, 26, 56, 91, 96 and 128 (correspond to linear terms in the feedback law of the nonlinear shift register), which do not include any linear terms from the nonlinear shift register. So, it means that it can be approximated by a linear combination of bits of linear register only.

It is not difficult to see that it is not possible to find the best linear approximation of the function $< \vec{y}^{(t)}, L_y >$ from the shift register bits by brutforce. However, using various assumptions (about the non-occurrence of variables in a linear combination) the authors of the work have managed to find $49152 \times 64 \times 32 = 2^{26.58}$ linear relations with a probability of more than $\frac{1}{2} + 2^{-55,2381}$ ($cov = 2^{-54,2381}$).

The complexity of the Grain-128a attack was $2^{115.4}$ with $2^{113.8}$ known bits of the output sequence. Note that the direct implementation of the attack towards to Grain-128AEAD and Grain-128a in authentication mode does not pass since only even bits of the output sequence are used for encryption in these algorithms and no good linear relationships were built in this case.

# 3 Implementation of correlation attack to GRAIN-128AEAD

In the previous paragraph we have formulated the main stages of the correlation attack. In this paragraph we will find linear relations with a high correlation. Under the assuming that there is an error in the implementation of the algorithm.

The method of generating keystream signs in the Grain-128AEAD cipher allows you to resist the methods of correlation analysis since only bits with even numbers are fed to the output. We will consider the case when the value 0 is always supplied from one of the wire. In our assumption in recurrence relation of the nonlinear shift register the value removed from the 91st place is always zero. In other words now recurrence relation has transformed to:

$$w_{127}^{(t+1)} = u_0^{(t)} \oplus w_0^{(t)} \oplus w_{26}^{(t)} \oplus w_{56}^{(t)} \oplus w_{96}^{(t)} \oplus \cdots = u_0^{(t)} \oplus \hat{\Sigma}_w(W^{(t)}).$$

Here $\hat{\Sigma}_w(W^{(t)}) = \Sigma_w(W^{(t)}) \oplus w_9^{(t)}$1This assumption allows us to construct a correlation attack, given the fact that every odd output sequence is unknown to us. Now let's use the basic idea of constructing linear relations taken from the work [4].

Denote $h^{(t)} = h(w_{12}^{(t)}, u_8^{(t)}, u_{13}^{(t)}, u_{20}^{(t)}, w_{95}^{(t)}, u_{42}^{(t)}, u_{60}^{(t)}, u_{79}^{(t)}, u_{94}^{(t)})$. Consider the following set
$Z = \{0, 13, 28, 48, 64\}$ and consider the sum $\bigoplus_{i \in Z} z^{(t+i)}$.

$$\bigoplus_{i \in Z} z^{(t+i)} = \bigoplus_{i \in Z} y^{(t+2i)} = \bigoplus_{i \in Z} F(W^{(t+2i)}, U^{(t+2i)}) =$$

$$= \bigoplus_{i \in Z} \left( h^{(t+2i)} \oplus u_{93}^{(t+2i)} \oplus \bigoplus_{j \in A} w_j^{(t+2i)} \right) =$$

$$= \bigoplus_{i \in Z} \left( h^{(t+2i)} \oplus u_{93}^{(t+2i)} \right) \oplus \bigoplus_{i \in Z} \bigoplus_{j \in A} w_j^{(t+2i)}.$$

Let's change the order of summation in the second sum and get the following:

$$\bigoplus_{i \in Z} w_j^{(t+2i)} = w_j^{(t)} \oplus w_j^{(t+26)} \oplus w_j^{(t+56)} \oplus w_j^{(t+96)} \oplus w_j^{(t+128)}, j \in A.$$

It is easy to notice that by substituting an expression for the value of the element $w_j^{(t+128)}$ in accordance with the law of recursion, we obtain the following expression:

$$\bigoplus_{j \in A} \bigoplus_{i \in Z} w_j^{(t+2i)} = \bigoplus_{j \in A} \Bigg( u_j^{(t)} \oplus w_{3+j}^{(t)} w_{67+j}^{(t)} \oplus w_{11+j}^{(t)} w_{13+j}^{(t)} \oplus w_{17+j}^{(t)} w_{18+j}^{(t)} \oplus$$

$$\oplus w_{27+j}^{(t)} w_{59+j}^{(t)} \oplus w_{40+j}^{(t)} w_{48+j}^{(t)} \oplus w_{61+j}^{(t)} w_{65+j}^{(t)} \oplus w_{68+j}^{(t)} w_{84+j}^{(t)} \oplus w_{70+j}^{(t)} w_{78+j}^{(t)} w_{82+j}^{(t)} \oplus$$

$$\oplus w_{22+j}^{(t)} w_{24+j}^{(t)} w_{25+j}^{(t)} \oplus w_{88+j}^{(t)} w_{92+j}^{(t)} w_{93+j}^{(t)} w_{95+j}^{(t)} \Bigg) = \bigoplus_{j \in A} u_j^{(t)} \oplus g.$$

Here

$$g = \bigoplus_{j\in A}\Bigg( w_{3+j}^{(t)} w_{67+j}^{(t)} \oplus w_{11+j}^{(t)} w_{13+j}^{(t)} \oplus w_{17+j}^{(t)} w_{18+j}^{(t)} \oplus$$
$$\oplus w_{27+j}^{(t)} w_{59+j}^{(t)} \oplus w_{40+j}^{(t)} w_{48+j}^{(t)} \oplus w_{61+j}^{(t)} w_{65+j}^{(t)} \oplus w_{68+j}^{(t)} w_{84+j}^{(t)} \oplus w_{70+j}^{(t)} w_{78+j}^{(t)} w_{82+j}^{(t)} \oplus$$
$$\oplus w_{22+j}^{(t)} w_{24+j}^{(t)} w_{25+j}^{(t)} \oplus w_{88+j}^{(t)} w_{92+j}^{(t)} w_{93+j}^{(t)} w_{95+j}^{(t)} \Bigg).$$

Thus, the linear variables have been removed from recurrence relation of nonlinear register.

Now for each $i \in Z$, we will construct linear approximations for $h^{(t+2i)}$ and calculate correlation. Denote by $\vec{\alpha}^{(i)}, i \in Z$ a binary vector of length 9. The correlation between the function $h^{(t+2i)}$ and its linear approximation $< \vec{x}, \vec{\alpha}^{(i)} >$ (then just $\vec{\alpha}^{(i)}$) takes only three values $\delta^{(i)}(\vec{\alpha}^{(i)}) \in \{0, 2^{-4}, -2^{-4}\}$. Denote $\vec{\alpha} = (\vec{\alpha}^{(0)}||\vec{\alpha}^{(13)}||\vec{\alpha}^{(28)}||\vec{\alpha}^{(48)}||\vec{\alpha}^{(64)})$. Note that for all values of $i \in Z$, the correlation will be the same. Therefore, according to Pilling-up lemma the correlation $\delta(\vec{\alpha})$ between $\vec{\alpha}$ and $\bigoplus_{i\in Z} h^{(t+2i)}$ will be equal to $\prod_{i=1}^{6} \delta^{(i)}(\vec{\alpha}^{(i)})$. Thus, $\delta(\vec{\alpha}) \in \{0,2^{-20},-2^{-20}\}$. We have the following approximate expression for a linear combination of elements of the output sequence

$$\bigoplus_{i\in Z} z^{(t+i)} \approx < U^{(t)}, L_s > \oplus \bigoplus_{i\in Z} \big(\alpha_0^{(i)} w_{12}^{(t+2i)} \oplus \alpha_4^{(i)} w_{95}^{(t+2i)}\big) \oplus g.$$

Denote by $G$ the term $\bigoplus_{i\in Z} \big(\alpha_0^{(i)} w_{12}^{(t+2i)} \oplus \alpha_4^{(i)} w_{95}^{(t+2i)}\big) \oplus g$. The correlation value of $\delta_G(\vec{\alpha})$ depending on the different values of $\vec{\alpha}$ is given in the table 1.

The linear part of G:

$$\alpha_0^{(0)} w_{12}^{(t)} \oplus \alpha_4^{(0)} w_{95}^{(t)} \oplus \alpha_0^{(13)} w_{38}^{(t)} \oplus \alpha_4^{(13)} w_{121}^{(t)} \oplus \alpha_0^{(28)} w_{68}^{(t)} \oplus \alpha_4^{(28)} w_{151}^{(t)} \oplus$$
$$\oplus \alpha_0^{(48)} w_{108}^{(t)} \oplus \alpha_4^{(48)} w_{191}^{(t)} \oplus \alpha_0^{(64)} w_{140}^{(t)} \oplus \alpha_4^{(64)} w_{223}^{(t)}$$

Here we have used equation $u_{i+j}^{(t)} = u_i^{(t+j)}$.

Since the function $G$ does not depend on the variables $w_{12}^{(t)}, w_{38}^{(t)}, w_{68}^{(t)}, w_{191}^{(t)}, w_{223}^{(t)}$ then it is necessary to consider only those vectors $\alpha$ for which the values $\alpha_0^{(0)}$,$\alpha_0^{(13)}$ $\alpha_0^{(28)}$,$\alpha_4^{(48)}$ $\alpha_4^{(64)}$ equal to 0. Otherwise $\delta_G(\vec{\alpha}) = 0$.

Thus we have the following approximation

$$\bigoplus_{i\in Z} z^{(t+i)} \approx \bigoplus_{i\in Z} u_{93}^{(t+2i)} \oplus \bigoplus_{j\in A} u_j^{(t)} \oplus \bigoplus_{i\in Z} \Big\langle (\alpha_1^{(i)}, \alpha_2^{(i)}, \alpha_3^{(i)}), (u_8^{(t+2i)}, u_{13}^{(t+2i)}, u_{20}^{(t+2i)}) \Big\rangle$$

$$\oplus \bigoplus_{i \in Z} \Big\langle (\alpha_5^{(i)}, \alpha_6^{(i)}, \alpha_7^{(i)}, \alpha_8^{(i)}), (u_{42}^{(t+2i)}, u_{60}^{(t+2i)}, u_{79}^{(t+2i)}, u_{94}^{(t+2i)}) \Big\rangle.$$

At the same time , the overall correlation is

$$\Delta = -\delta_G(\vec{\alpha}) \cdot \delta(\vec{\alpha}).$$

Now let's move on to finding the set $\mathbb{L}$. Using the obtained results we will write out an expression for the necessary linear relations. Denote by $\vec{\Theta}^{(t)}$ the first row of the matrix $(S^t(f))^T$. Then linear masks $L_s$ will be:

$$L_s = \bigoplus_{i \in Z} \Bigg( \alpha_1^{(i)} \vec{\Theta}^{(2i+8)} \oplus \alpha_2^{(i)} \vec{\Theta}^{(2i+13)} \oplus \alpha_3^{(i)} \vec{\Theta}^{(2i+20)} \oplus \alpha_5^{(i)} \vec{\Theta}^{(2i+42)}$$

$$\oplus \alpha_6^{(i)} \vec{\Theta}^{(2i+60)} \oplus \alpha_7^{(i)} \vec{\Theta}^{(2i+79)} \oplus \alpha_8^{(i)} \vec{\Theta}^{(2i+94)} \oplus \vec{\Theta}^{(2i+93)} \Bigg) \oplus \bigoplus_{j \in A} \vec{\Theta}^{(j)}$$

Note that for $i_1 = 28$ and $i_2 = 48$ the values $\vec{\Theta}^{(2i_1+60)}$ and $\vec{\Theta}^{(2i_2+20)}$ are the same and equal to $\vec{\Theta}^{(116)}$. The vector $L_s$ takes the same value when $(\alpha_6^{(28)}, \alpha_3^{(48)}) = (0,0)$ and $(\alpha_6^{(28)}, \alpha_3^{(48)}) = (1,1)$ or when $(\alpha_6^{(28)}, \alpha_3^{(48)}) = (0,1)$ and $(\alpha_6^{(28)}, \alpha_3^{(48)}) = (1,0)$. Also note that since the linear approximation for LFSR does not depend on $\alpha_0^{(i)}, \alpha_4^{(i)}$ for all $i \in Z$ then for all possible values of these two elements the bias for $L_s$ can be summarized. Consider the linear span $\mathbb{V}$ generated by the set of all binary vectors of length 45 whose elements in the places with numbers $0, 4, 9, 13, 18, 22, 27, 31, 36, 40$ run through all possible values from $\{0,1\}$. Let $\vec{\lambda}$ be a binary vector of length 45 which has units in the places with numbers 24 and 30. Then

$$\delta(L_s) = \sum_{\vec{v} \in \mathbb{V}} \Bigg( (-1) \Big( \delta_G(\vec{\alpha} \oplus \vec{v}) \cdot \delta(\vec{\alpha} \oplus \vec{v}) + \delta_G(\vec{\alpha} \oplus \vec{v} \oplus \vec{\lambda}) \cdot \delta(\vec{\alpha} \oplus \vec{v} \oplus \vec{\lambda}) \Big) \Bigg).$$

Thus we have obtained the final expression for the correlation between the signs of the output sequence and the linear approximation $L_s$.

Table 1: Correlations

| $\overrightarrow{\alpha_4}^{(13)}$ | $\overrightarrow{\alpha_4}^{(28)}$ | $\overrightarrow{\alpha_0}^{(48)}$ | $\overrightarrow{\alpha_0}^{(64)}$ | $\delta_G(\vec{\alpha})$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | $-2^{-34.313}$ |
| 0 | 0 | 0 | 1 | $2^{-36.1875}$ |
| 0 | 0 | 1 | 0 | $-2^{-37.586}$ |
| 0 | 0 | 1 | 1 | $2^{-39.4605}$ |
| 0 | 1 | 0 | 0 | $-2^{-35.898}$ |
| 0 | 1 | 0 | 1 | $2^{-37.7724}$ |
| 0 | 1 | 1 | 0 | $-2^{-39.171}$ |
| 0 | 1 | 1 | 1 | $2^{-41.0454}$ |
| 1 | 0 | 0 | 0 | $-2^{-35.3636}$ |
| 1 | 0 | 0 | 1 | $2^{-37.2381}$ |
| 1 | 0 | 1 | 0 | $-2^{-38.171}$ |
| 1 | 0 | 1 | 1 | $2^{-40.0454}$ |
| 1 | 1 | 0 | 0 | $-2^{-36.9486}$ |
| 1 | 1 | 0 | 0 | $2^{-38.823}$ |
| 1 | 1 | 1 | 0 | $-2^{-39.7559}$ |
| 1 | 1 | 1 | 1 | $2^{-41.6304}$ |

By iterating over all possible values of $\vec{\alpha}$ not more than $2^{45}$ we can form the set $\mathbb{L}$ we need. However note that the value $\delta(\vec{\alpha})$ is not equal to 0 if and only if every multiplier of $\delta^{(i)}(\vec{\alpha}^{(i)})$ for $i \in Z$ is not zero. For each $\vec{\alpha}^{(i)}$ the value of $\delta^{(i)}(\vec{\alpha}^{(i)})$ is not zero in 64 cases. Thus possible values of $\vec{\alpha}$ are droped from $2^{45}$ to $2^{27}$ The set $\mathbb{L}$ has a cardinality of the order of $O(2^{24})$. At the same time for each $L_s \in \mathbb{L}$ the bias is not less than $2^{-54}$. Using Theorem 1 we get that the algorithm will restore the true initial state with a probability equal to 0.9 with $\beta = 20$ fixed bits while the total complexity is $O(2^{113})$. With $\beta = 21$ the probability of successful completion of the attack is approximately 0.8.

# References


[1] Maximov A., "Cryptanalysis of the "grain" family of stream ciphers", *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security*, ASIACCS '06, ACM, 2006, 283–288.

[2] Dinur I., Shamir A., "Breaking Grain-128 with dynamic cube attacks", *Fast Software Encryption*, Springer, 2011, 167–187.

[3] Ding L., Guan J., "Related key chosen IV attack on Grain-128a stream cipher", *IEEE Transactions on Information Forensics and Security*, 2013, 803–809.

[4] Todo Y., Isobe T., Meier W., Aoki K., Zhang B., "Fast correlation attack revisited", *Advances in Cryptolog—CRYPTO 2018*, Springer International Publishing, 2018, 129–159.

[5] Lehmann M. and Meier W., "Conditional differential cryptanalysis of Grain-128a", *Cryptology and Network Security*, 2012, 1–11.

# Distinguishing attacks on Feistel ciphers based on linear and differential attacks

Denis Fomin

HSE University, Moscow, Russia
dfomin@hse.ru

**Abstract**

This paper considers a generalization of the Feistel network. The approaches to the construction of distinguishers based on the differential and linear attacks in the general case for Abelian groups are considered. It is shown that in the case of non-uniform distribution on the set of round function values, it is possible to construct a distinguisher using a differential attack for a larger number of rounds compared to the equal-probability case. The linear characteristic for the linear attack for transformations on Abelian groups with an arbitrary distribution is defined for the first time. The known method of constructing distinguishers using a linear attack is generalized to the case of the considered generalized Feistel network. Some estimates on the characteristics of the proposed methods of distinguishing are obtained.

**Keywords:** Feistel, block cipher, linear attack, differential attack, distinguisher

## Introduction

Block ciphers nowadays play a key role in information security. One of the main ways to build block ciphers is to use a Feistel network. A large number of ciphers based on Feistel network, including format preserving encryption algorithms, are known [1, 2, 3, 4].

In this paper, we propose to consider some generalization of Feistel network for the case of Abelian groups. Also, we will propose a distinguishing attacks based on linear and differential attacks.

## 1 Generalized Tweakable Feistel Network

In this section a base construction of analysed block cipher is presented. This construction generalized several well-known tweakable block ciphers that are known to be standards for format-preserving encryption (FPE) in several countries FF1, FF3, FEA-1, FEA-3 [3, 4].

Generalised tweakable Feistel network (GTFN) is a block cipher based on unbalanced Feistel network [5]. Feistel is a well-known construction that transforming any function $F$ ($F$-function) into permutation.

Let $q \in \mathbb{N}$, $K \in \mathbb{Z}_q^k$ — key, $T \in \mathbb{Z}_q^t$ — tweak of GTFN. In this work we denote the encryption function of GTFN by $E_{K,T} \colon \mathbb{Z}_q^Q \to \mathbb{Z}_q^Q$, where $Q$ is a fixed natural number. The encryption algorithm is an iterative function and each iteration is called "round". It takes $I \in \mathbb{N}$ iterations to compute the ciphertext from the plaintext.

In favor of greater generality in this work $F$-function of GTFN is a key, tweak and round-dependent mapping:

$$F \colon \mathbb{Z}_q^k \times \mathbb{Z}_q^t \times \mathbb{Z}_I \times \mathbb{Z}_q^R \to \mathbb{Z}_q^L,$$

$L, R \in \mathbb{N}$, $L + R = Q$. It allows us to generalize a large number of round functions that dependent of the part of key vector $K$ and the part of tweak vector $T$ at certain iterations. Cryptographic properties of GTFN depends on properties of $F$-function. Let for some $h \in \mathbb{N}$, $h < Q$, $L = \lceil Q/h \rceil$, then $R = Q - \lceil Q/h \rceil$.

An internal state of $i$ round of GTFN is represented as follows: $S^{(i)} = S_0^{(i)} \| S_1^{(i)}$, where $S_0^{(i)} \in \mathbb{Z}_q^L$, $S_1^{(i)} \in \mathbb{Z}_q^R$, "$\|$" — is a concatenation symbol. $S^{(0)}$ is a plaintext, $S^{(I)}$ is a ciphertext. Then the round function is evaluated as follows:

$$S^{(i)} = S_1^{(i-1)} \left\| \left( S_0^{(i-1)} + F\left(K, T, i, R^{(i-1)}\right)\right) \right.,$$

where "+" — is either

- an operation of group $\mathbb{Z}_{q^L}$, that we will denote as $\boxplus$;
- or operator of vector space $\mathbb{Z}_q^L$, that we will denote as $\oplus$.

Let's consider two most interesting cases:

1. $q = 2$ and "+" operator in round function is $\oplus$ and call it $\mathrm{GTFN}_{\oplus}$;
2. $q \neq 2$ and "+" operator in round function is $\boxplus$ and call it $\mathrm{GTFN}_{\boxplus}$.

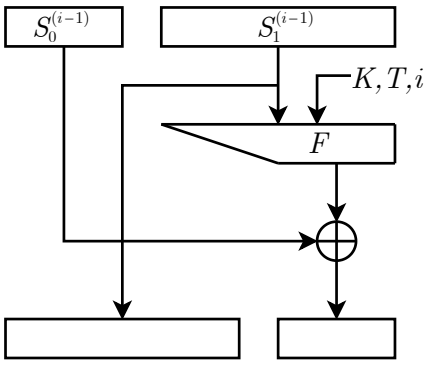



Figure 1: $\mathrm{GTFN}_{\oplus}$

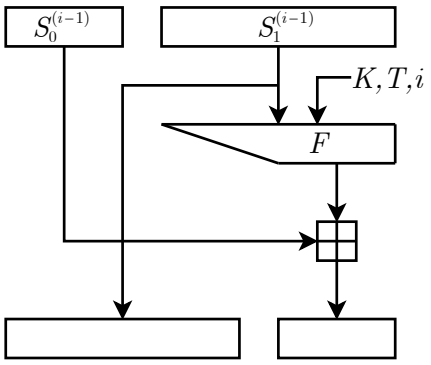



Figure 2: $\mathrm{GTFN}_{\boxplus}$

Figure 3: Variants of GTFN

In this work function $F$ for fixed $K, T$ realized a random function $\mathbb{Z}_q^R \to \mathbb{Z}_q^L$ according to the discrete distribution $\mathsf{D}$: $F(K, T, i, S_1^{(i-1)})$ is a realisation of a random variable with distribution $\mathsf{D}$. We will consider two kinds of distributions. The first one is an uniform discrete distribution $\mathsf{U}(\mathbb{Z}_q^L)$. The second one is a distribution of the following random variable:

$$\zeta = \xi \pmod{(q^L)}, \text{ where } \xi \sim \mathsf{U}\left(\mathbb{Z}_2^{\lceil L \cdot \log_2(q) \rceil}\right),$$

that we denote as $\mathsf{M}$. That means that:

$$\Pr\{\zeta = i\} = \begin{cases} 2 \cdot 2^{-2^{\lceil L \cdot \log_2(q) \rceil}}, & \text{where } i = 0, \ldots, 2^{\lceil L \cdot \log_2(q) \rceil} - q^L - 1 \\ 2^{-2^{\lceil L \cdot \log_2(q) \rceil}}, & \text{where } i = 2^{\lceil L \cdot \log_2(q) \rceil} - q^L - 1, \ldots, q^L - 1 \end{cases}.$$

This distribution appears, for example, in FF3-1, [3].

# 2 Differential Attack on GTFN

## 2.1 Differential trails

Let us first consider the algorithm GTFN with fixed tweak in the case $R = (h-1) \cdot L$. We can assume that GTFN is an unbalanced Feistel network whose internal state is represented as a concatenation of $h$ elements of $\mathbb{Z}_q$.

Let's consider the difference relations for $h$ rounds of the GTFN algorithm. With probability 1, the following difference relationship for $h$ rounds holds:

$$(\alpha\|\underbrace{0\|\ldots\|0}_{h-1}) \xrightarrow{1} (\underbrace{0\|\ldots\|0}_{h-1}\|\alpha) \xrightarrow{1}$$
$$\xrightarrow{1} (\underbrace{0\|\ldots\|0}_{h-2}\|\alpha\|\star) \xrightarrow{1} \cdots \xrightarrow{1} (\alpha\|\underbrace{\star\|\ldots\|\star}_{h-1}),$$

where above the arrows are written the probabilities of the corresponding difference relations, $\alpha \in \mathbb{Z}_q^L \backslash 0$, $\star$ — some (generally speaking different) elements of $\mathbb{Z}_q^L$. This property allows us to propose an efficient algorithm to distinguish $h$ rounds of the GTFN algorithm from a random substitution. Indeed, a similar difference relation for a random substitution must be performed with probability $q^{-L}$, which is less than 1. In this case, the required amount of material is estimated by the value $O\left(q^L\right)$.

The following difference relationship for $h+1$ rounds

$$(\alpha\|\underbrace{0\|\ldots\|0}_{h-1}) \xrightarrow{1} (\underbrace{0\|\ldots\|0}_{h-1}\|\alpha) \xrightarrow{q^{-L}}$$
$$\xrightarrow{q^{-L}} (\underbrace{0\|\ldots\|0}_{h-2}\|\alpha\|0) \xrightarrow{q^{-L}} \cdots$$
$$\cdots \xrightarrow{q^{-L}} (\alpha\|\underbrace{0\|\ldots\|0}_{h-1}) \xrightarrow{1} (\underbrace{0\|\ldots\|0}_{h-1}\|\alpha),$$

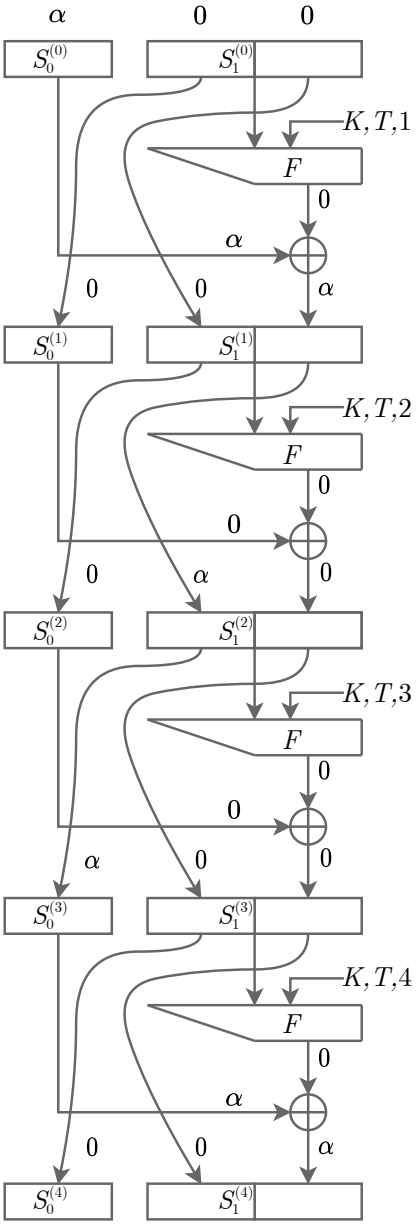



Figure 4: Differential trails for GTFN⊕ with $h = 3$

is performed with probability $q^{-(h-1)L}$, and for
a random permutation — with probability $q^{-hL}$.
Hence, in particular, it follows that with probability $2^{-2b_L}$ the following difference relation holds

$$(\alpha\|0\|0) \xrightarrow{1} (0\|0\|\alpha) \xrightarrow{2^{-b_L}} (0\|\alpha\|0) \xrightarrow{2^{-b_L}} (\alpha\|0\|0) \xrightarrow{1} (0\|0\|\alpha)$$

for 4 rounds of the $\mathrm{GTFN}_{\oplus}$ when $h = 3$ (see Figure 4).

## 2.2 Statistical issues

We can propose an algorithm to distinguish the $h+1$ rounds of the GTFN algorithm from random permutation. The idea of this algorithm is based on the statistical problem of distinguishing between two hypotheses:

- random sample observation from Bernoulli distribution with "success" probability equals to $q^{-(h-1)L}$;
- random sample observation from Bernoulli distribution with "success" probability equals to $q^{-(h)L}$.

Uniformly powerful test for these hypotheses problem based on Neyman–Pearson lemma. The difficulty of differental attack based on this test is about $O\left(q^{hL}\right)$.

Let on one tweak $t$ can be encrypted less or equal to $M$ different plain texts, and on one key can be processed less or equal to $T$ tweaks.

Let there are $M_j \leq M/2$ pairs of plain texts encrypted using $t_j$, $j = 1, \ldots, T$, tweak that have a difference $(\alpha\|0\| \ldots \|0)$ for some fixed $\alpha$. Then the statistics equivalent to the likelihood ratio statistics in this case looks as follows:

$$S_j(M_j) = \sum_{i=1}^{M_j} z_{i,j},$$

where $z_{i,j}$ is–an indicator that equals 1 if and only if $i$-th pair of plaintexts that has an input difference $(\alpha\|0\| \ldots \|0)$ is have the same difference between ciphertexts.

Simply increasing the material using different tweaks, values of $\alpha$ is generally speaking not correct. However, we can consider $S_j(M_j)$ at one tweak with fixed $\alpha$ as a random variable that has a binomial distribution with parameters $\mathrm{Bin}\,(M_j, q_i)$. In that case we can consider $N$ such observations ($N$ tweaks) and the statistic equivalent to the likelihood ratio statistic equals to:

$$K(N, M) = \sum_{j=1}^{N} S_j(M_j).$$

Considering different values of $\alpha$ also leads to an increase in the efficiency of the attack. Note that if the adversary has the ability to encrypt arbitrary texts, then he can choose texts in such a way as to obtain up to $M$ different values of $\alpha$ for which there will be about $M/2$ pairs of plaintexts for the chosen values of $\alpha$. Indeed, if the cryptoanalyst can encrypt $M = q^e$ plaintexts $(x_1, x_2, \ldots, x_h)$, where $x_1 \leq M$, the difference relations described above are fulfilled for any value of $\alpha \in \mathbb{Z}_q^e \backslash \{0\}$, $\alpha \leq M$. This potentially could increase the amount of material by a factor of $M/2$ (like in a multidimensional linear cryptanalysis [6, 7]).

### 2.3 Non-uniform case

Let's consider $\mathrm{GTFN}_{\boxplus}$ with round function that are chosen according $\mathsf{M}$ distribution. Let $N = 2^{\lceil L \log_2(q) \rceil}$, $N' = q^L$. The round function $F$ takes an arbitrary value from the set $A_0 = \{0, \ldots, N' - N - 1\}$ with probability $2/N'$, and a value from the set $A_1 = \{N' - N, \ldots, N - 1\}$ with probability $1/N'$.

We can consider the difference relation $F(x) + F(x + a) = b$ for this round function. When $a$ is fixed, the whole set $\mathbb{Z}_{q^R}$ is split into pairs of the form $(x, x + a)$. Let's find the probability that for a fixed pair, the sum of the function values on the corresponding arguments is equal to $b$. This is equivalent to the fact that two elements of the set $\{0, \ldots, N - 1\}$ are chosen at random according to the distribution $\mathsf{M}$. Denote $A_i + b = \{x + b | \, x \in A_i\}$. Find the intersection of $A_i$ and $A_j + b$, $i, j = 0, 1$:

$$W_{0,0} = |A_0 \cap (A_0 + b)| = \max\{N' - N - b, 0, N' - 2N + b, 2N' - 3N\}$$

$$W_{0,1} = |A_0 \cap (A_1 + b)| = \min\{2b, 2(N' - N), 2(N - b), 4N - 2N'\}$$

$$W_{1,1} = |A_1 \cap (A_1 + b)| = N - W_{0,0} - W_{0,1}.$$

Then the probability of the difference relation $F(x) + F(x + a) = b$ is equal to:

$$\Pr\{F(x) + F(x + a) = b\} = \frac{4W_{0,0}}{(N')^2} + \frac{2W_{0,1}}{(N')^2} + \frac{W_{1,1}}{(N')^2} = p_1(b).$$

The graph of this probability for the case $q = 10$, $h = 3$, $L = 3$ is shown in figure 5.

The last formula shows that as $N'$ increases, the probability of the difference relationship at fixed $x, a, b$ $F(x) + F(x + a) = b$ tends to $2^{N'}$.

This property helps to reduce the amount of material needed to apply the difference attack compared to the equal-probability case ($\mathsf{U}(\mathbb{Z}_{q^L})$).

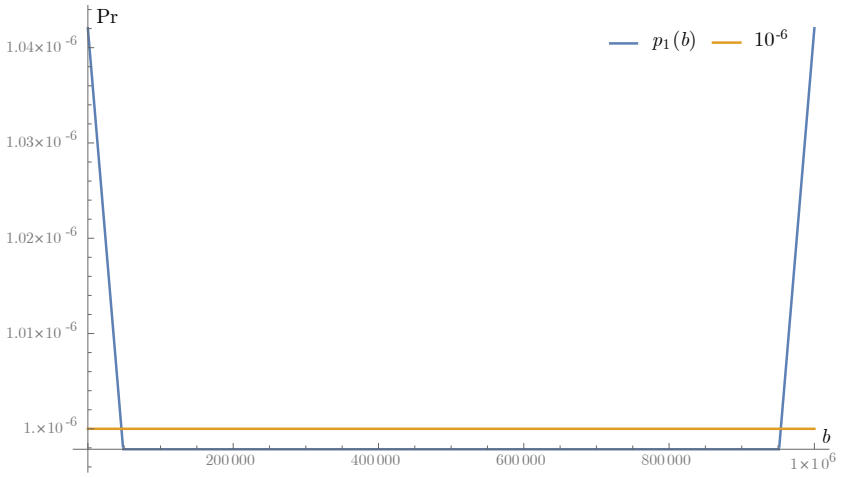



Figure 5: Graph of probability $p_1(b)$ in case $q = 10$, $h = 3$, $L = 3$

Moreover, it also allows to apply a difference attack for more rounds. Without losing generality, let us consider the special case of $\mathrm{GTFN}_{\boxplus}$ with $q = 10$, $h = 3$. Let's find the probability of the following $2h + 1$-rounds differential relation:

$$(\alpha \| 0 \| 0) \xrightarrow[2h+1 \text{ rounds}]{} (0 \| 0 \| \star),$$

where $\alpha \in \mathbb{Z}_{10^L}$ — a fixed value, $\star$ — any value of the set $\mathbb{Z}_{10^L}$. The differential above can be descripted as follows:

$$(\alpha \| 0 \| 0) \to (0 \| 0 \| \alpha) \to (0 \| \alpha \| \gamma) \to (\alpha \| \gamma \| \delta) \to (\gamma \| \delta \| \beta) \to \\ \to (\delta \| \beta \| 0) \to (\beta \| 0 \| 0) \to (0 \| 0 \| \beta),$$

where $\alpha \in \mathbb{Z}_{10^L}$ — a fixed value, $\beta, \gamma, \delta$ — some values of the set $\mathbb{Z}_{10^L}$. This differential relation is satisfied if

1. differential of function $F$ at the second round equals to differential of function $F$ a 5-th round;
2. differential of function $F$ at the third round equals to differential of function $F$ a 6-th round.

This probability is evaluated as follows:

$$\left( \sum_{b=0}^{10^L} p_1(b) \right)^2.$$

This probability is different from the case of an equal probability distribution:

$L = 3$; $10^{-6} + 1.61 \cdot 10^{-11}$ in case of $\mathsf{M}$ distribution vs $10^{-6}$ in equal-probability case;

$L = 4$; $10^{-8} + 4.01 \cdot 10^{-11}$ in case of $\mathsf{M}$ distribution vs $10^{-8}$ in equal-probability case;

$L = 5$; $10^{-10} + 7.24 \cdot 10^{-13}$ in case of $\mathsf{M}$ distribution vs $10^{-10}$ in equal-probability case;

$L = 6$; $10^{-12} + 1.17 \cdot 10^{-16}$ in case of $\mathsf{M}$ distribution vs $10^{-12}$ in equal-probability case.

### 2.4 $L \neq Q/h$ case

Previously, only case $L = Q/h$ was considered. It is easy to show that the proposed attack is applicable even in the case of $L \neq Q/h$, $L = \lceil Q/h \rceil$.

Let $\alpha \in \mathbb{Z}_q^L$, $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_L)$. Let $w = L - (Q - (h-1)L) = hL - Q$, $\alpha_1 = \ldots = \alpha_w = 0$. Then the following difference relationship for $h$ rounds holds:

$$(\alpha \| \underbrace{0 \| \ldots \| 0}_{h-1}) \xrightarrow[h \text{ rounds}]{1} (\alpha' \| \star)\,,$$

where $\alpha' = (\alpha_{w+1}, \alpha_{w+2}, \ldots, \alpha_L)$, $\star$ — some element of $\mathbb{Z}_q^{Q-L+w}$. As we can see in case $L = Q/h$ the value $w = 0$ and all statements shown earlier are correct.

## 3 Linear attack on GTFN

### 3.1 Mathematical background

In this section we presented some necessary mathematical results, which are a based on [8, 9, 10, 11].

Let $X$ be a finite Abelian group of order $n$. A character of $G$ is a homomorpism $\chi\colon X \to \mathbb{C}^\times$ of group $X$ to the multiplicative group of nonzero complex numbers $\mathbb{C}^\times$. The character $\chi_0(x) \equiv 1$ is the neutral character. The product of two characters is a character and the set of all characters of $X$ is a group $\widehat{X}$ of order $|X|$. The elements of $\widehat{X}$ can be enumerated by elements of $X$: $\widehat{X} = \{\chi_\alpha, \alpha \in X\}$. If $\chi$ and $\varphi$ are two characters of $X$, then

$$\sum_{x \in X} \chi(x) = \begin{cases} |X|, & \chi = \chi_0 \\ 0, & \text{otherwise} \end{cases}, \quad \sum_{x \in X} \chi(x) \cdot \overline{\varphi(x)} = \begin{cases} |X|, & \chi = \varphi \\ 0, & \text{otherwise} \end{cases};$$

$$\sum_{\alpha\in X}\chi_\alpha(x)=\begin{cases}|X|, & x=0\\ 0, & \text{otherwise}\end{cases},\ \sum_{\alpha\in X}\chi_\alpha(x)\cdot\overline{\chi_\alpha(y)}=\begin{cases}|X|, & x=y\\ 0, & \text{otherwise}\end{cases}.$$

Any Abelian group $X$ can be represented as a direct sum of cyclic groups:

$$X=G_1\dotplus G_2\dotplus\ldots\dotplus G_k,$$

where $G_i\cong\mathbb{Z}_{p_i^{n_i}}$, $p_i$ is a prime number, $n_i\in\mathbb{N}$, $i=1,2,\ldots,k$. The character of an arbitrary group $X$ is uniquely defined by the character assignment of groups $G_i$.

The scalar product of two functions $f_1$, $f_2$ with values in $\mathbb{C}^\times$ is defined as follows:

$$\langle f_1,f_2\rangle=\sum_{x\in X}f_1(x)\overline{f_2(x)}.$$

The Fourier coefficients of function $f\in\mathbb{C}^X$ is a function $C_\alpha^f\in\mathbb{C}^{\widehat{X}}$:

$$C_\alpha^f=\langle f,\overline{\chi_\alpha}\rangle=\sum_{x\in X}f(x)\overline{\chi_\alpha(x)},\ \alpha\in X.$$

These coefficients are defined the Fourier transform of $f$:

$$f=\frac{1}{|X|}\sum_{\alpha\in X}C_\alpha^f\chi_\alpha.$$

Let $X$, $Y$ are two finite Abelian groups and $f\in Y^X$. The group $Y$ is also represented as a direct sum of cyclic subgroups:

$$Y=H_1\dotplus H_2\dotplus\ldots\dotplus H_t,$$

where $H_j\cong\mathbb{Z}_{q_j^{m_j}}$, $q_j$ is a prime number, $m_j\in\mathbb{N}$, $j=1,2,\ldots,t$. And also a character of group $Y$ is uniquely defined by characters of $H_j$. And we can define the character $\psi_\beta(f)$, $\beta\in Y$, of function $f$:

$$\psi_\beta(f)=\frac{1}{|X|}\sum_{\alpha\in X}C_\alpha^{\psi_\beta(f)}\chi_\alpha.$$

For the simplicity we define $C_{\beta,\alpha}^f=C_\alpha^{\psi_\beta(f)}$, $\beta\in Y$, $\alpha\in X$:

$$C_{\beta,\alpha}^f=\sum_{x\in X}\psi_\beta(f(x))\overline{\chi_\alpha(x)}.$$

### 3.2 Probability function from the Fourier coefficients

Let $\mathsf{D}$ is a distribution of values of finite Abelian group $X$:

$$\Pr_{\mathsf{D}}\{x\} = p(x).$$

The function $p(x)$ can be represented using the Fourier transform as function of $\mathbb{C}^X$:

$$p(x) = \frac{1}{|X|}\sum_{\alpha\in X} C_\alpha^P \chi_\alpha(x).$$

Then $C_\alpha^p$ is the expected number of $\overline{\chi_\alpha}$:

$$C_\alpha^p = \sum_{x\in X} p(x)\overline{\chi_\alpha(x)} = \mathbf{E}\overline{\chi_\alpha}.$$

**Statement 1.** *Let $f \in Y^X$ be a function with arguments in finite Abelian group $X$ and with values in finite Abelian group $Y$. Then*

$$\mathbf{E}\psi_\beta(f(x)) = \frac{1}{|X|}\sum_{\alpha\in X} C_{\beta,\alpha}^f \cdot \mathbf{E}\chi_\alpha.$$

*Proof.*

$$\frac{1}{|X|}\sum_{\alpha\in X} C_{\beta,\alpha}^f \cdot \mathbf{E}\chi_\alpha = \frac{1}{|X|}\sum_{\alpha\in X}\left[\sum_{x\in X}\psi_\beta(f(x))\overline{\chi_\alpha(x)}\right]\cdot\sum_{x'\in X} p(x')\chi_\alpha(x') =$$

$$= \frac{1}{|X|}\sum_{x,x'\in X}\psi_\beta(f(x))p(x')\sum_{\alpha\in X}\overline{\chi_\alpha(x)}\chi_\alpha(x') =$$

$$= \sum_{x,x'\in X}\psi_\beta(f(x))p(x')\operatorname{Ind}(x=y) = \sum_{x\in X}\psi_\beta(f(x))p(x) = \mathbf{E}\psi_\beta(f(x)).$$

□

From this we obtain the following probability:

**Statement 2.** *Under the conditions of the previous statement:*

$$\Pr\{f(x)=b\} = \frac{1}{|Y|}\sum_{\beta\in Y}\mathbf{E}\psi_\beta(f)\overline{\psi_\beta(b)} = \frac{1}{|Y|}\sum_{\beta\in Y}\mathbf{E}\overline{\psi_\beta(f)}\psi_\beta(b).$$

*Proof.*

$$\frac{1}{|Y|}\sum_{\beta\in Y}\mathbf{E}\psi_\beta(f)\overline{\psi_\beta(b)} = \frac{1}{|Y|}\sum_{\beta\in Y}\left[\sum_{x\in X}\psi_\beta(f(x))p(x)\right]\overline{\psi_\beta(b)} =$$

$$= \frac{1}{|Y|}\sum_{x\in X}p(x)\sum_{\beta\in Y}\psi_\beta(f(x))\overline{\psi_\beta(b)} = \sum_{x\in X}p(x)\,\mathrm{Ind}(f(x)=b) = \Pr\{f(x)=b\}.$$

□

Let's consider the function $F(x)$ of the form $F(x) = (f(x), -x)$. In that case $F(x) \in (Y \dot{+} X)^X$. If $X$ and $Y$ are finite Abelian groups then $Z = Y \dot{+} X$ also a finite Abelian group and

$$Z = Y \dot{+} X = H_1 \dot{+} \ldots \dot{+} H_t \dot{+} G_1 \dot{+} \ldots \dot{+} G_k.$$

Let $\phi_\gamma$, $\gamma \in Z$, $\gamma = \beta\|\alpha$ — are characters of group $Z$. Then for function $F$:

$$\Pr\{F(x)=b\} = \frac{1}{|Z|}\sum_{\gamma\in Z}\mathbf{E}\phi_\gamma(F)\overline{\phi_\gamma(b)} =$$

$$= \sum_{\gamma\in Z}\mathbf{E}\left(\psi_\beta(f(x))\overline{\chi_\alpha(x)}\right)\overline{\psi_\beta(f(x))}\chi_\alpha(x).$$

We can see that $\mathbf{E}\left(\psi_\beta(f(x))\overline{\chi_\alpha(x)}\right)$ is a Fourier coefficient of function $F$ when $\mathsf{D} = \mathsf{U}$. In this work we call correlation coefficient of the linear approximation $(\chi_\alpha, \phi_\beta)$ of function $f$ the value

$$\mathbf{L}^F_{\beta,\alpha} = \mathbf{E}\left(\psi_\beta(f(x))\overline{\chi_\alpha(x)}\right).$$

If $Y$ and $X$ are the same groups the equation above can be rewritten as follows:

$$\mathbf{L}^F_{\beta,\alpha} = \mathbf{E}\left(\chi_\beta(f(x))\overline{\chi_\alpha(x)}\right).$$

### 3.3 Linear trails

As in the case of the differential trails let's consider the algorithm GTFN when $R = (h-1)\cdot L$. In this case, we can assume that GTFN is an unbalanced Feistel network whose internal state is represented as a concatenation of $h$ elements of $\mathbb{Z}_2^L$.

Consider the following linear relation on three rounds of the GTFN algorithm:

$$(\alpha\|\underbrace{0\|\ldots\|0}_{h-1}) \xrightarrow{c_1} (\underbrace{0\|\ldots\|0}_{h-1}\|\alpha) \xrightarrow{1}$$
$$\xrightarrow{1} (\underbrace{0\|\ldots\|0}_{h-2}\|\alpha\|0) \xrightarrow{1} \cdots \xrightarrow{1} (\alpha\|\underbrace{0\|\ldots\|0}_{h-1}),$$

where above the arrows are the values of the corresponding correlation coefficients.

Let's describe this relationship in more detail. The correlation coefficient $c_1$ in the first round

$$(\alpha\|\underbrace{0\|\ldots\|0}_{h-1}) \xrightarrow{c_1} (\underbrace{0\|\ldots\|0}_{h-1}\|\alpha)$$

equals to:

$$\mathbf{E}\left(\chi_\alpha\left(K,T,1,S_1^{(0)}\right)\overline{\chi_0\left(S_1^{(0)}\right)}\right) =$$
$$= \mathbf{E}\left(\chi_\alpha\left(K,T,1,S_1^{(0)}\right)\right),$$

where $F\left(K,T,1,S_1^{(0)}\right)$ — is $F$-function of the first round. In case of $\mathrm{GTFN}_\oplus$ algorithm this coefficient equals to:

$$2\cdot\Pr\left\{\left\langle 0,S_1^{(0)}\right\rangle = \left\langle\beta,F\left(b,1,T,S_1^{(0)}\right)\right\rangle\right\} - 1 = c_1.$$

Similarly we can consider the others correlation coefficients for the following relations:

$$(\underbrace{0\|\ldots\|0}_{h-1}\|\alpha) \xrightarrow{1} (\underbrace{0\|\ldots\|0}_{h-2}\|\alpha\|0),\ldots,(0\|\alpha\|\underbrace{0\|\ldots\|0}_{h-2}) \xrightarrow{1} (\alpha\|\underbrace{0\|\ldots\|0}_{h-1}).$$

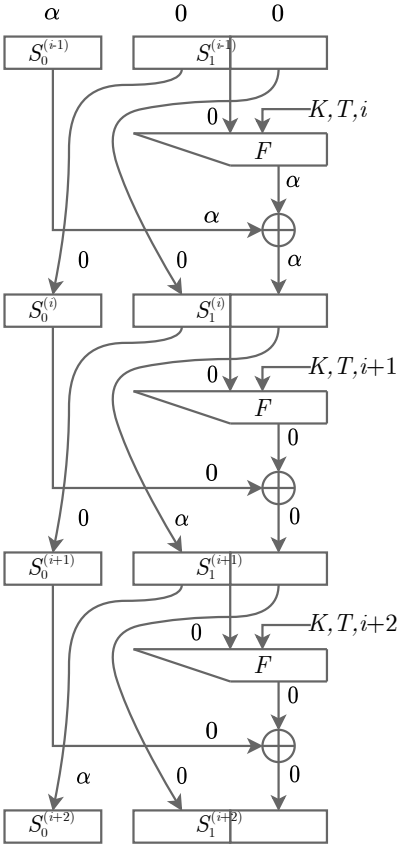



Figure 6: Linear trails for GTFN$\oplus$ with $h = 3$

It's easy to show, that

$$\mathbf{E}\left(\chi_0\left(F\left(K,T,i+1,S_1^{(i)}\right)\right)\overline{\chi_0\left(S_1^{(i)}\right)}\right)=1.$$

In case of $\mathrm{GTFN}_{\oplus}$ algorithm this coefficient equals to:

$$2\cdot\Pr\left\{\left\langle 0,S_1^{(i)}\right\rangle=\left\langle 0,F\left(K,T,i+1,S_1^{(i)}\right)\right\rangle\right\}-1=1.$$

Note that, as in [12], we can use the following approach. Let the set of plaintexts have the following form: $P=\{(x_1,x_2,\ldots,x_h)\}$, where $x_2,x_3,\ldots,x_h$ are fixed by some constants from the set $\mathbb{Z}_q^L$. Then for the first three rounds of the algorithm GTFN the absolute value of correlation coefficient is equal to 1. Indeed, on the first round, the values $F\left(K,T,1,S_1^{(0)}\right)$ will be the same and equal to some $y\in\mathbb{Z}_q^L$, from which it follows that

$$\left|\mathbf{E}\left(\chi_\alpha\left(F\left(K,T,1,S_1^{(0)}\right)\right)\overline{\chi_0\left(S_1^{(0)}\right)}\right)\right|=\left|\mathbf{E}\left(\chi_\alpha\left(y\right)\right)\right|=\left|\left(\chi_\alpha\left(y\right)\right)\right|=1.$$

In case of $\mathrm{GTFN}_{\oplus}$ algorithm this coefficient equals to:

$$2\cdot\Pr\left\{\left\langle 0,S_1^{(0)}\right\rangle=\langle\alpha,y\rangle\right\}-1=\pm 1.$$

It should be noted that when $L\neq Q/h$ all of the above will be performed similarly as in differential attack case.

### 3.4 Distribution of correlation coefficient $c_1$

Although the value of $c_1$ is unknown to us, we may know its distribution. If $\mathsf{D}=\mathsf{U}$ and $d=2$ this distribution is well studied:

**Theorem 1** ([13]). *For a random vectorial Boolean function $S\colon\mathbb{Z}_2^n\to\mathbb{Z}_2^m$, the value $\mathbf{L}_{\beta,\alpha}^S$ has the following distribution:*

$$\Pr\left\{\mathbf{L}_{\beta,\alpha}^S=i\right\}=2^{-2^n}\binom{2^n}{2^{n-1}+i}. \tag{1}$$

*And for a random permutation $S$ on the set $\mathbb{Z}_2^n$, the value $\mathbf{L}_{\beta,\alpha}^S$ has the following distribution:*

$$\Pr\left\{\mathbf{L}_{\beta,\alpha}^S=2i\right\}=\frac{\binom{2^{n-1}}{2^{n-2}+i}^2}{\binom{2^n}{2^{n-1}}}. \tag{2}$$

We can easily show that for a random vectorial Boolean function $S\colon \mathbb{Z}_2^n \to \mathbb{Z}_2^m$ as $n$ increases, the value $\mathbf{L}^S_{\beta,\alpha}$ will have a normal distribution with parameters $\mathcal{N}(0, 2^{-n})$. Using the De Moivre-Laplace theorem and the Stirling formula, we can show that the probability $\Pr\left\{\mathbf{L}^S_{\beta,\alpha} = 2i\right\}$ has the following approximation:

$$\Pr\left\{\mathbf{L}^S_{\beta,\alpha} = 2i\right\} = \frac{1}{\sqrt{2\pi}2^{n/2-2}} \cdot \exp\left\{-\frac{i^2}{2^{n-3}}\right\}.$$

The last equation is nothing but the density of a normal distribution. It follows, in particular, that as $n$ increases, the value $\mathbf{L}^S_{\beta,\alpha}$ will have a normal distribution with parameters $\mathcal{N}(0, 2^{-n})$ (just like a random function).

If $X$ and $Y$ are finite Abelian groups and $S$ is a random function $S \in Y^X$ the value $\mathbf{L}^S_{\beta,\alpha}$ has mean equals to 0 and variance $|X|^{-1}$. As $|X| \to \int$ the distribution of $\sqrt{|X|}\mathbf{L}^S_{\beta,\alpha}$ converges to the standard complex normal distribution $\mathcal{CN}(0, 1)$, [12].

If $\mathsf{D} \neq \mathsf{U}$ then the distribution of the value

$$\mathbf{L}^S_{\alpha,0} = \mathbf{E}\left(\chi_\alpha\left(K, T, 1, S_1^{(0)}\right)\right)$$

should be estimated.

### 3.5 Statistical issues

Let the set of plaintexts $A$ consist of elements of the form: $(x\|a_1\|a_2\|\ldots\|a_{h-1})$, where $a_0, a_1, \ldots, a_{h-1}$ — some fixed elements of $\mathbb{Z}_q^L$ for some $L \in \mathbb{N}$, and the set of values $x$ is some subset $\mathbb{Z}_q^L$. Consider the following linear relation on $h \cdot r + h$ rounds of the GTFN algorithm, similar to those considered in [12]:

$$(\alpha\|\underbrace{0\|\ldots\|0}_{h-1}) \xrightarrow{1} (\underbrace{0\|\ldots\|0}_{h-1}\|\alpha) \xrightarrow{1} (\underbrace{0\|\ldots\|0}_{h-2}\|\alpha\|0) \xrightarrow{1} \cdots$$
$$\cdots \xrightarrow{1} (\alpha\|\underbrace{0\|\ldots\|0}_{h-1}) \xrightarrow{c_{h+1}} (\underbrace{0\|\ldots\|0}_{h-1}\|\alpha) \xrightarrow{1} \ldots \xrightarrow{1} (\alpha\|\underbrace{0\|\ldots\|0}_{h-1}) \xrightarrow{c_{2h+1}}$$
$$\xrightarrow{c_{2h+1}} \ldots \xrightarrow{c_{r-h+1}} (\underbrace{0\|\ldots\|0}_{h-1}\|\alpha) \xrightarrow{1} \ldots \xrightarrow{1} (\alpha\|\underbrace{0\|\ldots\|0}_{h-1}).$$

Using the pilling-up lemma the correlation coefficient $\mathcal{C}_1 = \mathbf{L}^{\mathrm{GTFN}}_{(\alpha\|0\|\ldots\|0),(\alpha\|0\|\ldots\|0)}$ can be estimated as follows:

$$\mathcal{C}_1 = \prod_{i=1}^{r/h-h} c_{1+h\cdot i},$$

where $c_{1+h\cdot i} = \mathbf{L}^F_{\alpha,0}$. Each of the values $c_{1+h\cdot i}$ is a realization of a random variable with a distribution given above, and is not known to the cryptanalyst. Whereas a random permutation will have a correlation coefficient equals to the value $\mathcal{C}_0$, which is a realization of a random variable with the uniform distribution. The distribution of $\mathcal{C}_0$ is well known and we also suppose that the distribution of $\mathcal{C}_1$ is also known to a cryptanalyst.

Let $x_1, x_2, \ldots, x_M$ — are plaintexts and $y_1, y_2, \ldots, y_M$, are corresponding ciphertexts, $x_i, y_i \in X$, $i = 1, 2, \ldots, M$. Then the logarithm of likelihood function of a dunction $S \in X^X$ is equal to:

$$\ln \prod_{i=1}^{M} \Pr\{F(x_i) = y_i\} = \sum_{i=1}^{M} \ln\left(\Pr\{F(x_i) = y_i\}\right) =$$
$$= \sum_{i=1}^{M} \ln\left(\frac{1}{|X|}\left(1 + \sum_{\alpha',\beta' \in X\setminus 0} \mathbf{L}^S_{\beta',\alpha'} \overline{\chi_{\beta'}(y_i)} \chi_{\alpha'}(x_i)\right)\right) =$$
$$= \sum_{i=1}^{M} \ln\left(1 + \sum_{\alpha',\beta' \in X\setminus 0} \mathbf{L}^S_{\beta',\alpha'} \overline{\chi_{\beta'}(y_i)} \chi_{\alpha'}(x_i)\right) - M \ln|X|.$$

Then the statistics based on logarithm of likelihood function is asymptotically equivalent to:

$$\sum_{\alpha',\beta' \in X\setminus 0} \mathbf{L}^S_{\beta',\alpha'} \sum_{i=1}^{M} \overline{\chi_{\beta'}(y_i)} \chi_{\alpha'}(x_i).$$

With $M \to \infty$ the sum $\sum_{i=1}^{M} \overline{\chi_{\beta'}(y_i)} \chi_{\alpha'}(x_i)$ converges to $\overline{\mathbf{L}^S_{\beta',\alpha'}}$, then

$$\sum_{\alpha',\beta' \in X\setminus 0} \mathbf{L}^S_{\beta',\alpha'} \sum_{i=1}^{M} \overline{\chi_{\beta'}(y_i)} \chi_{\alpha'}(x_i) \to M \sum_{\alpha',\beta' \in X\setminus 0} \left|\mathbf{L}^S_{\beta',\alpha'}\right|^2.$$

As we consider plaintexts of the form $(x\|a_1\|a_2\|\ldots\|a_{h-1})$, where $a_0, a_1, \ldots, a_{h-1}$ — some fixed elements of $\mathbb{Z}_q^L$ and $\alpha' = \beta'$ of the form $(\alpha\|0\|\ldots\|0)$ then the equation above is equal to:

$$M \sum_{\alpha \in \mathbb{Z}_q^L \setminus 0} \left|\mathbf{L}^S_{(\alpha\|0\|\ldots\|0),(\alpha\|0\|\ldots\|0)}\right|^2.$$

Let $\mathbf{D}\mathcal{C}_0$ is the variance of correlation coefficient of a random function and $\mathbf{D}\mathcal{C}_1$ is the variance of a correlation coefficient

$$\mathbf{D}\mathcal{C}_1 = \mathbf{L}^{\mathrm{GTFN}}_{(\alpha\|0\|\ldots\|0),(\alpha\|0\|\ldots\|0)} \approx \left(\mathbf{D}\mathbf{L}^F_{\alpha,0}\right)^{r/h-h}.$$

Then for a successful attack the ratio between $M$, $N$ (tweak and other plaitexts quantity) and $|X| = q^L$ should be:

$$M \cdot N \cdot |X| \approx O\left(\left(\mathbf{D}\mathcal{C}_1 - \mathbf{D}\mathcal{C}_0\right)^{-1}\right).$$

## Conclusion

This paper considers one generalization of Feistel networks. The transformations of the round function of the considered block cipher are performed by an Abelian group.

Distinguisher based on linear and difference attack is proposed. The proposed linear attack is based on the results of [12] and generalized it, but in this article the attack does not depend on the form of tweak.

## References


[1] National Institute of Standards and Technology., "Data encryption standard (DES).", FIPS Publication 46-3, October 1999..

[2] Vasily Dolmatov., "Gost 28147-89: Encryption, decryption, and message authentication code (mac) algorithms. *RFC*, 5830:1–19, March 2010.".

[3] M. Dworkin., "Recommendation for block cipher modes of operation: methods for format-preserving encryption. nist special publication 800 38gr1 (february 2019). `https://doi.org/10.6028/NIST.SP.800-38A`.".

[4] Jung-Keun Lee, Bonwook Koo, Dongyoung Roh, Woo-Hwan Kim, and Daesung Kwon., "Format-preserving encryption algorithms using families of tweakable blockciphers. In Jooyoung Lee and Jongsung Kim, editors, *ICISC*, volume 8949 of *Lecture Notes in Computer Science*, pages 132–159. Springer, 2014.".

[5] Bruce Schneier and John Kelsey., "Unbalanced feistel networks and block cipher design. In Dieter Gollmann, editor, *FSE*, volume 1039 of *Lecture Notes in Computer Science*, pages 121–144. Springer, 1996.".

[6] Miia Hermelin., "*Multidimensional linear cryptanalysis.* PhD thesis, Aalto University, Espoo, Helsinki, Finland, 2010. base-search.net (ftaaltouniv:oai:aaltodoc.aalto.fi:123456789/4802).".

[7] Miia Hermelin, Joo Yeon Cho, and Kaisa Nyberg., "Multidimensional linear cryptanalysis of reduced round serpent. In Yi Mu, Willy Susilo, and Jennifer Seberry, editors, *ACISP*, volume 5107 of *Lecture Notes in Computer Science*, pages 203–215. Springer, 2008.".

[8] Tim Beyne., "Block cipher invariants as eigenvectors of correlation matrices. *J. Cryptol.*, 33(3):1156–1183, 2020.".

[9] Thomas Baignères, Pascal Junod, and Serge Vaudenay., "How far can we go beyond linear cryptanalysis? In Pil Joong Lee, editor, *ASIACRYPT*, volume 3329 of *Lecture Notes in Computer Science*, pages 432–450. Springer, 2004.".

[10] Ali Aydin Selçuk and Ali Biçak., "On probability of success in linear and differential cryptanalysis. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *SCN*, volume 2576 of *Lecture Notes in Computer Science*, pages 174–185. Springer, 2002.".

[11] Thomas Baignères, Jacques Stern, and Serge Vaudenay., "Linear cryptanalysis of non binary ciphers. In Carlisle M. Adams, Ali Miri, and Michael J. Wiener, editors, *Selected Areas in Cryptography*, volume 4876 of *Lecture Notes in Computer Science*, pages 184–211. Springer, 2007.".

[12] Tim Beyne., "Linear Cryptanalysis of FF3-1 and FEA. Cryptology ePrint Archive, Report 2021/815, 2021. `https://ia.cr/2021/815`.".

[13] Joan Daemen and Vincent Rijmen., "Probability distributions of correlation and differentials in block ciphers. *J. Math. Cryptol.*, 1(3):221–242, 2007.".

# The PRF pCollapserARX optimal cryptographic characteristic automated search by CASCADA

Sergey Polikarpov, Vadim Prudnikov, and Konstantin Rumyantsev

Southern Federal University, Russian Federation
polikarpovsv@sfedu.ru, prudnikov@sfedu.ru, rumyancev@sfedu.ru

**Abstract**

In this work we presents the results of the search for optimal cryptographic characteristics for pCollapserARX32-4x2, pCollapserARX64-8x2 and pCollapserARX128-16x2, belonging to the same family of ARX-based PRFs.

For this we actively used CASCADA – an open-source Python 3 library to evaluate the security of cryptographic primitives, specially block ciphers, against distinguishing attacks with bit-vector SMT solvers.

The results of the search for optimal cryptographic characteristics using CASCADA showed the resistance of the first 2 rounds (out of 4) of the PRF pCollapserARX to building distinguishers for attacks such as linear, differential, Related-Key differential, RX differential and Related-Key differential cryptanalysis.

Considering that the 1st round does not work in dynamic mode, we believe that the main contribution to resistance comes from the 2nd round, which works in dynamic mode. This confirms the correctness of both the use of pseudo-dynamic substitutions as a nonlinear element and the use of weak ARX-functions in their composition.

**Keywords:** cryptanalysis; pseudo-random function; pseudo-dynamic sbox.

## 1 Introduction

The structure of the PRF "Collapser" (a black hole) was first presented at CTCrypt'2015 [1]. It consists of sequentially connected pseudo-dynamic substitution (*PD-sboxes*). This structure was a demonstrator of the possibility of using *PD-sboxes* in cryptographic transformations. In [2] the PRF "pCollapser" (parallel Collapser) was proposed, in which number of disadvantages of the "Collapser" were eliminated. In "pCollapser", all pseudo-dynamic substitution boxes *PD-sbox* work in parallel and independently of each other within one round.

The authors called a pseudo-dynamic substitution a structure of fixed substitutions, the behavior of which resembles the behavior of a dynamic substitution. A description of this structure can be found in [1].

A high-performance PRF pCollapserARX256-32x2 was presented at the RusCrypto'2022 conference accessible at (RusCrypto'2022: `https://www.ruscrypto.ru/resource/archive/rc2022/files/02_polikarpov_rumyantsev_prudnikov.pdf`) – software-oriented PRF based on *PD-sbox* functions, where each *PD-sbox* consist from 4 sboxARX functions (nonbijective functions that using only modulo Addition, Rotate and Xor operations).

The main purpose this PRF – is to be used as a high-performance PRF in modes where reversible round functions are not required: AEAD, CTR, Sponge and etc.

Performance of the PRF pCollapserARX256-32x2 using usuba compiler [3] and instruction set: AVX256 – 2.8 cycles/byte; AVX512 – 1.8 cycles/byte (on one core of the Intel Core i7-11700K processor).

The main idea behind the PRF pCollapserARX256-32x2 is the use of *PD-sbox* (pseudo-dynamic substitution box) as a non-linear element - a special function that allows you to radically change the properties of a group of nested functions [4, 5, 6, 7]. Initially, it was supposed to use fixed bijective sboxes as nested functions. But the research conducted by the authors showed that the use of ARX-functions with initially weak cryptographic properties is well suited as nested functions.

It was shown that combining 4 weak ARX functions in *PD-sbox* allows one to obtain properties of equivalent sboxes close to those of randomly generated sboxes of the same dimension.

This paper presents the results of the search for optimal cryptographic characteristics for pCollapserARX32-4x2, pCollapserARX64-8x2 and pCollapserARX128-16x2 – PRF belonging to the same family with the pCollapserARX256-32x2 PRF.

The obtained results confirm the correctness of the PRF structure "pCollapserARX" and the ideas embedded in it.

## 2 Short description of the PRF "pCollapserARX"

The pCollapserARX32-4x2, pCollapserARX64-8x2 and pCollapserARX128-16x2 PRFs under consideration are built on an identical principle:

1. Using 4 rounds of transformation.
2. Each round uses 16 ARX-functions in parallel.
3. All ARX-functions are grouped into four *PD-sboxes*.

4. To ensure the dynamic mode of *PD-sboxes* operation, input/output control state values are used and formed.

5. The master key initializes the internal state, round keys are not used (similar to stream ciphers).

6. The first round is preparatory, in it *PD-sboxes* work in static mode, but forms control values for the next round.

**Designations in the PRF name:**

pCollapserARX256-32x2:

p - parallel;

ARX - type of operations used;

256 - block size in bits ($L_{block}$);

32x2 - word size in bits ($L_{word}$) (and in/out size of the ARX functions);

32 - subword size in ARX function (use 32 bit ARX operations).

**Used params and vectors**.

$N_{words} = 4$ - number of input words in the block;

$N_{rows} = 4$ - number of ARX-functions in one *PD-sbox*.

$L_{block} = L_{word} \times 4$, bit - block size in bits;

$L_{key} = (L_{block})$; $(2 \times L_{block})$ or $(4 \times L_{block})$, bit - master-key size in bits;

Used vectors:

$m = \{m_0, m_1, m_2, m_3\}$ - vector of the input message;

$c = \{c_0, c_1, c_2, c_3\}$ - cipher-text vector;

$s = \{s_{00}, s_{01}, \ldots, s_{33}\}$ - control state vector;

$d = \{d_{00}, d_{01}, \ldots, d_{33}\}$ - round constants.

During the researching of cryptographic characteristics by CASCADA, the following changes were made to the pCollapserARX structure. These changes are intended to correct the shortcomings of the originally proposed version of pCollapser, including to increase the weights of linear and differential characteristics:

1. Internal control state size increased to $L_{control\_state} = N_{rows} \times N_{words} \times L_{word}$, bit

2. Changed the formation/updating of the control state.

3. Added "extended key" generation. The "extended key" includes the initial control state and round key 1 (for the first round). There are no round keys for rounds 2-4, instead an updatable control state is used.

Parameters of the PRF pCollapserARX family are given in table 1.

Table 1: Parameters of the PRF pCollapserARX family

| pCollapserARX: | | | | |
|---|---|---|---|---|
| params | 32-4x2 | 64-8x2 | 128-16x2 | 256-32x2 |
| $L_{block}$, bit | 32 | 64 | 128 | 256 |
| $L_{word}$, bit | 8 | 16 | 32 | 64 |
| $N_{rounds}$ | 4 | 4 | 4 | 4 |
| $N_{words}$ | 4 | 4 | 4 | 4 |
| $N_{rows}$ | 4 | 4 | 4 | 4 |
| $L_{key}$, bit | 32 | 64 | 128 | 256 |
| $L_{control\_state}$, bit | 128 | 256 | 512 | 1024 |

## 2.1 Structure of used ARX functions

The structure of ARX-functions was selected based on the provision of cryptographic properties (as part of *PD-sbox*) and ensuring optimal use of the capabilities of processors and hardware platforms. Table 2 shows the parameters of the ARX-functions for the PRF pCollapserARX128-16x2.

Table 2: Parameters of the ARX-functions for the PRF pCollapserARX128-16x2

| | t0 | t1 | t2 | t3 | t4 | t5 | t6 | t7 |
|---|---|---|---|---|---|---|---|---|
| funcARX0: | 4 | 8 | 8 | 4 | 4 | 8 | 0 | 0 |
| funcARX1: | 4 | 8 | 4 | 8 | 8 | 4 | 4 | 4 |
| funcARX2: | 8 | 4 | 4 | 8 | 4 | 8 | 8 | 8 |
| funcARX3: | 8 | 4 | 8 | 4 | 8 | 4 | 12 | 12 |

To obtain the parameters values of the ARX-functions, it is necessary to double the values from table 2. Similarly, the parameters of smaller versions are obtained, only the values of the parameters are proportionally reduced.

Figure 1 shows the structure of the used ARX-functions.

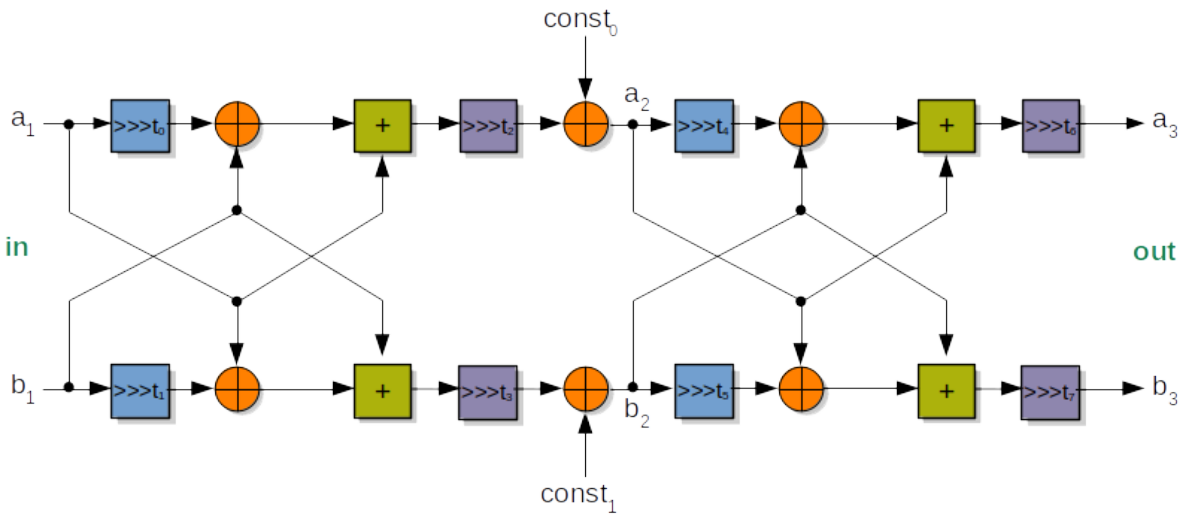



Figure 1: Structure of the used ARX-functions

## 2.2 Pseudo-dynamic substitution boxes function

Figure 2 shows the Pseudo-dynamic substitution box function, which consists of four parallel ARX functions. The *PD-Sbox* input-output dimension corresponds to the dimension of the used ARX functions.

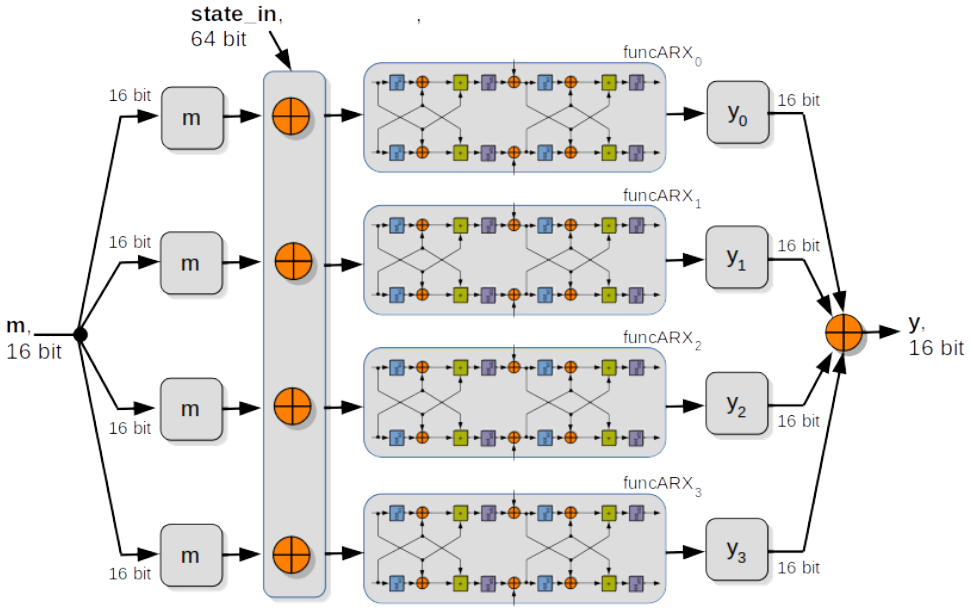



Figure 2: The Pseudo-dynamic substitution box function

Expression for base *PD-sbox* out:

$$c_i = \bigoplus_{j=0}^{3} funcARX_j(m_i \oplus s_j^i), \tag{1}$$

where: $i$ - index for n-bit word from input/output vector and, thereafter, index of *PD-sbox*; $j$ - index of *PD-sbox* component; $m_i$ - n-bit words from input vector; $c_i$ - n-bit words from output vector; $funcARX$ - ARX-function (components of *PD-sbox*); $s_j^i$ - n-bit words from control state input vector (individual for each *PD-sbox*).

Expressions for *PD-sbox* local (individual) control states output:

$$g_n^i = c_i \oplus funcARX_j(m_i \oplus s_j^i) = \bigoplus_{n=0,\, n\neq i}^{3} funcARX_j(m_i \oplus s_j^i). \qquad (2)$$

In the fact, with this expression we implement 4 sub *PD-sboxes* (with 3 ARX-function in each).

For a pseudo-dynamic substitution box *PD-sbox* was performed primary analysis of differential [4, 5] and linear [6] properties, proposed computationally efficient method for determining averaged distribution of differentials [8] and of linear properties [9], founded on a class of *PD-sbox* with a perfect averaged distribution of differentials in static mode of work [7].

It was shown in (`www.ruscrypto.ru/resource/archive/rc2022/files/02_polikarpov_rumyantsev_prudnikov.pdf`) that combining 4 weak ARX functions in *PD-sbox* allows one to obtain properties of equivalent sboxes close to those of randomly generated sboxes of the same dimension.

## 2.3 Round of the PRF pCollapserARX

Figure 3 shows the structure of one PRF pCollapserARX round, where:
$m = \{m_0, m_1, m_2, m_3\}$ - vector of the input message;
$c = \{c_0, c_1, c_2, c_3\}$ - cipher-text vector;
$s = \{s_{00}, s_{01}, \ldots, s_{33}\}$ - control state vector;
$d = \{d_{00}, d_{01}, \ldots, d_{33}\}$ - round constants;
funcARX0 ... funcARX3 - ARX-functions.

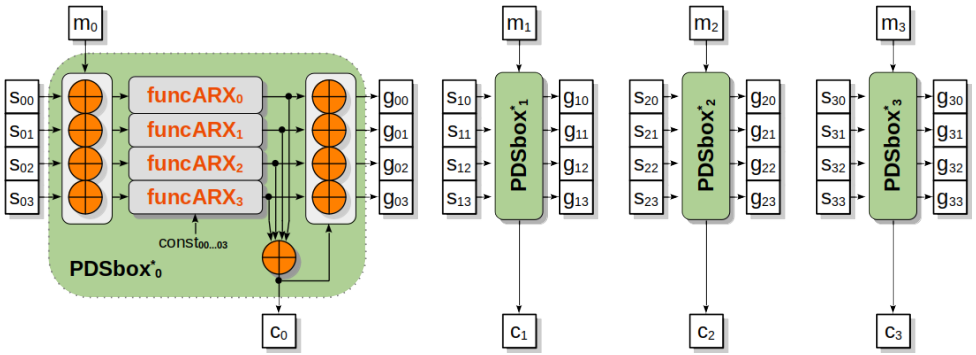



Figure 3: Structure of one round PRF pCollapserARX (without new control state generation elements)

At the end of each round, the control state is generated for subsequent rounds. It includes two steps:

1. Formation of the combined control state $s^* = \{s_0^*, s_1^*, s_2^*, s_3^*\}$ (figure 4)

2. Distribution of $s^*$ over individual control states $s = \{s_{00}, s_{01}, \ldots, s_{33}\}$ for each individual *PD-sbox* (figure 5)

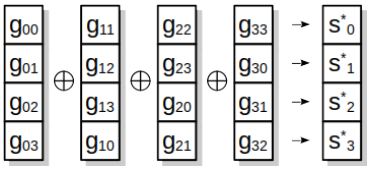



Figure 4: Formation of the output control state. Step 1.

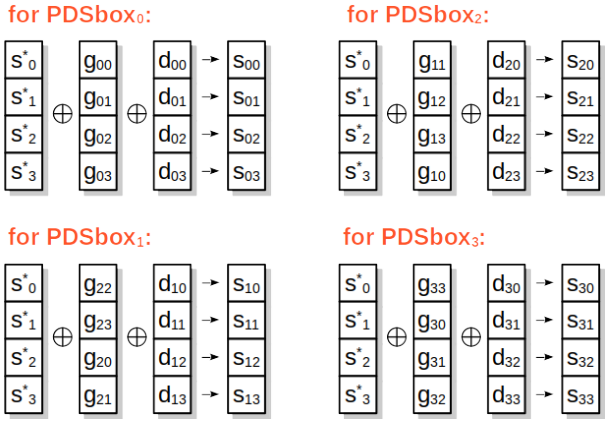



Figure 5: Formation of the output control state. Step 2.

Expression for function, that create new control state output vector (Figures 4 and 5):

Step 1:

$$g^i = (g_0^i, g_1^i, g_2^i, g_3^i) = g^i <<< (i \cdot Lword), \tag{3}$$

$$s^* = (s_0^*, s_1^*, s_2^*, s_3^*) = \bigoplus_{i=0}^{3} g^i. \tag{4}$$

Step 2:

$$s_j^i = s_j^* \oplus d_j^i \oplus g_j^i, \tag{5}$$

where: $g^i = (g_0^i, g_1^i, g_2^i, g_3^i)$ – out control state values from each i-th *PD-sbox*; $a <<< b$ – cyclic shift bits in a vector $a$ by $b$ elements in a left direction; $d_j^i$ – n-bit words from constant/decollision vector (individual for each *PD-sbox*).

## 2.4 Generating "expanded key"

Figure 6 (left part) shows "expanded-key" generation for the "pCollapserARX-64". This procedure consists in simply feeding the master key into the main input $m$ and performing 4 rounds of PRF pCollapserARX transformation. The initial value of the internal state $S_n$ is set to zero. The "expanded key" is the values of the output block on the third and fourth round.

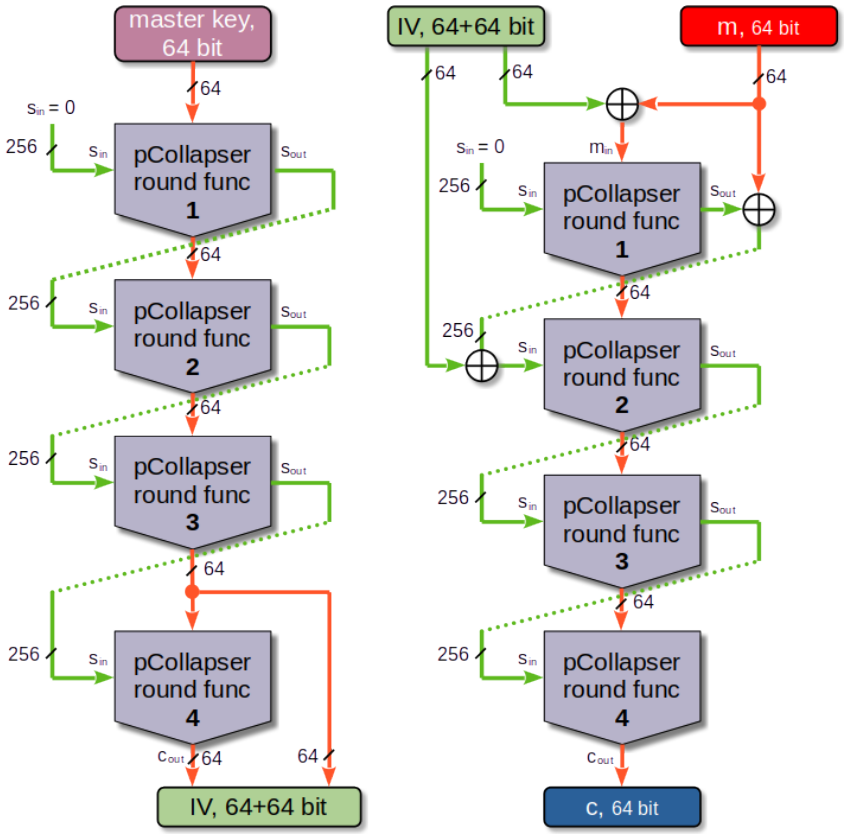



Figure 6: "Expanded-key" generation (left) and main rounds (right) for the "pCollapserARX64"

Figure 6 (right side) shows the pCollapserARX PRF structure using 4 rounds of transformation. The input plaintext is mixed with the "expanded key" part corresponding to the output of the third round of the key expansion procedure. To improve the dynamic mode of the second round, the input message is mixed with the second part of the "expanded key" and the values of the control input (In this case, 64 bit values are repeated 4 times to get 256 bits).

Before encryption an "expanded key" generation function need to used for preparing initial internal state of the "pCollapserLWC-64". For each plaintext block encryption same initial internal state used.

# 3 Searching cryptographic characteristics of the PRF "pCollapserARX"

## 3.1 Characteristics search tool

SAT/SMT solvers are an effective tool for automated search for the best (for cryptanalysis) characteristics of cryptographic functions and have been actively used in recent years [10, 11].

Ranea A. and Rijmen V. in [12] proposed a powerful tool called **CASCADA** - (Characteristic Automated Search of Cryptographic Algorithms for Distinguishing Attacks) is an open-source Python 3 library to evaluate the security of cryptographic primitives, specially block ciphers, against distinguishing attacks with bit-vector SMT solvers, accessible at (Github: `https://github.com/ranea/CASCADA`).

"The tool CASCADA implements the bit-vector property framework and several SMT-based automated search methods to evaluate the security of ciphers against differential, related-key differential, rotational-XOR, impossible-differential, impossible-rotational-XOR, related-key impossible-differential, linear and zero-correlation cryptanalysis" [12].

To search for characteristics using CASCADA, we created the file "**pCollapserARX_full.py**" containing the implementation of the PRF pCollapserARX and the file "**test_collapser.py**" that allows you to start searching for various cryptographic characteristics for the PRF pCollapserARX accessible at (Github: `https://github.com/pruvad/CTCrypt2023`).

The result of the CASCADA work is the weights of the found optimal characteristics:

$$w = -log_2 P(.),$$

where $P(.)$ is the probability of occurrence of the input/output difference for the tested function (for differential analysis) or the correlation value (for linear analysis).

The following computer and software was used to search for characteristics: AMD Ryzen 5 2600, 32GB RAM, OS Ubuntu 22.04, Python version 3.10, CASCADA version - snapshot from `github.com/ranea/CASCADA` downloaded 24 febrary 2023.

## 3.2 Results of the search for optimal characteristics

The results obtained are shown in Tables 3 - 5. The "Correlation" column shows the weights for the linear characteristics; the XorDiff column shows the

weights for the XOR differences, the RXDiff column shows the rotational-XOR differences.

By default, the search for characteristics is performed by increasing the weight: the search starts with the minimum weight (for example, $w = 0$) and, in the absence of a solution from the SAT-solver, the weight value is increased by one and the search for a solution is retried.

For pCollapserARX64 and pCollapserARX128, the search for characteristics takes a significant amount of time (in excess of 5 days). With an increase in the value of the tested weight, the analysis time increases significantly, so tables 4 and 5 show only those lower values for which the SAT solver could not find a solution and this was displayed explicitly.

It should be noted that the search for a solution by the SAT solver for obviously large values of $w$ (relative to the real weight) is much faster. Thus, setting deliberately large values of $w$, we were able to explicitly obtain the upper values of the weights in a reasonable time.

For cases when the search for characteristics was not performed, the symbol "–" is put in the table. To search for Related-Key XorDiff characteristics, two rounds were used to form the "expanded key" (instead of 4). The last line of each table shows the applicability limit for the corresponding cryptanalysis method, taking into account the dimensions of the inputs/outputs.

Table 3: Weights of found characteristics for pCollapserARX32

| $w = -log_2 P(.)$ | | | | |
|---|---|---|---|---|
| Nrounds | Correlation | XorDiff | Related-Key XorDiff | RXDiff |
| 1 | 8 | 16 | 16 | 1276 |
| 2 | 26 | 32 | 64 | $2510 < w$ |
| 3 | 37 | 32 | 64 | $3060 < w$ |
| 4 | 50 | 32 | 64 | – |
| bounds | 16 | 32 | 32 | 32 |

Table 4: Weights of found characteristics for pCollapserARX64

| $w = -log_2 P(.)$ | | | |
|---|---|---|---|
| Nrounds | Correlation | XorDiff | RXDiff |
| 1 | 11 | 19 | $1508 < w < 1603$ |
| 2 | 42 | 88 | – |
| 3 | $55 < w < 88$ | 88 | – |
| 4 | $61 < w$ | – | – |
| bounds | 32 | 64 | 64 |

Table 5: Weights of found characteristics for pCollapserARX128

| $w = -log_2 P(.)$ | | | |
|---|---|---|---|
| Nrounds | Correlation | XorDiff | RXDiff |
| 1 | 11 | 19 | $1136 < w$ |
| 2 | $49 < w < 63$ | $110 < w < 180$ | – |
| 3 | – | – | – |
| bounds | 64 | 128 | 128 |

According to table 3, you can see that after round 2, the weight of the (Related-Key) XorDiff characteristic does not increase. An analysis of the found characteristics shows the presence of "collisions" at the outputs of the rounds, which manifests itself in the collide of the output differences in subsequent rounds.

Found differential characteristics (for pCollapserARX32):

1 : Ch($w$=16, id=fa 00 00 00, od=8c 00 00 00)
2 : Ch($w$=32, id=00 00 00 fa, od=00 00 00 48)
3 : Ch($w$=32, id=00 00 00 af, od=00 00 00 00)
4 : Ch($w$=32, id=00 00 00 af, od=00 00 00 00)

However, an experimental study of pCollapserARX miniversions showed that in reality the probability of a collision at the output of the second and subsequent rounds corresponds to the probability of a collision at the output of a random function (of the same dimension). Due to the presence of an actively updated internal state, which significantly exceeds the size of the input / output, the presence of a collision at the output of the second round will not lead to collide output values in subsequent rounds.

The discrepancy between the found characteristics and the real ones is explained by the fact that the differential model for the SAT solver is built taking into account the hypothesis of stochastic equivalence (also known as the Markov-cipher assumption [13]). This allows to split and analyze iterative functions in parts, thereby dramatically reducing the complexity of finding characteristics.

Research results showed, that the hypothesis of stochastic equivalence does not hold for the pCollapserARX (the intermediate propagations of properties within the characteristic are not independent – due to the nature of the dynamic operation of *PD-sboxes*) and the absence of round keys (updatable internal control state is used). This manifests itself in the form of a large number of found invalid (incompatible) characteristics.

In tables 3 - 5, we left the weights for the first found characteristics without checking their correctness in order to show the lower bound of the

weight values for pCollapserARX. Real values will be higher, but additional resource-intensive experiments are needed to determine the real optimal characteristics.

The problem also affects the search for characteristics for related-key, impossible-differential, related-key impossible-differential and zero-correlation cryptanalysis.

For this case we can use experimental script, created by Ranea and based on ideas from [10, 11]. As noted in the script description: "In this script we construct a model that describe difference transitions and value transitions simultaneously. ... Thus, the characteristics found for the constructed model are guaranteed to be valid, and we can recover from the SAT solution all the characteristics involved." Note that this script is experimental, and it has not been fully tested. Script available at (Github: `github.com/ranea/CASCADA/blob/master/cascada/experimental/diffvalchsearch.py`)

The results obtained using this script are shown in 6:

Table 6: Weights of found characteristics by *diffvalchsearch.py* script

| $w = -log_2 P(.)$ | | | | |
|---|---|---|---|---|
| Nrounds | pCollapserARX16 | pCollapserARX32 | pCollapserARX64 | pCollapserARX128 |
| 1 | 13 | 18 | 21 | 19 |
| 2 | 48 | $73 < w$ | $100 < w$ | $100 < w$ |
| 3 | 75 | – | – | – |
| bounds | 16 | 32 | 64 | 128 |

Found differential characteristics with conventional approach (for mini-version pCollapserARX16):

$$1 : \mathrm{Ch}(w{=}11,\ \mathrm{id}{=}0\ 0\ \mathrm{f}\ 0,\ \mathrm{od}{=}0\ 0\ 6\ 0)$$
$$2 : \mathrm{Ch}(w{=}23,\ \mathrm{id}{=}\mathrm{f}\ 0\ 0\ 0,\ \mathrm{od}{=}7\ 0\ 0\ 0)$$
$$3 : \mathrm{Ch}(w{=}24,\ \mathrm{id}{=}0\ 0\ 0\ \mathrm{f},\ \mathrm{od}{=}0\ 0\ 0\ 0)$$
$$4 : \mathrm{Ch}(w{=}24,\ \mathrm{id}{=}0\ 0\ 0\ \mathrm{f},\ \mathrm{od}{=}0\ 0\ 0\ 0)$$

Found valid differential characteristics (for pCollapserARX16):

$$1 : \mathrm{Ch}(w{=}13,\ \mathrm{id}{=}0\ \mathrm{d}\ 0\ 0,\ \mathrm{od}{=}0\ 8\ 0\ 0)$$
$$2 : \mathrm{Ch}(w{=}48,\ \mathrm{id}{=}0\ 0\ 8\ 0,\ \mathrm{od}{=}3\ 8\ \mathrm{a}\ \mathrm{f})$$
$$3 : \mathrm{Ch}(w{=}75,\ \mathrm{id}{=}0\ 0\ 8\ 0,\ \mathrm{od}{=}\mathrm{f}\ 0\ \mathrm{b}\ 0)$$

As you can see by the example of the mini version of pCollapserARX16, the obtained valid characteristics that do not collide (the output difference is non zero) and the weight value increases with each round. Last results

showed, that the hypothesis of stochastic equivalence does not hold for the pCollapserARX.

We tested the search for optimal characteristics using CASCADA for known algorithms (for example, in AES, SPECK, SKINNY, NOEKEON, RECTANGLE) using the usual Markov-cipher assumption approach and using the diffvalchsearch.py script. The weights of the obtained characteristics coincide, including with the known published results.

## 4 Conclusions

For the first time, a publicly available description of the PRF pCollapserARX for the CASCADA framework has been proposed, which allows to generate SAT-models and search for optimal cryptographic characteristics.

The results of the search for optimal cryptographic characteristics using CASCADA showed the resistance of the first 2 rounds (out of 4) of the PRF pCollapserARX to building distinguishers for attacks such as linear, differential, Related-Key differential, RX differential and Related-Key differential cryptanalysis.

Considering that the 1st round does not work in dynamic mode, we believe that the main contribution to resistance comes from the 2nd round, which works in dynamic mode. This confirms the correctness of both the use of pseudo-dynamic substitutions as a nonlinear element and the use of weak ARX-functions in their composition.

The presence of additional 3 and 4 rounds gives a significant security margin.

It is shown that, unlike most known symmetric cryptoalgorithms, the usual approach based on the Markov-cipher assumption is not suitable for searching for valid differential characteristics of PRF pCollapserARX, and a special model is required to search them, while the real weights of optimal characteristics are much higher.

Separately, it is worth noting the extremely large weight values for RX-Differences, perhaps there is an error in determining such characteristics.

## References

[1] Polikarpov S.V., Rumyantsev K.E., Kozhevnikov A.A., "On differential properties of a symmetric cryptoalgorithm based on pseudo-dynamic substitutions", *Mat. Vopr. Kriptogr.*, **7**, 2016 https://doi.org/10.4213/mvk186, 91–102.

[2] Polikarpov S.V., Kozhevnikov A.A., Rumyantsev K.E., "Pseudo-random function PCOLLAPSER providing extreme parallelism of information processing", *Izvestiya SFedU. Engi-*

*neering Sciences*, **5**, 2019 http://izv-tn.tti.sfedu.ru/index.php/izv_tn/article/view/388, 88–100.

[3] Darius Mercadier, Pierre-Évariste Dagand, "Usuba: high-throughput and constant-time ciphers, by construction.", *In Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 2019). Association for Computing Machinery, New York, NY, USA.*, 2019 https://doi.org/10.1145/3314221.3314636, 157–173.

[4] Polikarpov S.V., Rumyantsev K.E., Kozhevnikov A.A., "Investigation of linear characteristics of pseudo-dynamic substitutions", *Izvestiya SFedU. Engineering Sciences*, **5**, 2015 http://old.izv-tn.tti.sfedu.ru/wp-content/uploads/2015/5/11.pdf, 111–123.

[5] Polikarpov S.V., Kozhevnikov A.A., "Pseudo-dynamic substitutions: investigation of linear properties", *Izvestiya SFedU. Engineering Sciences*, **8**, 2015 http://old.izv-tn.tti.sfedu.ru/wp-content/uploads/2015/8/2.pdf, 19–31.

[6] Polikarpov S.V., Rumyantsev K.E., Kozhevnikov A.A., "Pseudo-dynamic substitutions: research of differential characteristics", Physical and mathematical methods and information technologies in natural science, engineering and humanities: collection of materials of the international scientific e-symposium, 2015, 77–89.

[7] Polikarpov S., Petrov D., Kozhevnikov A., "On A Class Pseudo-Dynamic Substitutions PD-Sbox, With A Perfect Averaged Distribution of Differentials in Static Mode of Work", Proceedings of the 2017 International Conference on Cryptography, Security and Privacy (ICCSP 17), 2017, 17–21.

[8] Polikarpov S., Rumyantsev K., Petrov D., "Computationally efficient method for determining averaged distribution of differentials for pseudo-dynamic substitutions", AIP Conference Proceedings 1952, 2018 https://doi.org/10.1063/1.5032053.

[9] Polikarpov S.V., Prudnikov V.A., Rumyantsev K.E., "Computationally efficient method for determining averaged linear properties for pseudo-dynamic substitutions", *Izvestiya SFedU. Engineering Sciences*, **5**, 2020 http://izv-tn.tti.sfedu.ru/index.php/izv_tn/article/view/388, 221–229.

[10] Sadegh Sadeghi, Vincent Rijmen, Nasour Bagheri. Proposing an MILP-based Method for the Experimental Verification of Difference Trails. (Cryptology ePrint Archive, Paper 2020/632),2020 https://eprint.iacr.org/2020/632.

[11] Jinyu Lu, Yunwen Liu, Tomer Ashur, Bing Sun, Chao Li. Improved Rotational-XOR Cryptanalysis of Simon-like Block Ciphers. (Cryptology ePrint Archive, Paper 2022/402),2022 https://eprint.iacr.org/2022/402.

[12] Ranea, A. Rijmen, V. Characteristic Automated Search of Cryptographic Algorithms for Distinguishing Attacks (CASCADA). (Cryptology ePrint Archive, Paper 2022/513),2022 https://eprint.iacr.org/2022/513.

[13] Zheng, X. Yongqiang, L. Lin, J. Mingsheng, W. Willi, M. Do NOT Misuse the Markov Cipher Assumption Automatic Search for Differential and Impossible Differential Characteristics in ARX Ciphers,2022 https://eprint.iacr.org/2022/135.pdf.

# About "$k$-bit security" of MACs based on hash function Streebog

Vitaly Kiryukhin

LLC «SFB Lab», JSC «InfoTeCS», Moscow, Russia
vitaly.kiryukhin@sfblaboratory.ru

**Abstract**

Various message authentication codes (MACs), including HMAC-Streebog and Streebog-K, are based on the keyless hash function Streebog. Under the assumption that the compression function of Streebog is resistant to the related key attacks, the security proofs of these algorithms were recently presented at CTCrypt 2022.

We carefully detail the resources of the adversary in the related key settings, revisit the proof, and obtain tight security bounds. Let $n$ be the bit length of the hash function state. If the amount of processed data is less than about $2^{n-k}$ blocks, then for HMAC-Streebog-512 and Streebog-K, the only effective method of forgery (or distinguishing) is guessing the $k$-bit secret key or the tag if it is shorter than the key. So, we can speak about "$k$-bit security" without specifying the amount of material, if the key length is no longer than half of a state. The bound for HMAC-Streebog-256 is worse and equal to $2^{\frac{n}{2}-k}$ blocks.

**Keywords:** Streebog, PRF, HMAC, provable security

## 1 Introduction

Russian hash function Streebog [1] is based on a modified Merkle-Damgård (MD) approach [6, 7]. The latter, as is well known, includes: padding the message $M$ and splitting it into $b$-bit blocks; iteratively applying the compression function $\mathsf{g}$ to the message block and the $n$-bit previous state; the initial state is the predefined constant; the last state is the result of hashing. Streebog uses $n = b = 512$, and its compression function is 12-rounds AES-like block cipher in Miyaguchi-Preneel mode. The output length can be either $\tau = 512$ or 256 bits.

Streebog has two features that differentiate it from the "plain" MD cascade:

– before processing the $i$-th block, the state is summed modulo 2 with the number of already hashed bits;

– the last call of the compression function is used to "mix" the checksum (modulo $2^n$) of all message blocks.

These features play an important role, especially when Streebog is used as the core for a keyed cryptoalgorithm, for example, a message authentication code (MAC) or a pseudorandom function (PRF).

Perhaps the most widespread and well-known way to construct a keyed transformation from a keyless hash function H is HMAC [8]

$$\mathsf{HMAC}(K, M) = \mathsf{H}\left((\overline{K} \oplus opad)||\mathsf{H}(\overline{K} \oplus ipad||M)\right),$$

where $\overline{K}$ is obtained by padding the secret key $K$ with zero bits, *opad* and *ipad* are different nonzero constants. Streebog can also be used in HMAC [2, 3], but security proofs [8, 12, 11, 14, 17] were proposed for HMAC with the "plain" MD hash function and therefore cannot be directly applied for HMAC-Streebog. The proof of the latter's security was initially given in [16] and later detailed in [32] by the reduction to the properties of g and not to the hash function itself.

On the other hand, the features of Streebog give a rise to a more efficient keyed mode, namely Streebog-K ("Keyed Streebog") [32]

$$\mathsf{Streebog\text{-}K}(K, M) = \mathsf{H}(\overline{K}||M),$$

where H is Streebog itself. Due to one hashing instead of two, the computation speed increases up to two times compared to HMAC.

The checksum used in Streebog leads to many so-called related keys inside both HMAC-Streebog and Streebog-K even when these cryptoalgorithms themselves are used in the *single-key* setting. The input of the last call of g is the secret key $\overline{K}$ summed with the message's blocks that are directly controlled by the adversary. As far as we know, by now there are no attacks for the compression function in the related-key setting that would be better than generic ones. Non-trivial results [23] were proposed only for the round-reduced version of g.

Despite this, generic related-key attacks have the great impact on the security bounds. Guessing any one of the $q$ related keys (and therefore all of them due to known relations) can be $q$ times faster than guessing a single secret key. However, if different related keys are used to process different inputs, then the adversary should choose a specific key when guessing, not any one. This simple observation fortunately holds (sometimes partially) for MACs based on Streebog.

We start by introducing the notation (section 2), and then provide high-level description of Streebog and the analyzed MACs (section 3).

Next, in section 4 we develop the $PRF\text{-}RKA$ threat model, which includes the above-mentioned observation by detailing the adversary's re-

sources. We discuss the properties of $\mathsf{g}$ in the introduced model and give the corresponding heuristic estimates.

Only one observation is not enough to obtain a result, in section 5 we present a new security proof using the example of Streebog-K. Assuming "good" properties of $\mathsf{g}$ the obtained security bounds can be described quite simply: up to about $2^{n-k}$ processed blocks, the only effective method of forgery (or distinguishing) is guessing the $k$-bit secret key or the tag if it is shorter than the key. For a $k$-bit key, in any case, it should be assumed that the amount of data does not exceed $2^k$. Hence, if $k \leq \frac{n}{2}$, then Streebog-K can be considered as "$k$-bit secure" without specifying the amount of material.

The attacks described in the sixth section demonstrate that the obtained estimates cannot be further improved (with the possible exception of a small multiplicative constant). In other words, each term in the upper bounds corresponds to a certain attack that has almost the same probability (the lower bound).

In the seventh section, similar results are given for HMAC-Streebog. Note that if the 256-bit version of Streebog is used in HMAC, then the bound is significantly worse ($2^{\frac{n}{2}-k}$).

## 2 Notations

We use the following notations throughout the paper:

$n = 512$ – block size in bits; $k \leq 512$ – key size in bits; $\oplus$ – bitwise XOR operation; $\boxplus$, $\boxminus$ – addition and subtraction modulo $2^n = 2^{512}$; $||$ – concatenation of binary strings;

$V^n$ – the set of all $n$-bit strings with naturally defined operations "$\oplus$" and "$\boxplus$";

$\mathsf{sum}_{\boxplus}(M) = m_1 \boxplus m_2 \boxplus \ldots \boxplus m_l$ – the checksum (modulo $2^n$) of $l$ blocks from the padded message $M||\mathtt{10\ldots0} = m_1||m_2||...||m_l$;

$\mathrm{Func}(\mathbf{X}, \mathbf{Y})$ – the set of all mappings from the set $\mathbf{X}$ to the set $\mathbf{Y}$;

$X \xleftarrow{\mathrm{R}} \mathbf{X}$ – uniform and random selection of element $X$ from the set $\mathbf{X}$.

The adversary is modeled by an interactive probabilistic algorithm that has access to other algorithms (oracles). We denote by $\mathrm{Adv}^{TM}_{\mathsf{Alg}}(\mathcal{A})$ a quantitative characterization (advantage) of the capabilities of the adversary $\mathcal{A}$ in realizing a certain threat, defined by the model $TM$, for the cryptographic scheme Alg. The resources of $\mathcal{A}$ are measured in terms of time and query complexities. The time complexity $t$ includes the description size of $\mathcal{A}$ in some computation model. The query complexity $q$ is measured in the number of adaptively chosen input/output pairs. Without loss of generality, we assume

that $\mathcal{A}$ always uses exactly $q$ unique queries (with no redundant or repeating queries). The result of computations of $\mathcal{A}$ after interacting with oracle $\mathcal{O}$ is some binary value $x$, which is denoted as $\mathcal{A}^{\mathcal{O}} \Rightarrow x$.

The maximum of the advantage among all resource constrained adversaries is denoted by

$$\mathrm{Adv}^{TM}_{\mathsf{Alg}}(t,q) = \max_{\mathcal{A}(t',q'):t'\le t,\ q'\le q,} \mathrm{Adv}^{TM}_{\mathsf{Alg}}(\mathcal{A}).$$

Some threat models, which would be addressed later, imply different types of resources, like the number of queries to different oracles, the length of these queries, etc. The advantage for such models is defined in similar way.

The cryptoalgorithm $\mathsf{Alg}$ is informally called secure in the threat model $TM$ ($TM$-secure) if $\mathrm{Adv}^{TM}_{\mathsf{Alg}}(t,q) < \varepsilon$, where $\varepsilon$ is some small value determined by the requirements for the strength of the cryptosystem and the resources $t$ and $q$ are comparable to those available to the adversary in practice.

To demonstrate the practical sense of the obtained results, we substitute heuristic estimates based on assumptions into derived security bounds. The resulting estimates are denoted by symbol "$\lessapprox$" meaning "less or equal if the assumptions are true".

Next, we prove anew (see also [32]) that $\mathsf{HMAC\text{-}Streebog}$ and $\mathsf{Streebog\text{-}K}$ are secure presudorandom functions (PRF).

**Definition.** *The advantage of $\mathcal{A}$ in the model $PRF$ ($PRF$-$CMA$ – indistinguishability from a random function under chosen message attack) for the keyed cryptoalgorithm $\mathsf{F} : \mathbf{K} \times \mathbf{X} \to \mathbf{Y}$ is*

$$\mathrm{Adv}^{PRF}_{\mathsf{F}}(\mathcal{A}) = \Pr\left(K \xleftarrow{\mathrm{R}} \mathbf{K}; \mathcal{A}^{\mathsf{F}_K(\cdot)} \Rightarrow 1\right) - \Pr\left(\mathsf{R} \xleftarrow{\mathrm{R}} \mathrm{Func}(\mathbf{X},\mathbf{Y}); \mathcal{A}^{\mathsf{R}(\cdot)} \Rightarrow 1\right),$$

*where $\mathbf{K}$, $\mathbf{X}$, $\mathbf{Y}$ are spaces of the keys, messages, and outputs respectively. The resources of $\mathcal{A}$ are $t$ computations and $q$ queries to the oracle, $l$ the maximum length of the queries (in $n$-bit blocks) if elements from $\mathbf{X}$ have variable length.*

By "$k$-bit security" we informally mean that the probability of realizing a certain threat (or the distinguishing advantage) with the time complexity $t$ is about $t/2^k$. All our statements about "$k$-bit security" are, first of all, true for the distinguishers in the $PRF$ model, and, therefore, the same statements is true for more dangerous threats, including forgeries [9] and key recovery attacks.

## 3 Streebog and MACs

Streebog hashes the message $M$ as follows. The text is padded with bit string $\mathtt{10}\ldots\mathtt{0}$. At least one bit is always added, even if the message bit length $L < 2^n$ is already divisible by $n$. The string $M' = M||\mathtt{10}\ldots\mathtt{0}$ is divided into $(l+1)$ blocks of $n = 512$ bits $M' = m_0||m_1||\ldots||m_l$. The compression function is sequentially applied to the previous state, the block and the counter

$$h_{i+1} = \mathsf{g}(h_i, m_i, \mathbf{i}),\ i = 0, ..., l,\ h_0 = IV_\tau \in V^n,$$

where $IV_\tau$ is a predefined constant which is different in both versions of the hash function, $\tau \in \{256, 512\}$, the $n$-bit counter $\mathbf{i} = (i \cdot n)$ represents the number of already hashed bits.

Two more transformations are performed at the finalizing stage: the bit length $L$ and the checksum $\Sigma = \mathsf{sum}_{\boxplus}(M) = m_0 \boxplus ... \boxplus m_l$ are "mixed" with the state

$$h_{l+2} = \mathsf{g}(h_{l+1}, L, \mathbf{0}),\ H = \mathsf{g}(h_{l+2}, \Sigma, \mathbf{0}).$$

If 256-bit hash function is used, the output $H$ is truncated to 256 bit. Here and further, the integers at the input of $\mathsf{g}$ are implicitly converted into $n$-bit vectors.

The compression function is based on a 12-rounds AES-like block cipher $\mathsf{E}$ in Miyaguchi-Preneel mode

$$\mathsf{g}(h_i, m_i, \mathbf{i}) = \mathsf{E}(h_i \oplus \mathbf{i}, m_i) \oplus h_i \oplus m_i = h_{i+1}.$$

In [19], the equivalent representation was proposed (see also details in [32]). The counter $\mathbf{i}$ ceases to be a parameter of the compression function. The latter is simplified to $\mathsf{g}(h, m) = \mathsf{E}(h, m) \oplus h \oplus m$ (and further in the text, this is what is meant by $\mathsf{g}$). After processing the $i$-th block, the state is summed modulo 2 with the constant $\Delta_i = \mathbf{i} \oplus (\mathbf{i} \boxplus \mathbf{1})$, $i = 0, \ldots, l-1$, but after the $l$-th block, another constant is used, namely $\widetilde{\Delta}_l = \mathbf{l}$, and $\Delta_i \neq \widetilde{\Delta}_i$, $\forall i = 0, ..., 2^n - 1$. Thus, both versions of Streebog can be expressed as

$$\mathsf{H}_\tau(M) = \mathsf{msb}_\tau\left(\mathsf{g}(\mathsf{g}(\ldots(\mathsf{g}(\mathsf{g}(IV_\tau, m_0) \oplus \Delta_0, m_1) \oplus \Delta_1)\ldots \oplus \widetilde{\Delta}_l, L), \Sigma)\right),$$

where $\mathsf{msb}_\tau : V^n \to V^\tau$ is the $\tau$ most significant bits. Next, we omit the index $\tau$ if any of its values are suitable.

Various keyed cryptoalgorithms use Streebog in a black-box way, without making any changes to the Streebog itself, but only preparing the input for it. These algorithms are usually used as message authentication codes (MAC) and key derivation functions (KDF).

Here we list the formulas of the analyzed algorithms based on Streebog [2, 32], and also mention their features,

$$\mathsf{HMAC\text{-}Streebog}(K, M) = \mathsf{H}\left((\overline{K} \oplus opad)||\mathsf{H}(\overline{K} \oplus ipad||M)\right),$$
$$\mathsf{Streebog\text{-}K}(K, M) = \mathsf{H}(\overline{K}||M).$$

The secret key $K \in V^k$ is padded with $(n-k)$ zero bits if necessary $\overline{K} = (K||0...0)$. Two different $n$-bit constants $ipad \neq opad$ are used in HMAC-Streebog.

The key length for HMAC-Streebog, according to [2], is $256 \leq k \leq 512$ bits, and the same restriction is proposed in [32] for Streebog-K. Further, for generality, we assume that $k \leq n$.

HMAC-Streebog and Streebog-K can use both versions of H (with 256-bit and 512-bit output). Due to the double hashing in the first one, this leads to a significant impact on the security bounds.

Next, we describe our results using the example of Streebog-K, the last section and Appendix B are devoted to HMAC-Streebog.

For the sake of consistency with the previously introduced notation, let $\overline{K} = m_0$ and $M = m_1||...||m_l$ (fig. 1). By the cascade transformation Csc we mean further

$$\mathsf{Csc}(K_{\mathsf{Csc}}, M) = \mathsf{g}(\ldots \mathsf{g}(\mathsf{g}(K_{\mathsf{Csc}} \oplus \Delta_0, m_1) \oplus \Delta_1, m_2) \ldots \oplus \widetilde{\Delta}_l, L),\ K_{\mathsf{Csc}} \in V^n,$$

assuming that the input $M$ of arbitrary bit length is padded by $\mathtt{10}\ldots\mathtt{0}$, and the length $L$ increases by $n$ because of the key.

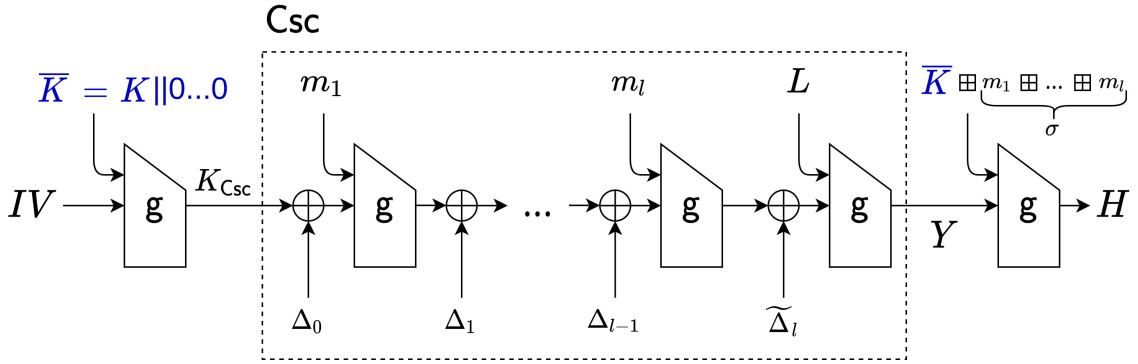



Figure 1: The equivalent representation of Streebog-K. The checksum is $\Sigma = \overline{K} \boxplus \sigma$, and $\sigma = \mathsf{sum}_{\boxplus}(M)$. The result of cascade is $Y = \mathsf{Csc}(K_{\mathsf{Csc}}, M)$.

# 4 Related key settings

For all the considered cryptoalgorithms, security is reduced to the properties of the compression function $\mathsf{g}$ under the various related key attacks (RKA). We capture all the required properties in the following definition.

**Definition**. *The advantage of $\mathcal{A}$ in the model $PRF\text{-}RKA_{\circledast}$ for the keyed cryptoalgorithm $\mathsf{F} : \mathbf{K} \times \mathbf{X} \to \mathbf{Y}$ is*

$$\mathrm{Adv}_{\mathsf{F}}^{PRF\text{-}RKA_{\circledast}}(\mathcal{A}) = \Pr\left(K \xleftarrow{\mathrm{R}} \mathbf{K}; \mathcal{A}^{\mathsf{F}_{K\circledast\cdot}(\cdot)} \Rightarrow 1\right) - \\ - \Pr\left(K \xleftarrow{\mathrm{R}} \mathbf{K}; \mathsf{R}_i \xleftarrow{\mathrm{R}} \mathrm{Func}(\mathbf{X}, \mathbf{Y}),\ \forall i \in \mathbf{K}; \mathcal{A}^{\mathsf{R}_{K\circledast\cdot}(\cdot)} \Rightarrow 1\right),$$

*where $\mathbf{K}$, $\mathbf{X}$, $\mathbf{Y}$ are spaces of the keys, messages, and outputs respectively. The $w$-ary operation "$\circledast$" over $\mathbf{K}$ is the parameter of the model. The query from $\mathcal{A}$ consists of the input $x \in \mathbf{X}$ and the relation $\kappa \in \mathbf{K}^{w-1}$. The response is the value $y = \mathsf{F}_{K\circledast\kappa}(x)$ (resp. $y = \mathsf{R}_{K\circledast\kappa}(x)$). The resources of $\mathcal{A}$ are $t$ computations and $q$ queries to the oracle. The content of queries is limited by the number of relations ($r$) and by the number different relations ($d$) queried with the same $x$ ($d \le r \le q$).*

We omit $d$ in the notations if $d \le r$, and also omit $r$ if $r \le q$.

Note, if "$\circledast$" is the unary identity operation, then $PRF\text{-}RKA_{\circledast}$ is essentially the same as the usual $PRF$ model.

Through the paper we instantiate the $PRF\text{-}RKA_{\circledast}$ model using binary operations "$\oplus$" and "$\boxplus$" over $V^n$. The $\mathsf{HMAC\text{-}Streebog}$ analysis also required the introduction of a ternary operation, denoted by "$\boxplus \circ \oplus$" (so, the key $K$ under the relation $\kappa = (\phi, \sigma)$ is $(K \oplus \phi) \boxplus \sigma$). Further in the text, "$\circledast$" denotes any of these three operations.

The main novelty introduced in the above definition is the parameter $d$. We show its importance for the generic attacks against arbitrary PRF $\mathsf{F} : \mathbf{K} \times \mathbf{X} \to \mathbf{Y}$. Let, for example, $\circledast = \boxplus$ and $\mathbf{K} = V^k$.

In the absence of restrictions ($d = r = q$), the adversary can query a single $x$ under the different relations $(x, \kappa_1)$,...,$(x, \kappa_q)$ and obtain $y_1$,...,$y_q$, $y_i = \mathsf{F}_{K\boxplus\kappa_i}(x)$. Next, the adversary repeats $t$ times: guess the key $\tilde{K}$; compute $\tilde{y} = \mathsf{F}_{\tilde{K}}(x)$; find $\tilde{y}$ among the stored $(y_1, ..., y_q)$. If $\tilde{K}$ is equal to ***any*** of the related keys used $(K \boxplus \kappa_1, ..., K \boxplus \kappa_q)$, let this be $K \boxplus \kappa_i$, then certainly $\tilde{y} = y_i$. Hence, the attacker obtains $\tilde{K} = K \boxplus \kappa_i$ and computes the key $K = \tilde{K} \boxminus \kappa_i$ using the known relation $\kappa_i$. The success probability is upper bounded by $t \cdot q \cdot 2^{-k}$. False positives $\tilde{y} \in (y_1, ..., y_q)$ do not affect the essence and are ignored here.

Now let $d = 1$ and $r \le q$. In this case, the queries are $(x_1, \kappa_1)$,...,$(x_q, \kappa_q)$. Regardless of the relations, all inputs are different, $x_i \ne x_j$, $1 \le i < j \le q$.

Therefore, before the guessing attempt, the adversary can choose only one key (i.e. under one relation) that he will try to find. In other words, he can compute $\tilde{y} = \mathsf{F}_{\tilde{K}}(x_i)$ and check $\tilde{y} = y_i$, hoping that $\tilde{K} = K \boxplus \kappa_i$. Matches with other keys $K \boxplus \kappa_j$, $j \neq i$, cannot be verified because of $x_i \neq x_j$. Therefore, in such conditions, the success probability is about $t \cdot 2^{-k}$ and does not depend on the total number of queries.

Thus, assuming the absence of specific vulnerabilities, the best distinguishing method is key guessing, and the advantage is bounded by

$$\mathrm{Adv}_{\mathsf{F}}^{PRF\text{-}RKA_{\boxplus}}(t,q,r,d) \lessapprox \frac{t \cdot d}{2^k} \leq \frac{t \cdot r}{2^k} \leq \frac{t \cdot q}{2^k}.$$

Note that the presented estimates are heuristic in nature. However, these inequalities are easy to prove if $\mathsf{F}$ is considered as a family of $2^k$ random functions (i.e. in the so-called random oracle model). The same considerations make it possible to ignore attacks based on "free" precomputations [15].

To the best of our knowledge, there are no attacks on the Streebog compression function that would be better than the generic ones. The round-reduced versions of $\mathsf{g}$ were considered in the secret-key [21, 22] and the related-key settings [23]. The situation is similar with the keyless settings (preimages and various collisions) [24, 25, 26, 27, 28, 29, 30, 31], that is also an indirect argument in favor of good cryptographic properties under the related key attacks.

Therefore, we use

$$\mathrm{Adv}_{\mathsf{g}^{\triangledown}}^{PRF\text{-}RKA_{\circledast}}(t,q,r,d) \lessapprox \frac{t \cdot d}{2^k}, \tag{1}$$

as an heuristic estimate for $\mathsf{g}^{\triangledown}_{\overline{K}}(\cdot) = \mathsf{g}(\cdot, \overline{K})$, $k \leq n$, $\mathbf{K} = \{\overline{K} : \ K \in V^k\}$, instead of $t \cdot q \cdot 2^{-k}$ used in [32].

If the secret key of $\mathsf{g}$ is the $n$-bit state $h$ of the hash function, $\mathsf{g}^{\triangleright}_h(\cdot) = \mathsf{g}(h, \cdot)$, then the bound [32] remains the same

$$\mathrm{Adv}_{\mathsf{g}^{\triangleright}}^{PRF\text{-}RKA_{\oplus}}(t,q,r \leq 2) \lessapprox \frac{2 \cdot t}{2^n} + \frac{q(q-1)}{2^{n+1}}. \tag{2}$$

Recall that, for all the cryptoalgorithms under consideration, $\mathsf{g}^{\triangleright}$ is used with no more than two related keys ($d \leq r \leq 2$). The relation is defined by $\phi_i = \widetilde{\Delta}_i \oplus \Delta_i$, $i = 1, ..., l$. The second term in (2) arises due to the birthday-paradox distinguisher (see also [22]).

We emphasize that some new effective attacks on full-round $\mathsf{g}$ can potentially affect the heuristic estimates and, consequently, the claims about "$k$-bit security". We are convinced that the construction of such methods is

extremely difficult, but anyway the theorems presented below hold. Their statements are essentially about the high-level design of the Streebog-based cryptoalgorithms, and not the hash function itself.

## 5 Revision of PRF-security for Streebog-K

The PRF-security bound of Streebog-K (denoted here for compactness as KH) presented in [32] as

$$\begin{aligned}\mathrm{Adv}^{PRF}_{\mathsf{KH}}(t,q,l) \leq & \mathrm{Adv}^{PRF\text{-}RKA_{\boxplus}}_{\mathsf{g}^{\triangledown}}(t',q',r=q',d=q')+ \\ & +q\cdot l'\cdot \mathrm{Adv}^{PRF\text{-}RKA_{\oplus}}_{\mathsf{g}^{\triangleright}}(t',q,r=2)+\frac{q^2+q}{2^{n+1}}, \end{aligned} \tag{3}$$

where $t'=t+O(q\cdot l)$, $q'=q+1$, $l'=l+1$.

Recall that the $i$-th message $M_i$ is transformed as follows

$$\mathsf{KH}_K(M_i)=\mathsf{g}^{\triangledown}_{\overline{K}\boxplus\sigma_i}(\mathsf{Csc}(\mathsf{g}^{\triangledown}_{\overline{K}}(IV),M_i)),$$

and $K_{\mathsf{Csc}}=\mathsf{g}^{\triangledown}_{\overline{K}}(IV)$, the result of the cascade $Y_i=\mathsf{Csc}(K_{\mathsf{Csc}},M_i)$, the relation is determined by $\sigma_i=\mathsf{sum}_{\boxplus}(M_i)$, $1\leq i\leq q$, (see fig. 1).

By "collision" in this section we mean the coincidence of any pair of elements in the sequence $IV,Y_1,Y_2,...,Y_q$, and denote it as C, the opposite event is denoted by $\overline{\mathrm{C}}$.

The term $\mathrm{Adv}^{PRF\text{-}RKA_{\boxplus}}_{\mathsf{g}^{\triangledown}}$ is the most significant when $k<n$, and is the only term that depends on the key length $k$. With $d=q'$, the $PRF\text{-}RKA_{\boxplus}$ model allows the inputs $(x,\sigma)$ for $\mathsf{g}^{\triangledown}$ to have *any* form, because of this, the estimate increases by $t\cdot q'\cdot 2^{-k}$. Whereas in fact, until there has been a "collision", the values of $x$ in all queries to $\mathsf{g}^{\triangledown}$ are *different* ($d=1$). Otherwise ("collision" has occurred), the attacker has already achieved his goal, the probability of this is taken into account in the third term of the bound.

The formalization of the above considerations is expressed by the following theorem.

**Theorem (PRF-security of Streebog-K).** *The advantage of the adversary in the PRF model attacking* Streebog-K *is bounded by* $\mathrm{Adv}^{PRF}_{\mathsf{KH}}(t,q,l)\leq$

$$\leq \mathrm{Adv}^{PRF\text{-}RKA_{\boxplus}}_{\mathsf{g}^{\triangledown}}(t',q',q',d=1)+\mathrm{Adv}^{PRF}_{\mathsf{Csc}}(t',q,l')+\frac{q^2+q}{2^{n+1}}, \tag{4}$$

*where* $t'=t+O(q\cdot l)$, $q'=q+1$, $l'=l+1$.

Proof.

Let's consider $\widetilde{\mathsf{KH}}(M_i) = \mathsf{f}_{\overline{K}\boxplus\sigma_i}(\mathsf{Csc}(\mathsf{f}_{\overline{K}\boxplus 0}(IV), M_i))$, the first and the last calls of $\mathsf{g}^{\triangledown}$ are replaced in $\mathsf{KH}$ by a family of $2^n$ random functions $\mathsf{f}$ indexed by $\sigma_i$. If the "collision" does not occur, then the cascade key $K_{\mathsf{Csc}} = \mathsf{f}_{\overline{K}\boxplus 0}(IV)$ is not observed by the attacker and is truly random. Moreover, under the same conditions, $\widetilde{\mathsf{KH}}$ is indistinguishable from a random function $\mathsf{R}$, due to the fact that regardless of $\sigma_i$, all values $IV, Y_1, ..., Y_q$ requested from $\mathsf{f}$ are not repeated.

The "collision" that occurred as a result of the adversary's interaction with $\mathsf{KH}$ is denoted as $\mathcal{A}^{\mathsf{KH}(\cdot)} \Rightarrow (b, \mathrm{C})$. What we mean by this is that $b \in \{0, 1\}$ is the immediate result returned by the adversary, and the "collision" is an implicit side result. Note that anyone who knows $K_{\mathsf{Csc}}$ can easily determine whether there was the "collision" or not. We omit $b$ in the notation if its value can be any, $\Pr(\mathcal{A}^{\mathsf{KH}(\cdot)} \Rightarrow \mathrm{C}) = \Pr(\mathcal{A}^{\mathsf{KH}(\cdot)} \Rightarrow (1, \mathrm{C})) + \Pr(\mathcal{A}^{\mathsf{KH}(\cdot)} \Rightarrow (0, \mathrm{C}))$.

By the definition of the $PRF$ model,

$$\mathrm{Adv}^{PRF}_{\mathsf{KH}}(\mathcal{A}) = \Pr(\mathcal{A}^{\mathsf{KH}(\cdot)} \Rightarrow 1) - \Pr(\mathcal{A}^{\mathsf{R}(\cdot)} \Rightarrow 1).$$

Using the formula of total probability, we get

$$\Pr(\mathcal{A}^{\mathsf{KH}(\cdot)} \Rightarrow 1) = \Pr(\mathcal{A}^{\mathsf{KH}(\cdot)} \Rightarrow (1, \mathrm{C})) + \Pr(\mathcal{A}^{\mathsf{KH}(\cdot)} \Rightarrow (1, \overline{\mathrm{C}})).$$

As we explained above,

$$\Pr(\mathcal{A}^{\mathsf{R}(\cdot)} \Rightarrow 1) = \Pr(\mathcal{A}^{\widetilde{\mathsf{KH}}(\cdot)} \Rightarrow (1, \overline{\mathrm{C}})).$$

By grouping the terms and using the triangle inequality, we obtain $\mathrm{Adv}^{PRF}_{\mathsf{KH}}(\mathcal{A}) =$

$$\begin{aligned}
&= \Big(\Pr(\mathcal{A}^{\mathsf{KH}(\cdot)} \Rightarrow (1, \mathrm{C})) + \Pr(\mathcal{A}^{\mathsf{KH}(\cdot)} \Rightarrow (1, \overline{\mathrm{C}}))\Big) - \Pr(\mathcal{A}^{\widetilde{\mathsf{KH}}(\cdot)} \Rightarrow (1, \overline{\mathrm{C}})) \leq \\
&\leq \Big(\Pr(\mathcal{A}^{\mathsf{KH}(\cdot)} \Rightarrow (1, \overline{\mathrm{C}})) - \Pr(\mathcal{A}^{\widetilde{\mathsf{KH}}(\cdot)} \Rightarrow (1, \overline{\mathrm{C}}))\Big) + \Pr(\mathcal{A}^{\mathsf{KH}(\cdot)} \Rightarrow (1, \mathrm{C})) \leq \\
&\leq \Big(\Pr(\mathcal{A}^{\mathsf{KH}(\cdot)} \Rightarrow (1, \overline{\mathrm{C}})) - \Pr(\mathcal{A}^{\widetilde{\mathsf{KH}}(\cdot)} \Rightarrow (1, \overline{\mathrm{C}}))\Big) + \Pr(\mathcal{A}^{\mathsf{KH}(\cdot)} \Rightarrow \mathrm{C}) \leq \\
&\leq \Big(\Pr(\mathcal{A}^{\mathsf{KH}(\cdot)} \Rightarrow (1, \overline{\mathrm{C}})) - \Pr(\mathcal{A}^{\widetilde{\mathsf{KH}}(\cdot)} \Rightarrow (1, \overline{\mathrm{C}}))\Big) + \\
&\quad + \Big(\Pr(\mathcal{A}^{\mathsf{KH}(\cdot)} \Rightarrow \mathrm{C}) - \Pr(\mathcal{A}^{\widetilde{\mathsf{KH}}(\cdot)} \Rightarrow \mathrm{C})\Big) + \Pr(\mathcal{A}^{\widetilde{\mathsf{KH}}(\cdot)} \Rightarrow \mathrm{C}) = \\
&= \epsilon + \epsilon_{coll} + p_{coll}.
\end{aligned}$$

Let's use both $\epsilon$ and $\epsilon_{coll}$ at the same time. In other words, we utilize in the single algorithm $\mathcal{B}_1$: the ability of $\mathcal{A}$ to distinguish between $\mathsf{KH}$ and $\widetilde{\mathsf{KH}}$ when there are no collisions (term $\epsilon$); the advantage $\epsilon_{coll}$ arising from the

difference in the probability of collisions. We assume that $\epsilon \geq 0$ and $\epsilon_{coll} \geq 0$, otherwise, we can invert the corresponding result.

$\mathcal{B}_1$ attacks $\mathsf{g}^{\triangledown}$ in the $PRF\text{-}RKA_{\boxplus}$ model. Initially, $\mathcal{B}_1$ queries the cascade key $K_{\mathsf{Csc}} = \mathcal{O}(IV, 0)$ from the oracle $\mathcal{O} \in \{\mathsf{g}^{\triangledown}, \mathsf{f}\}$. When processing each query $M_i$ from $\mathcal{A}$, the algorithm $\mathcal{B}_1$ computes $Y_i = \mathsf{Csc}(K_{\mathsf{Csc}}, M_i)$ and $\sigma_i = \mathsf{sum}_{\boxplus}(M_i)$. The value of $Y_i$ is written in memory. Next, $\mathcal{B}_1$ checks the "collision" condition. If $Y_i \in \{IV, Y_1, ..., Y_{i-1}\}$ then $\mathcal{B}_1$ returns 1 (due to $\epsilon_{coll} \geq 0$, the "collision" is interpreted as an interaction with $\mathsf{KH}$) and turns off $\mathcal{A}$ (further interaction does not make sense). Otherwise ($Y_i \notin \{IV, Y_1, ..., Y_{i-1}\}$), $\mathcal{B}_1$ makes query $(Y_i, \sigma_i)$ to the oracle and transmits the response to $\mathcal{A}$. If the "collision" conditions have never been met after $q$ queries, then the result of $\mathcal{B}_1$ is the result of $\mathcal{A}$.

The computation resources of $\mathcal{B}_1$ is $t' = t + O(q \cdot l)$, no more than $(q+1)$ queries are made to the oracle, the number of the related keys $r \leq q+1$, no value is requested from the oracle twice, $d = 1$.

Until the "collision" occurs, $\mathcal{B}_1$, interacting with $\mathsf{g}^{\triangledown}$ or $\mathsf{f}$, perfectly simulates for $\mathcal{A}$ oracles $\mathsf{KH}$ or $\widetilde{\mathsf{KH}}$ respectively. The distinguishing advantage of $\mathcal{B}_1$ is equal to

$$\begin{aligned}\mathrm{Adv}^{PRF\text{-}RKA_{\boxplus}}_{\mathsf{g}^{\triangledown}}(\mathcal{B}_1) &= \Pr(\mathcal{B}_1^{\mathsf{g}^{\triangledown}_{K\boxplus}.(\cdot)} \Rightarrow 1) - \Pr(\mathcal{B}_1^{\mathsf{f}_{K\boxplus}.(\cdot)} \Rightarrow 1) = \\ &= \Big(\Pr(\mathcal{A}^{\mathsf{KH}(\cdot)} \Rightarrow (1, \overline{\mathrm{C}})) + \Pr(\mathcal{A}^{\mathsf{KH}(\cdot)} \Rightarrow \mathrm{C})\Big) - \\ &- \Big(\Pr(\mathcal{A}^{\widetilde{\mathsf{KH}}(\cdot)} \Rightarrow (1, \overline{\mathrm{C}})) + \Pr(\mathcal{A}^{\widetilde{\mathsf{KH}}(\cdot)} \Rightarrow \mathrm{C})\Big) = \epsilon + \epsilon_{coll}.\end{aligned}$$

All that remains is to limit the value of $p_{coll} = \Pr(\mathcal{A}^{\widetilde{\mathsf{KH}}(\cdot)} \Rightarrow \mathrm{C})$. We construct the algorithm $\mathcal{B}_2$ that can effectively distinguish $\mathsf{Csc}$ from a random function $\mathsf{R}$. $\mathcal{B}_2$ passes the request $M_i$ from $\mathcal{A}$ to its own oracle $\mathcal{O} \in \{\mathsf{Csc}, \mathsf{R}\}$, receives the response $Y_i$ and stores this value in memory. When processing each query, $\mathcal{B}_2$ checks if the "collision" occurred. If it did, $\mathcal{B}_2$ returns 1 and turns off $\mathcal{A}$. Otherwise, $\mathcal{B}_2$ generates a random value (simulates $\mathsf{f}$) and returns it to $\mathcal{A}$. If there is no "collision" after $q$ queries, then the result of $\mathcal{B}_2$ is 0. The advantage of $\mathcal{B}_2$ is lower bounded by

$$\begin{aligned}\mathrm{Adv}^{PRF}_{\mathsf{Csc}}(\mathcal{B}_2) &= \Pr(\mathcal{B}_2^{\mathsf{Csc}(K_{\mathsf{Csc}},\cdot)} \Rightarrow 1) - \Pr(\mathcal{B}_2^{\mathsf{R}(\cdot)} \Rightarrow 1) \geq \\ &\geq \Pr(\mathcal{A}^{\widetilde{\mathsf{KH}}(\cdot)} \Rightarrow \mathrm{C}) - \left(\frac{q \cdot (q-1)}{2^{n+1}} + \frac{q}{2^n}\right),\end{aligned}$$

where $\dfrac{q}{2^n}$ is the probability of $IV \in \{Y_1, ..., Y_q\}$, and $\dfrac{q \cdot (q-1)}{2^{n+1}}$ corresponds to the collision among $q$ values $Y_1, ..., Y_q$ returned from the random oracle

$\mathsf{R}(\cdot)$. Therefore,

$$\Pr(\mathcal{A}^{\widetilde{\mathsf{KH}}(\cdot)} \Rightarrow \mathrm{C}) \leq \mathrm{Adv}^{PRF}_{\mathsf{Csc}}(\mathcal{B}_2) + \frac{q^2+q}{2^{n+1}}.$$

$\mathcal{B}_2$ appends a block containing $L$ to the messages from $\mathcal{A}$, so that because of the extra block we have $l' = l + 1$. $\square$

The PRF-security of the cascade $\mathsf{Csc}$ is proved in [32] as a separate lemma

$$\mathrm{Adv}^{PRF}_{\mathsf{Csc}}(t', q, l') \leq q \cdot l' \cdot \mathrm{Adv}^{PRF\text{-}RKA_{\oplus}}_{\mathsf{g}^{\triangleright}}(t', q, 2). \tag{5}$$

Direct substitution of the heuristic estimate (2) into (5) leads to an inaccurate result, due to the fact that (2) depends quadratically on $q$.

Hence, a more general bound, also stated in [32], is convenient for us here

$$\mathrm{Adv}^{PRF}_{\mathsf{Csc}}(\mathcal{A}) \leq \sum_{i=1}^{q}\sum_{j=1}^{l} \mathrm{Adv}^{PRF\text{-}RKA_{\oplus}}_{\mathsf{g}^{\triangleright}}(\mathcal{B}_{i,j}),$$

where $\mathcal{B}_{i,j}$ corresponds to some inner node of the special tree formed by queries. The root of the tree is $K_{\mathsf{Csc}}$, the nodes are the intermediate secret states, the results ($Y_1$,...,$Y_q$) are stored in leaves. Each edge of the tree is labeled with the the block from the messages.

So, this tree has at least $l$ and at most $(1 + q \cdot (l-1)) \leq ql$ nodes (the multiplier in (5)). The lower bound is exact when all messages differ only in the last $l$-th block. The opposite is achieved when all messages are different in the first block. The adversary $\mathcal{B}_{i,j}$ makes $1 \leq q_{i,j} \leq q$ queries to the oracle, and $q_{i,j}$ also corresponds to the number of the edges from the node. So if $q_{i,j}$ increases by one, then the number of leaves also becomes one more, but there are no more leaves in total than $q$. Thus, we have inequality

$$\sum_i \sum_j (q_{i,j} - 1) \leq q,$$

and the PRF-security of the cascade is estimated as $\mathrm{Adv}^{PRF}_{\mathsf{Csc}}(t', q, l') \lessapprox$

$$\lessapprox \sum_{i=1}^{q}\sum_{j=1}^{l'} \left( \frac{2 \cdot t'}{2^n} + \frac{q_{i,j} \cdot (q_{i,j}-1)}{2^{n+1}} \right) \leq \frac{2 \cdot t' \cdot q \cdot l'}{2^n} + \frac{q^2}{2^{n+1}}. \tag{6}$$

By using the result (4) of the theorem and the estimates (1), (6), we finally obtain

$$\mathrm{Adv}^{PRF}_{\mathsf{KH}}(t, q, l) \lessapprox \frac{t'}{2^k} + \frac{2 \cdot t' \cdot q \cdot l'}{2^n} + \frac{q^2+q}{2^n},\ t' \approx t,\ l' = l + 1, \tag{7}$$

and make a claim about "$k$-bit security". If the key length $\frac{n}{2} \leq k \leq n$ and the amount of the processed blocks $q \cdot l < 2^{n-k-1}$, then the most significant term is $\frac{t}{2^k}$ the probability of successfully guessing the key. Obviously, the data constraint $q \cdot l < 2^k$ is always assumed. Hence, for a key of shorter length $k \leq \frac{n}{2}$, again, the most significant term is the first one.

The statement above concerns distinguishing attacks, but the same holds if the adversary's goal is to forge. The *probability* of at least one successful forgery in $\nu$ attempts is bounded by [9, Proposition 7.3] (SUF – Strong UnForgeablility)

$$\mathrm{Adv}^{SUF}_{\mathsf{KH}}(t, q, l, \nu) \leq \mathrm{Adv}^{PRF}_{\mathsf{KH}}(t', q + \nu, l) + \frac{\nu}{2^\tau},\ t' \approx t.$$

So, if the output length is sufficiently large ($\tau \geq k$), then the statement about "$k$-bit security" is also true in this case. For small $\tau < k$, we make a reservation that another attack strategy is tag guessing.

We emphasize that exceeding the border of $2^{n-k}$ blocks may be quite *acceptable*. The probability of a forgery in one attempt is greater than "ideal" $2^{-\tau}$, but in most practical cases it is *negligible*, even if the number of the processed blocks far exceeds $2^{n-k}$.

Note for completeness that the bound similar to (7) can be obtained by using results of [14, 17] for the HMAC with the "plain" MD hash function, say HMAC-SHA-512 [4]. However, SHA-512$(K||\cdot)$, unlike Streebog, is completely insecure as PRF.

The "sponge"-based hash functions (for example, SHA-3 [5]) can be used with the key in the prefix as a secure PRF. The security bound for the keyed sponge [10, Theorem 1] is also close to (7) if we consider the sponge "capacity" $c$ as the state size $n$.

## 6 Attacks and tightness of the upper bounds

The attacks and the proofs are "the two sides of the same coin" [18]. The proofs give us the upper bounds of the insecurity (it can't be worse than that), the attacks provide the lower bounds (the attacker can definitely act with such an advantage or probability of success).

In this section we show that for Streebog-K "as a high-level design" both bounds are close. Therefore, (7) cannot be improved by more than a small multiplicative factor, so we can consider it tight enough.

On the contrary, we find it somewhat paradoxical that Streebog-K "as a real MAC with a real compression function g" may be even *more secure*.

Further, it can be refuted by some kind of sophisticated attack showing that (7) is tight for this case as well. Another way to refine the estimates seems to be random oracle model, which in simple words means an unconditional belief that $\mathsf{g}$ is a family of random functions.

The first term in (7) obviously corresponds to the straightforward key guessing. We should consequently also note that 4 computations of $\mathsf{g}$ are required to verify one key. Hence, the probability of success (correct guessing using $t$ computations) is 4 times less than the upper bound $t \cdot 2^{-k}$.

The third term in (7) is almost achievable with the birthday-paradox attack. The adversary: queries tags $H_i$ for the messages $M_i = m_i||m'_i$, $m_i \boxplus m'_i = const$, $1 \le i \le q$; looks for a collision ($H_i = H_j$); makes one additional query $M_i||P$, $P \in V^n$ and obtains the tag $H_{q+1}$; finally makes a forgery $M_j||P$ with tag $H_{q+1}$. The probability of a collision is about $q^2 \cdot 2^{-(n+1)}$, which is approximately half the upper estimate.

The reason for the above-mentioned "paradox" is the second term $t \cdot q \cdot l \cdot 2^{-n}$ arising from the imperfection of the cascade. If we consider time- and data-balanced attacks ($t \approx q \cdot l$), then the bound depends quadratically on the amount of data $q^2 \cdot l^2 \cdot 2^{-n}$. From this point of view, the best known attack is $l$ times worse. The probability of the birthday-paradox attack equal to $\approx q^2 \cdot l \cdot 2^{-n}$ if long $l$-block messages are used [20, 32]. We also don't know the matching attack for "real" Streebog-K if the computational power of the adversary is greater ($t \gg q \cdot l$).

The attack for the "plain" MD cascade (i.e. without counters and $\Delta_i$) with tight distinguishing advantage $t \cdot q \cdot l \cdot 2^{-n}$ can be easily constructed by using the properties of the random mapping graph, but in HMAC and Streebog-K the output of the cascade is not directly observed due to the key-dependent finalization. Counters also make attacks more difficult.

To demonstrate the accuracy of the upper estimates, we use a rather artificial trick proposed in [13, 14] for HMAC with the "plain" cascade. We construct a weak compression function $\mathsf{w}$ so that the cascade degrades as well as with $\mathsf{g}$, but with $\mathsf{w}$ it would be easily exploited in an attack (see details in Appendix A).

**Proposition**. *For any arbitrary compression function* $\mathsf{g}$ *and any resources* $(t, q)$ *of the adversary there exists "weak" function* $\mathsf{w}$,

$$\mathrm{Adv}^{PRF\text{-}RKA_{\boxplus}}_{\mathsf{w}^{\triangledown}} \approx 3{\cdot}\mathrm{Adv}^{PRF\text{-}RKA_{\boxplus}}_{\mathsf{g}^{\triangledown}},\ \mathrm{Adv}^{PRF\text{-}RKA_{\oplus}}_{\mathsf{w}^{\triangleright}} \approx 3{\cdot}\mathrm{Adv}^{PRF\text{-}RKA_{\oplus}}_{\mathsf{g}^{\triangleright}} = 3{\cdot}\epsilon^{\triangleright}.$$

*If* $\mathsf{w}$ *is used in* Streebog-K *instead of* $\mathsf{g}$*, then there is an attack with distinguishing advantage of about* $\frac{1}{2}\epsilon^{\triangleright} \cdot q \cdot l$.

So we can imply $\epsilon^{\triangleright} \approx t \cdot 2^{-n}$ and obtain the matching attack for the

second term in (7).

It is interesting to note that $\mathsf{w}$ by construction depends on $\epsilon^{\triangleright}$, and hence on the resources of the adversary. All this only emphasizes the artificiality of the approach.

Nevertheless, one way or another, each of the three terms in (7) corresponds to an attack with approximately the same advantage. Hence, the proved security bound is tight.

## 7 HMAC-Streebog

The results obtained were presented using the example of Streebog-K, but the main ideas are applicable to other cryptoalgorithms. Here we briefly present our results for HMAC-Streebog, the proofs are given in Appendix B.

The relation between the keys is defined in HMAC-Streebog by "$\oplus$" and "$\boxplus$" simultaneously. When processing a message, up to 4 related keys are used, two of them are new for each message ($2 \cdot q$ in total), the two remaining ones ($\overline{K} \oplus ipad$ and $\overline{K} \oplus opad$) don't change. The value of $IV$ is queried at least twice under the different related keys ($\mathsf{g}^{\triangledown}_{\overline{K} \oplus ipad}(IV)$ and $\mathsf{g}^{\triangledown}_{\overline{K} \oplus opad}(IV)$), hence, in the $PRF\text{-}RKA_{\boxplus \circ \oplus}$ model, we are bounded by $d = 2$.

HMAC uses two hash function calls and consequently two cascades. When analyzing the "collision" event, we look at the outputs of two transformations at once, so two terms $\mathrm{Adv}^{PRF}_{\mathsf{Csc}}$ arise.

The "$k$-bit security" statement holds for HMAC-Streebog-512, in fact, as well as for Streebog-K. Whereas for the 256-bit version ($\tau = \frac{n}{2}$), the "inner" collision occurs after the first call of the hash function with the probability $\approx q^2 \cdot 2^{-\frac{n}{2}+1}$, that strongly affects the estimate. Thus, we can speak about "$k$-bit security" of HMAC-Streebog-256 only if $q \cdot l < 2^{\frac{n}{2}-k}$.

**Theorem (PRF-security of HMAC-Streebog).** *The advantage of the adversary in the PRF model attacking* HMAC-Streebog *is bounded by*

$$\mathrm{Adv}^{PRF}_{\mathsf{HMAC\text{-}Streebog}}(t,q,l) \leq \mathrm{Adv}^{PRF\text{-}RKA_{\boxplus \circ \oplus}}_{\mathsf{g}^{\triangledown}}(t',q',q',d=2) + \\ + \mathrm{Adv}^{PRF}_{\mathsf{Csc}}(t',q,l') + \mathrm{Adv}^{PRF}_{\mathsf{Csc}}(t',q,l'_\tau) + \frac{2q^2+q}{2^n} + \frac{q^2}{2^{\tau+1}},$$

*where* $t' = t + O(q \cdot l)$, $\tau \in \{256, 512\}$, $q' = 2 \cdot q + 2$, $l' = l + 1$, $l'_\tau \in \{2, 3\}$.

# 8 Conclusion

The security of Streebog-based MACs (including HMAC-Streebog and Streebog-K) as PRF and MAC in the single-key setting is reduced to the security of the compression function in the related key settings ($PRF\text{-}RKA$). We observed that, if the adversary does not query the same input under the different related keys, then the advantage is many times lower than in the general case. An appropriate refinement for the formal model was proposed, and then we re-proved the $PRF$-security of the mentioned MACs based on Streebog. The resulting security bounds are tight and cannot be significantly improved.

In fact, up to $2^{n-k}$ processed blocks, the only effective way of forgery (or distinguishing) is guessing the $k$-bit key or tag, $n = 512$ is the bit length of the hash function state. For HMAC-Streebog-256, this bound is worse and is equal to $2^{\frac{n}{2}-k}$. If the amount of data is much larger than $2^{n-k}$, then the probability of forgery remains insignificant for most practical cases, we just cannot talk about the "ideality".

The new estimates are especially important in practice for the Streebog-based MACs using relatively short keys (for example, 128 bit), and for some lightweight Streebog-like solutions.

The security proofs themselves use only the "standard model" without any heuristics. All statements about "$k$-bit security" are consequences that are obtained under the assumption of "good" properties of the compression function. The latter are confirmed by numerous negative results of cryptanalysis.

As always, we warn that the estimates do not take into account, say, side-channel attacks and other threats not included in the formal model. All the presented results are about adaptively chosen messages attacks in the single-key setting.

# 9 Acknowledgements

The author is grateful to Andrey Shcherbachenko and the anonymous reviewer(s) for the careful consideration of the article and a lot of useful detailed comments and suggestions.

# References


[1] *GOST R 34.11-2012 – National standard of the Russian Federation – Information technology – Cryptographic data security – Hash function*, 2012.

[2] *R 50.1.113-2016 – Information technology – Cryptographic data security – Cryptographic algorithms accompanying the use of electronic digital signature algorithms and hash functions*, 2016.

[3] Smyshlyaev S., Alekseev E., Oshkin I., Popov V., Leontiev S., Podobaev V., Belyavsky D., "RFC 7836 - Guidelines on the Cryptographic Algorithms to Accompany the Usage of Standards GOST R 34.10-2012 and GOST R 34.11-2012", March 2016.

[4] *Secure Hash Standard (SHS) – NIST FIPS – 180-4*, 2015.

[5] *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions – NIST FIPS – 202*, 2015.

[6] Damgård I., "A design principle for hash functions", CRYPTO 1989, Lect. Notes Comput. Sci., **435**, 1990, 416–427.

[7] Merkle R., "One way wash functions and DES", CRYPTO 1989, Lect. Notes Comput. Sci., **435**, 1990, 428–446.

[8] Bellare M., Canetti R., Krawczyk H., "Keying Hash Functions for Message Authentication", Crypto'96, Lect. Notes Comput. Sci., **1109**, 1996, 1–15.

[9] Bellare M., Goldreich O., Mityagin A., "The power of verification queries in message authentication and authenticated encryption", *Cryptology ePrint Archive: Report 2004/304*, 2004.

[10] Bertoni G., Daemen J., Peeters M., Van Assche G., "On the security of the keyed sponge construction", Symmetric Key Encryption Workshop, **2011** (2011).

[11] Koblitz N., Menezes A., "Another look at HMAC", *J. Math. Cryptol.*, **7:3** (2013), 225–251.

[12] Bellare M., "New proofs for NMAC and HMAC: security without collision-resistance", CRYPTO 2006, Lect. Notes Comput. Sci., **4117**, April 2014, 602–619.

[13] Krzysztof Pietrzak, "A Closer Look at HMAC", 2013.

[14] Gaži P., Pietrzak K., Rybár M., "The Exact PRF-Security of NMAC and HMAC", CRYPTO 2014, Lect. Notes Comput. Sci., **8616**, August 2014, 113–130.

[15] Bernstein D.J., Lange T., "Non-uniform cracks in the concrete: the power of free precomputation", ASIACRYPT 2013, Lect. Notes Comput. Sci., **8270**, 2013, 321–340.

[16] Alekseev E.K., Oshkin I.B., Popov V.O., Smyshlyaev S.V., "On the cryptographic properties of algorithms accompanying the applications of standards GOST R 34.11-2012 and GOST R 34.10-2012", *Mat. Vopr. Kriptogr.*, **7**:1 (2016), 5–38.

[17] Nandi M., "A New and Improved Reduction Proof of Cascade PRF", *Cryptology ePrint Archive: Report 2021/097*, 2021.

[18] Bellare M., "Practice-Oriented Provable-Security", ISW 97, Lect. Notes Comput. Sci., **1396**, 1998, 221–231.

[19] Guo J., Jean J., Leurent G., Peyrin T., Wang L., "The usage of counter revisited: second-preimage attack on new Russian standardized hash function", SAC 2014, Lect. Notes Comput. Sci., **8781**, 2014, 195–211.

[20] Dinur I., Leurent G., "Improved generic attacks against hash-based MACs and HAIFA", CRYPTO 2014, Lect. Notes Comput. Sci., **8616**, 2014, 149–168.

[21] Abdelkhalek A., AlTawy R., Youssef A. M., "Impossible differential properties of reduced round Streebog", C2SI 2015, Lect. Notes Comput. Sci., **9084**, 2015, 274–286.

[22] Kiryukhin V., "Streebog compression function as PRF in secret-key settings", *Mat. Vopr. Kriptogr.*, **13**:2 (2022), 99–116.

[23] Kiryukhin V., "Related-key attacks on the compression function of Streebog", CTCrypt 2022.

[24] AlTawy R., Youssef A. M., "Preimage attacks on reduced-round Stribog", AFRICACRYPT 2014, Lect. Notes Comput. Sci., **8469**, 2014, 109–125.

[25] AlTawy R., Kircanski A., Youssef A. M., "Rebound attacks on Stribog", ICISC 2013, Lect. Notes Comput. Sci., **8565**, 2014, 175–188.

[26] Lin D., Xu S., Yung M., "Cryptanalysis of the round-reduced GOST hash function", Inscrypt 2013, Lect. Notes Comput. Sci., **8567**, 2014, 309–322.

[27] Ma B., Li B., Hao R., Li X., "Improved cryptanalysis on reduced-round GOST and Whirlpool hash function", ACNS 2014, Lect. Notes Comput. Sci., **8479**, 2014, 289–307.

[28] Wang Z., Yu H., Wang X., "Cryptanalysis of GOST R Hash Function", *Information Processing Letters*, **114** (2014), 655–662.

[29] Kölbl S., Rechberger C., "Practical attacks on AES-like cryptographic hash functions", LATINCRYPT 2014, Lect. Notes Comput. Sci., **8895**, 2014, 259–273.

[30] Ma B., Li B., Hao R., Li X., "Improved (pseudo) preimage attacks on reduced-round GOST and Grøstl-256 and studies on several truncation patterns for AES-like compression functions", IWSEC 2015, Lect. Notes Comput. Sci., **9241**, 2015, 79–96.

[31] Hua J., Dong X., Sun S., Zhang Z., Hu L., Wang X., "Improved MITM Cryptanalysis on Streebog", *Cryptology ePrint Archive, Paper 2022/568*, 2022.

[32] Kiryukhin V., "Keyed Streebog is a secure PRF and MAC", CTCrypt 2022.

# A Attack on the "weakened" cascade

Let for some arbitrary compression function $\mathsf{g}$

$$\mathrm{Adv}_{\mathsf{g}^{\triangledown}}^{PRF\text{-}RKA_{\boxplus}}(t,q,q,1)=\epsilon^{\triangledown},$$
$$\mathrm{Adv}_{\mathsf{g}^{\triangleright}}^{PRF\text{-}RKA_{\oplus}}(t,q,2)=\epsilon^{\triangleright},$$

As a special illustrative case, we can consider $\epsilon^{\triangledown}=t\cdot 2^{-k}$, $\epsilon^{\triangleright}=2\cdot t\cdot 2^{-n}$, and we also recall that $t\geq q\cdot l$.

We "spoil" two keys $P,P'\in V^n$ in $\mathsf{g}^{\triangledown}$ and $\epsilon^{\triangleright}\cdot 2^n$ keys in $\mathsf{g}^{\triangleright}$. Let $\mathbf{W}\subset V^n$ be the set of "weak" keys, $|\mathbf{W}|\approx\epsilon^{\triangleright}\cdot 2^n$. For any weak key $W\in\mathbf{W}$ and any constant $\Delta_i=\mathbf{i}\oplus(\mathbf{i}\boxplus\mathbf{1})$, $\mathbf{i}=i\cdot n$, we require that $(W\oplus\Delta_i)\in\mathbf{W}$. Each $\Delta_i$ belongs to the set

$$\boldsymbol{\Delta}=\{(2^u-1)\cdot n,\ 1\leq u\leq(n-\log_2(n))\}=\{\mathbf{1},\mathbf{3},\mathbf{7},\mathbf{15},...\}\,.$$

We begin with an empty $\mathbf{W}$, choose an arbitrary $W\notin\mathbf{W}$, add elements from the following set to $\mathbf{W}$:

$$\{W\}\cup\{W\oplus\Delta;\ \Delta\in\boldsymbol{\Delta}\}\cup\{W\oplus\Delta\oplus\Delta';\ \Delta,\Delta'\in\boldsymbol{\Delta}\};$$

continue until there are less than $\epsilon^{\triangleright}\cdot 2^n$ elements in $\mathbf{W}$. At each iteration, no more than $n^2$ elements are added to $\mathbf{W}$. Therefore, the cardinality of $\mathbf{W}$ can differ from $\epsilon^{\triangleright}\cdot 2^n$ only by this insignificant value.

The "weakened" version of the compression function is defined as

$$\mathsf{w}(x,y)=\begin{cases}W_0, & x\in\mathbf{W}\text{ and }y\in\{P,P'\},\\ \mathsf{g}(x,y), & \text{otherwise},\end{cases}$$

where $W_0 \in \mathbf{W}$ is some fixed element. Thus, in total we redefine $2 \cdot |\mathbf{W}|$ values.

If the adversary does not interact with "weak" keys, then $\mathsf{g}^{\triangledown}$ and $\mathsf{w}^{\triangledown}$ (also as $\mathsf{g}^{\triangleright}$ and $\mathsf{w}^{\triangleright}$) are indistinguishable. In the first case ($\mathsf{w}^{\triangledown}_{\overline{K}}(\cdot) = \mathsf{w}(\cdot, \overline{K})$, $K \in V^k$), the interaction is carried out with $q$ related-keys. The probability that there are $P$ or $P'$ among them does not exceed a negligible $\frac{2 \cdot q}{2^k}$. Only two related keys are used in $\mathsf{w}^{\triangleright}_h(\cdot) = \mathsf{w}(h, \cdot)$, $h \in V^n$, the probability of their belonging to the set of weak ones is estimated as $2 \cdot \epsilon^{\triangleright}$. Thus, due to the appearance of weak keys, the distinguishing advantage in both cases increases slightly

$$\mathrm{Adv}^{PRF\text{-}RKA_{\boxplus}}_{\mathsf{w}^{\triangledown}}(t, q, q, 1) \leq \epsilon^{\triangledown} + \frac{2 \cdot q}{2^k} \leq 3 \cdot \epsilon^{\triangledown},$$
$$\mathrm{Adv}^{PRF\text{-}RKA_{\oplus}}_{\mathsf{w}^{\triangleright}}(t, q, 2) \leq \epsilon^{\triangleright} + 2 \cdot \epsilon^{\triangleright} = 3 \cdot \epsilon^{\triangleright}.$$

In distinguishing attack on "weakened" Streebog-K (instantiated with $\mathsf{w}$ instead of $\mathsf{g}$), $\frac{q}{2}$ pairs of queries $(M_i, M'_i)$ to the oracle $\mathcal{O} \in \{\text{Streebog-K}, \mathsf{R}\}$ are made

$$M_i = m_i^{(1)} || m_i^{(2)} || ... || m_i^{(l-2)} || P || P',$$
$$M'_i = m_i^{(1)} || m_i^{(2)} || ... || m_i^{(l-2)} || P' || P,$$

and the tags obtained are $H_i = \mathcal{O}(M_i)$, $H'_i = \mathcal{O}(M'_i)$, $1 \leq i \leq \frac{q}{2}$. The blocks $m_i^{(j)}$ are randomly chosen from $\{P, P'\}$, $1 \leq j \leq l-2$. Note, that $M_i \neq M'_i$, but the first $(l-2)$ blocks, the lengths, and the checksums are equal.

In the case $\mathcal{O} = \text{Streebog-K}$, we assume, as usual, that after processing the $j$-th block, the secret state looks random, but if the state falls into the set $\mathbf{W}$, then it remain as such until the end of the cascade. The longer the message being processed, the more likely it is that a weak key will occur during processing. We recall, that for all $W \in \mathbf{W}$ the following holds

$$\mathsf{w}(W, m_i^{(j)}) = W_0 \in \mathbf{W}, \quad W_0 \oplus \Delta_j \in \mathbf{W},$$
$$\mathsf{w}(W, P) = \mathsf{w}(W, P') = W_0 \in \mathbf{W}.$$

Hence, the probability of the collision $H_i = H'_i$ for one pair $(M_i, M'_i)$ is about $\approx l \cdot \epsilon^{\triangleright}$, and for $\frac{q}{2}$ attempts we have $\approx \frac{q}{2} \cdot l \cdot \epsilon^{\triangleright}$. The collision event is used as a distinguishing feature.

If the attacker interacts with a random function $\mathsf{R}$, then the probability of at least one collision among $\frac{q}{2}$ independent attempts is upper bounded by $\frac{q}{2^n}$. We emphasize that the collision is considered only between messages in the same pair, and not among all possible pairs, and therefore the probability

increases linearly, not quadratically. Hence, the distinguishing advantage is about $\left(\frac{q}{2}\cdot l\cdot\epsilon^{\triangleright}-\frac{q}{2^n}\right)\approx\frac{q}{2}\cdot l\cdot\epsilon^{\triangleright}$.

The resulting lower bound and the upper bound ($q\cdot l\cdot 3\epsilon^{\triangleright}$) differ by about 6 times, this is negligible. Thus, the described extremely synthetic example shows, that the second term in (7) is also tight.

The same is true for "weakened" HMAC-Streebog.

# B HMAC-Streebog

**Theorem (PRF-security of HMAC-Streebog).** *The advantage of the adversary in the PRF model attacking* HMAC-Streebog *is bounded by*

$$\begin{aligned}\mathrm{Adv}^{PRF}_{\mathsf{HMAC\text{-}Streebog}}(t,q,l)\leq&\mathrm{Adv}^{PRF\text{-}RKA_{\boxplus\circ\oplus}}_{\mathsf{g}^{\triangledown}}(t',q',q',d=2)+\\&+\mathrm{Adv}^{PRF}_{\mathsf{Csc}}(t',q,l')+\mathrm{Adv}^{PRF}_{\mathsf{Csc}}(t',q,l'_\tau)+\frac{2q^2+q}{2^n}+\frac{q^2}{2^{\tau+1}},\end{aligned}$$

*where* $t'=t+O(q\cdot l)$, $\tau\in\{256,512\}$, $q'=2\cdot q+2$, $l'=l+1$, $l'_\tau\in\{2,3\}$.

Proof.

Recall that HMAC-Streebog (for compactness, we denote it here as HMAC) is represented as

$$\mathsf{HMAC}(K,M)=\mathsf{H}\left((\overline{K}\oplus opad)||\mathsf{H}(\overline{K}\oplus ipad||M)\right),$$

where $ipad$, $opad\in V^n$, $ipad\neq opad$, $\overline{K}=(K||0...0)\in V^n$, $K\in V^k$. Streebog-512 or Streebog-256 can be used as H (see also figures 2 and 3).

Let the values in the first (resp. the second) call of the hash function be indicated by the superscript "$I$" (resp. "$O$")

$$\begin{aligned}K^I&=\overline{K}\oplus ipad,&K^O&=\overline{K}\oplus opad,\\K^I_{\mathsf{Csc}}&=\mathsf{g}^{\triangledown}_{K^I}(IV),&K^O_{\mathsf{Csc}}&=\mathsf{g}^{\triangledown}_{K^O}(IV),\\H^I&=\mathsf{H}(K^I\,||\,M),&H^O&=\mathsf{H}(K^O\,||\,H^I),\\Y^I&=\mathsf{Csc}(K^I_{\mathsf{Csc}},M),&Y^O&=\mathsf{Csc}(K^O_{\mathsf{Csc}},H^I),\\\sigma^I&=\mathsf{sum}_{\boxplus}(M),&\sigma^O&=\mathsf{sum}_{\boxplus}(H^I).\end{aligned}$$

Just as in the case of Streebog-K, we define "idealized" function

$$\widetilde{\mathsf{HMAC}}(M)=\mathsf{f}_{K^O\boxplus\sigma^O_i}\left(...\mathsf{f}_{K^I\boxplus\sigma^I_i}(\mathsf{Csc}(\mathsf{f}_{K^I\boxplus 0}(IV),M_i))\right),$$

where the first and the last calls of $\mathsf{g}^{\triangledown}$ in both hash functions are replaced by a family of $2^n$ random functions f indexed by $(\phi,\sigma)\in V^n\times V^n$. Related keys can be represented as $(\overline{K}\oplus\phi)\boxplus\sigma$, and $\phi\in\{ipad,opad\}$.

The "collision" (C) here is treated as a coincidence among $(2q+1)$ values

$$IV, Y_1^I, ..., Y_q^I, Y_1^O, ..., Y_q^O.$$

As before, if there is no "collision" ($\overline{\mathrm{C}}$), then $\widetilde{\mathsf{HMAC}}$ is indistinguishable from a random function $\mathsf{R}(\cdot)$,

$$\Pr(\mathcal{A}^{\mathsf{R}(\cdot)} \Rightarrow 1) = \Pr(\mathcal{A}^{\widetilde{\mathsf{HMAC}}(\cdot)} \Rightarrow (1, \overline{\mathrm{C}})),$$

and we obtain $\mathrm{Adv}^{PRF}_{\mathsf{HMAC}}(\mathcal{A}) =$

$$= \Big(\Pr(\mathcal{A}^{\mathsf{HMAC}(\cdot)} \Rightarrow (1, \mathrm{C})) + \Pr(\mathcal{A}^{\mathsf{HMAC}(\cdot)} \Rightarrow (1, \overline{\mathrm{C}}))\Big) - \Pr(\mathcal{A}^{\widetilde{\mathsf{HMAC}}(\cdot)} \Rightarrow (1, \overline{\mathrm{C}})) \leq$$
$$\leq \Big(\Pr(\mathcal{A}^{\mathsf{HMAC}(\cdot)} \Rightarrow (1, \overline{\mathrm{C}})) - \Pr(\mathcal{A}^{\widetilde{\mathsf{HMAC}}(\cdot)} \Rightarrow (1, \overline{\mathrm{C}}))\Big) +$$
$$+ \Big(\Pr(\mathcal{A}^{\mathsf{HMAC}(\cdot)} \Rightarrow \mathrm{C}) - \Pr(\mathcal{A}^{\widetilde{\mathsf{HMAC}}(\cdot)} \Rightarrow \mathrm{C})\Big) + \Pr(\mathcal{A}^{\widetilde{\mathsf{HMAC}}(\cdot)} \Rightarrow \mathrm{C}) =$$
$$= \epsilon + \epsilon_{coll} + p_{coll}.$$

Algorithm $\mathcal{B}_1$ attacks $\mathsf{g}^{\triangledown}$ in the $PRF\text{-}RKA_{\boxplus \circ \oplus}$ model, its actions are also similar to the previous case. Initially, $\mathcal{B}_1$ queries the cascade keys

$$K^I_{\mathsf{Csc}} = \mathcal{O}(IV, (ipad, 0)) \quad \text{and} \quad K^O_{\mathsf{Csc}} = \mathcal{O}(IV, (opad, 0)),$$

from the oracle $\mathcal{O} \in \{\mathsf{g}^{\triangledown}, \mathsf{f}\}$.

When processing each query $M_i$ from $\mathcal{A}$, the algorithm $\mathcal{B}_1$ computes

$$Y_i^I = \mathsf{Csc}(K^I_{\mathsf{Csc}}, M_i) \quad \text{and} \quad \sigma_i^I = \mathsf{sum}_{\boxplus}(M_i).$$

The value of $Y_i^I$ is written in memory. Next, $\mathcal{B}_1$ checks the "collision" condition. If $Y_i^I \in \{IV, Y_1^I, Y_1^O ..., Y_{i-1}^I, Y_{i-1}^O\}$ then $\mathcal{B}_1$ returns 1 and turns off $\mathcal{A}$. Otherwise, $\mathcal{B}_1$ makes query $(Y_i^I, (ipad, \sigma_i^I))$ to the oracle, receives $H_i^I$, computes $Y_i^O = \mathsf{Csc}(K^O_{\mathsf{Csc}}, H_i^I)$ and $\sigma_i^O = \mathsf{sum}_{\boxplus}(H_i^I)$, saves $Y_i^O$ in memory. The "collision" is checked again, if $Y_i^O \in \{IV, Y_1^I, Y_1^O ..., Y_i^I\}$, then $\mathcal{B}_1$ returns 1 and turns off $\mathcal{A}$. Otherwise, $\mathcal{B}_1$ makes query $(Y_i^O, (opad, \sigma_i^O))$ and transmits the response to $\mathcal{A}$.

If the "collision" conditions have never been met after $q$ queries, then the result of $\mathcal{B}_1$ is the result of $\mathcal{A}$. No more than $(2q+2)$ queries are made to the oracle, only the $IV$ value is requested from the oracle twice under different keys, $d = 2$.

The distinguishing advantage of $\mathcal{B}_1$ is equal to

$$\mathrm{Adv}^{PRF\text{-}RKA_{\boxplus \circ \oplus}}_{\mathsf{g}^{\triangledown}}(\mathcal{B}_1) = \Pr(\mathcal{B}_1^{\mathsf{g}^{\triangledown}_{(\overline{K} \oplus \cdot) \boxplus \cdot}(\cdot)} \Rightarrow 1) - \Pr(\mathcal{B}_1^{\mathsf{f}_{(\overline{K} \oplus \cdot) \boxplus \cdot}(\cdot)} \Rightarrow 1) = \epsilon + \epsilon_{coll}.$$

We utilize $p_{coll}$ in the algorithm $\mathcal{B}_2$ to distinguish between a pair of cascades $(\mathsf{Csc}(K^I_{\mathsf{Csc}},\cdot),\mathsf{Csc}(K^O_{\mathsf{Csc}},\cdot))$ and a pair of random functions $(\mathsf{R}^I(\cdot),\mathsf{R}^O(\cdot))$. Recall that in $\widetilde{\mathsf{HMAC}}$, keys $K^I_{\mathsf{Csc}}$ and $K^O_{\mathsf{Csc}}$ are random and independent.

Algorithm $\mathcal{B}_2$ on the $i$-th query $M_i$ from $\mathcal{A}$: makes query $M_i$ to the first oracle ($\mathsf{Csc}(K^I_{\mathsf{Csc}},\cdot)$ or $\mathsf{R}^I(\cdot)$); obtains $Y^I_i$; checks "collision". If so, then, $\mathcal{B}_2$ turns off $\mathcal{A}$ and returns 1. Otherwise, $\mathcal{B}_2$ generates random $H^I_i$ (simulation of $\mathsf{f}$), makes query $H^I_i$ to the second oracle ($\mathsf{Csc}(K^O_{\mathsf{Csc}},\cdot)$ or $\mathsf{R}^O(\cdot)$), and obtains $Y^O_i$ (or finds this value in memory if $H^I_i$ was previously requested). If the "collision" occurs, then $\mathcal{B}_2$ turns off $\mathcal{A}$ and returns 1. Otherwise, $\mathcal{B}_2$ passes the randomly generated $H^O_i$ to $\mathcal{A}$.

If there is no "collision" after $q$ queries, then the result of $\mathcal{B}_2$ is 0.

Interaction with cascades makes it possible to perfectly simulate $\widetilde{\mathsf{HMAC}}$ for $\mathcal{A}$, as long as there is no "collision", hence

$$\Pr(\mathcal{B}_2^{\mathsf{Csc}(K^I_{\mathsf{Csc}},\cdot),\mathsf{Csc}(K^O_{\mathsf{Csc}},\cdot)}\Rightarrow 1)=\Pr(\mathcal{A}^{\widetilde{\mathsf{HMAC}}(\cdot)}\Rightarrow \mathrm{C}).$$

The probability of "collision" in the case when $\mathcal{B}_2$ interacts with $(\mathsf{R}^I(\cdot),\mathsf{R}^O(\cdot))$ is estimated as

$$\Pr(\mathcal{B}_2^{\mathsf{R}^I(\cdot),\mathsf{R}^O(\cdot)}\Rightarrow 1)\le\frac{q^2}{2^{\tau+1}}+\frac{(2q+1)\cdot(2q)}{2^{n+1}},$$

where the first term takes into account the collision among $H^I_1,\ldots H^I_q$. Thus,

$$\begin{aligned}\epsilon_{\mathsf{Csc}}&=\Pr(\mathcal{B}_2^{\mathsf{Csc}(K^I_{\mathsf{Csc}},\cdot),\mathsf{Csc}(K^O_{\mathsf{Csc}},\cdot)}\Rightarrow 1)-\Pr(\mathcal{B}_2^{\mathsf{R}^I(\cdot),\mathsf{R}^O(\cdot)}\Rightarrow 1)\ge\\&\ge\Pr(\mathcal{A}^{\widetilde{\mathsf{HMAC}}(\cdot)}\Rightarrow \mathrm{C})-\left(\frac{q^2}{2^{\tau+1}}+\frac{2q^2+q}{2^n}\right),\\ p_{coll}&=\Pr(\mathcal{A}^{\widetilde{\mathsf{HMAC}}(\cdot)}\Rightarrow \mathrm{C})\le\epsilon_{\mathsf{Csc}}+\left(\frac{q^2}{2^{\tau+1}}+\frac{2q^2+q}{2^n}\right).\end{aligned}$$

In turn, the advantage $\epsilon_{\mathsf{Csc}}$ may simply be bounded by the "hybrid argument"

$$\epsilon_{\mathsf{Csc}}\le\mathrm{Adv}^{PRF}_{\mathsf{Csc}}(\mathcal{B}^I_2)+\mathrm{Adv}^{PRF}_{\mathsf{Csc}}(\mathcal{B}^O_2).$$

The algorithms $\mathcal{B}^I_2$ and $\mathcal{B}^O_2$ make $q$ queries each. Queries from the first algorithm are no longer than $l'=(l+1)$ block. If $\tau=256$ (resp. $\tau=512$) then $\mathcal{B}^O_2$ makes 2-block (resp. 3-block) queries. □

By using the heuristic estimates (1), (6), we obtain $\mathrm{Adv}^{PRF}_{\mathsf{HMAC\text{-}Streebog}}(t,q,l)\lessapprox$

$$\begin{aligned}&\lessapprox\frac{2\cdot t'}{2^k}+\left(\frac{2\cdot t'\cdot q\cdot l'}{2^n}+\frac{q^2}{2^{n+1}}\right)+\left(\frac{2\cdot t'\cdot q\cdot l'_\tau}{2^n}+\frac{q^2}{2^{n+1}}\right)+\frac{2q^2+q}{2^n}+\frac{q^2}{2^{\tau+1}}\le\\&\le\frac{t'}{2^{k-1}}+\frac{t'\cdot q\cdot l''}{2^{n-1}}+\frac{3q^2+q}{2^n}+\frac{q^2}{2^{\tau+1}},\end{aligned}\tag{8}$$

where $l' = l + 1$, $l'_\tau \in \{2, 3\}$, $l'' = l + 4$.

Hence, for the case $\tau = n$, estimate (8) is close to the same (7) for Streebog-K.

However, for the 256-bit version ($\tau = \frac{n}{2}$), the most significant may be the last $\tau$-dependent term. In this case, we can speak about "$k$-bit security" only if $ql < 2^{n-k-1}$ and $q < 2^{\frac{n}{2}-k}$. We don't know the matching forgery attack for this case, but the distinguishing is trivial. For $q = 2^{\frac{n}{2}-1}$, the probability of a collision among the outputs of a random function is about half as low as the corresponding probability for HMAC-Streebog-256.

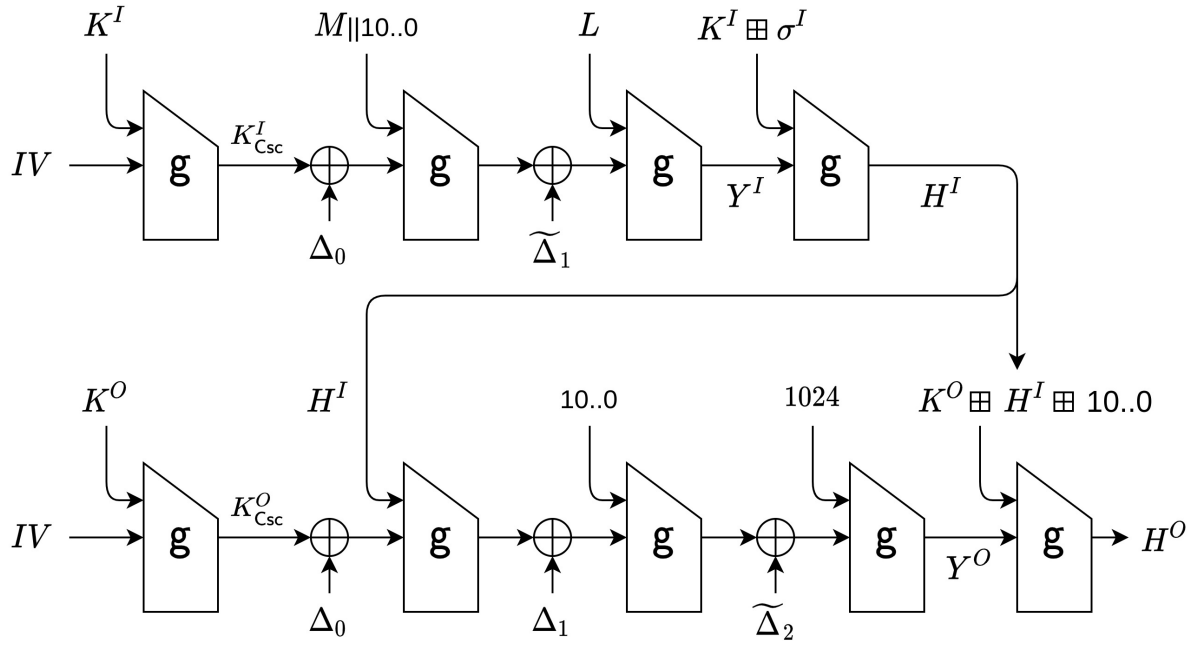



Figure 2: HMAC-Streebog-512 with equivalent representation. The message $M$ consists of $L < 512$ bits.

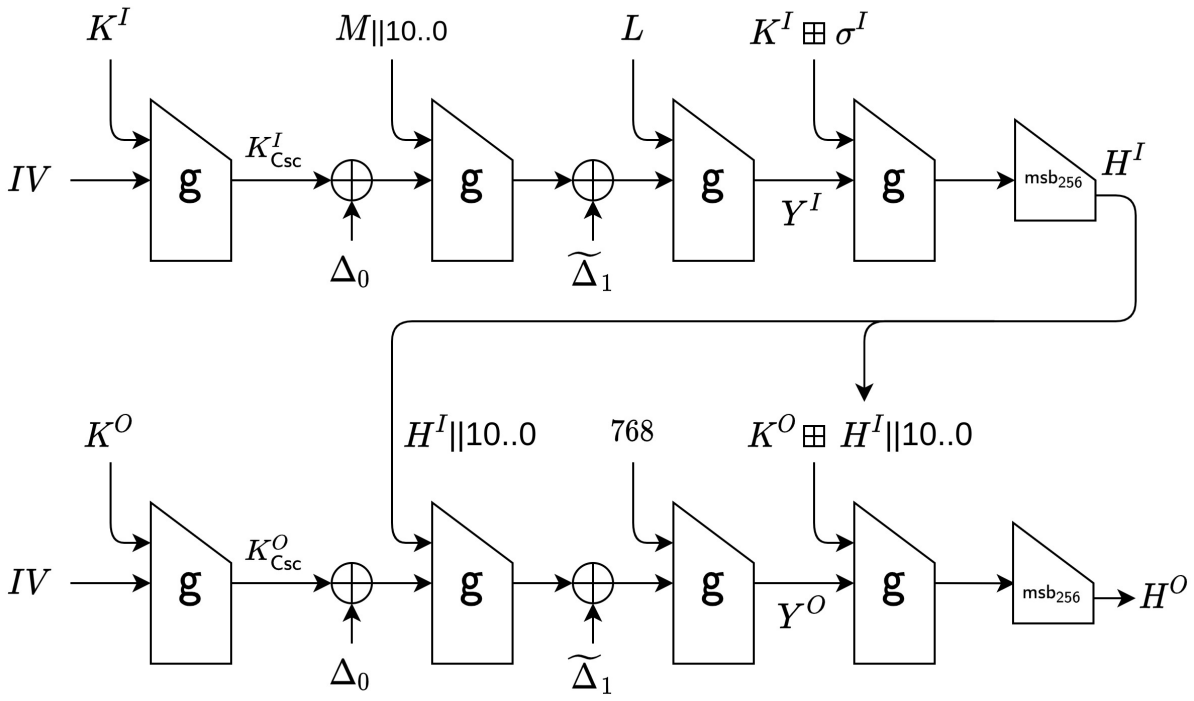



Figure 3: HMAC-Streebog-256 with equivalent representation. The message $M$ consists of $L < 512$ bits.

# Streebog as a Random Oracle

Liliya Akhmetzyanova, Alexandra Babueva, and Andrey Bozhko

CryptoPro LLC, Moscow, Russia
{lah, babueva, bozhko}@cryptopro.ru

**Abstract**

The random oracle model is an instrument used for proving that protocol has no structural flaws when settling with standard hash properties is impossible or fairly difficult. In practice, however, random oracles have to be instantiated with some specific hash functions, which are not random oracles. Hence, in the real world, an adversary has broader capabilities than considered in the random oracle proof — it can exploit the peculiarities of a specific hash function to achieve its goal. In a case when a hash function is based on some building block, one can go further and show that even if the adversary has access to that building block, the hash function still behaves like a random oracle under some assumptions made about the building block. Thereby, the protocol can be proved secure against more powerful adversaries under less complex assumptions. The indifferentiability notion formalizes that approach.

In this paper we study whether Streebog, a Russian standardized hash function, can instantiate a random oracle from that point of view. We prove that Streebog is indifferentiable from a random oracle under an ideal cipher assumption for the underlying block cipher.

**Keywords:** Streebog, GOST, Random Oracle, Indifferentiability

## 1 Introduction

The random oracle model, introduced by Bellare and Rogaway in [9], assumes that every party of the protocol and an adversary has access to a random oracle, which is used instead of a hash function. A random oracle [9] is an ideal primitive which models a random function. It provides a random output for each new query, and identical input queries are given the same answer. The random oracle model allows proving that the protocol does not have any structural flaws in situations when it is impossible or fairly difficult to settle with standard hash properties, which is the case for many efficient and elegant solutions. For example, such protocols and mechanisms as TLS [3], IPSec [2], and Schnorr signature [16, 15] were analyzed in the random oracle model; Russian standardized versions of TLS [4] and IPSec

[6], as well as SESPAKE protocol [5, 8], shortened ElGamal signature [7], to-be-standardized RSBS blind signature [18], and postquantum Shipovnik signature [19] are also analyzed in the random oracle model.

In practice, however, being idealized primitives, random oracles do not exist and have to be instantiated with some specific hash functions, which are not random oracles. Hence, in the real world, an adversary has broader capabilities than considered in the random oracle proof — it can exploit the peculiarities of a specific hash function to achieve its goal. To address such a situation, one can go further and consider the design of the hash function to show that, under some less complex and more specific assumptions than the whole function being a random oracle, it behaves like a random oracle. To do that, one must first understand what "behaves like a random oracle" mean and what assumptions to make.

These questions for a particular class of hash functions are addressed by Coron et al. in [10, 11]. They study the case when an arbitrary-length hash function is built from some fixed-length building block (like an underlying compression function or a block cipher). They come up with a definition, based on the indifferentiability notion of Maurer et al. [14], of what it means to implement a random oracle with such construction, in the assumption that the building block itself is an ideal primitive. The definition is chosen in a way that any hash function satisfying it can securely instantiate a random oracle in a higher-level application (under the assumption that the building block is an ideal primitive). Hence, idealized assumptions are made about less complex and lower-level primitive, and, as a result, more adversarial capabilities are accounted for.

In this paper we study whether Streebog, a Russian standardized hash function [1], can instantiate a random oracle. We recall that Streebog has always been a popular target for analysis. An overview of the results which study standard properties of the algorithm can be found in [17]. A recent paper [13] by Kiryukhin studies keyed version of Streebog as a secure pseudo-random function in a related-key resilient PRF model for an underlying block cipher, highlighting some important high-level design features of Streebog.

Since Streebog is a modified Merkle-Damgard construction based on LSX-style block cipher in Miyaguchi-Preneel mode, we adopt the notion of Coron et al. The paper's main result is presented in Section 3 – we prove that Streebog is indifferentiable from a random oracle under an ideal cipher assumption for the underlying block cipher. We benefit greatly from the work done in [10, 11] since their analysis is focused on Merkle-Damgard constructions with a block cipher in Davis-Meyer mode. However, Streebog's design

features and a different structure of compression function do not allow us to use the paper's results and provoke several challenges.

## 2 Definitions

Let $|a|$ be the bit length of the string $a \in \{0,1\}^*$, the length of an empty string is equal to 0. For a bit string $a$ we denote by $|a|_n = \lceil |a|/n \rceil$ the length of the string $a$ in $n$-bit blocks. Let $0^u$ be the string consisting of $u$ zeroes.

For a string $a \in \{0,1\}^*$ and a positive integer $l \leqslant |a|$ let $\mathrm{msb}_\ell(a)$ be the string, consisting of the leftmost $l$ bits of $a$. For nonnegative integers $l$ and $i$ let $\mathrm{str}_l(i)$ be $l$-bit representation of $i$ with the least significant bit on the right, let $\mathrm{int}(M)$ be an integer $i$ such that $\mathrm{str}_l(i) = M$. For bit strings $a \in \{0,1\}^{\leqslant n}$ and $b \in \{0,1\}^{\leqslant n}$ we denote by $a + b$ a string $\mathrm{str}_n(\mathrm{int}(a) + \mathrm{int}(b) \bmod 2^n)$. If the value $s$ is chosen from a set $S$ uniformly at random, then we denote $s \xleftarrow{\mathcal{U}} S$.

A block cipher $E$ with a block size $n$ and a key size $k$ is the permutation family $\left(E_K \in Perm(\{0,1\}^n) \mid K \in \{0,1\}^k\right)$, where $K$ is a key.

### 2.1 Streebog hash function

The Streebog hash function is defined in [1]. For the purposes of the paper we will define Streebog as a modification of Merkle-Damgard construction, which is applied to a prefix-free encoding of the message; in that we follow the approach of [10, 11]. We will also make the use of the equivalent representation of Streebog from [12]. For Streebog the length of an internal state in Merkle-Damgard construction is $n = 512$ and the length of the output $k$ is either 256 or 512.

Let us define a compression function $h : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$, which is based on 12-rounds LSX-like block cipher $E : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$, where the first argument is a key, in Miyaguchi-Preneel mode:

$$h(y,x) = E(y,x) \oplus x \oplus y.$$

We also define a prefix-free encoding $g : \{0,1\}^* \to (\{0,1\}^n, \{0,1\}^n)^*$, which takes as an input a message $X$:

$$g(X) = (x_1, \Delta_1) \| (x_2, \Delta_2) \| \ldots \| (x'_l \| 10^{n-1-|x'_l|}, \tilde{\Delta}_l) \| (L, 0) \| (\Sigma, 0),$$

where $L = |X|$, $l = \lfloor L/n \rfloor + 1$, $X = x_1 \| \ldots \| x'_l$, where $x_1, \ldots, x_{l-1} \in \{0,1\}^n$, $x'_l \in \{0,1\}^{<n}$ and $x'_l$ is an empty string if $L$ is already divisible by $n$; $\Delta_i = \mathrm{str}_n(i \cdot n) \oplus \mathrm{str}_n((i-1) \cdot n)$, $\tilde{\Delta}_i = \mathrm{str}_n((i-1) \cdot n)$ and $\Sigma =$

$\sum_{i=1}^{l-1} x_i + (x'_l \| 10^{n-1-|x'_l|})$. The encoding pads the message with $10^{n-1-|x'_l|}$, then it splits the message in blocks of length $n$, computes the counter value for each block and appends two last blocks of the encoding, the bit length $L$ and the checksum $\Sigma$, which correspond to the finalizing step of Streebog.

Finally, we define the hash function Streebog on Figure 1, where $IV$, $|IV| = 512$ is a predefined constant, different for $k = 256$ and $k = 512$. On Figure 2 Streebog is depicted schematically.

Streebog$(X)$

$l \leftarrow \lfloor |X|/n \rfloor + 1$
$(x_1, c_1) \| (x_2, c_2) \| \ldots \| (x_l, c_l) \| (x_{l+1}, c_{l+1}) \| (x_{l+2}, c_{l+2}) \leftarrow g(X)$
$y_1 \leftarrow IV$
**for** $i = 1 \ldots l+2$ **do** :
  $y_{i+1} \leftarrow h(y_i, x_i) \oplus c_i$
**return** $\mathrm{msb}_k(y_{l+3})$

Figure 1: Streebog hash function

We will call a sequence of triples $(y_1, x_1, z_1), (y_2, x_2, z_2), \ldots, (y_{l+2}, x_{l+2}, z_{l+2})$, where $z_i = h(y_i, x_i) \oplus y_i \oplus x_i$, which appears during a computation of Streebog on an input $X$, a *computational chain* for $X$.

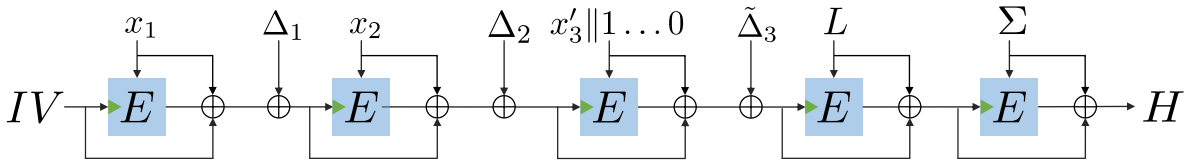



Figure 2: Streebog computation, $l = 3$

## 2.2 Indifferentiability

The following strategy is often applied to prove the security of a cryptosystem with some component (or primitive). One first proves that the system is secure in case of using idealized primitive. Secondly, one proves that the real primitive is indistinguishable from an idealized one. Informally, two algorithms $A$ and $B$ are computationally indistinguishable if no (efficient) algorithm $\mathcal{D}$ is able to distinguish whether it is interacting with $A$ or $B$.

In the current paper we consider two types of the ideal primitives: random oracles and ideal ciphers. A random oracle [9] is an ideal primitive which models a random function. It provides a random output for each new query,

identical input queries are given the same answer. An ideal cipher is an ideal primitive that models a random block-cipher $\mathcal{E} : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$, each key $K \in \{0,1\}^\kappa$ defines a random permutation on $\{0,1\}^n$. The ideal cipher provides oracle access to $\mathcal{E}$ and $\mathcal{E}^{-1}$; that is, on query $(+, K, x)$, it answers $c = E(K, x)$, and on query $(-, K, c)$, it answers $x$ such that $c = E(K, x)$.

Obviously, a random oracle (ideal cipher) is easily distinguishable from a hash function (block cipher) if one knows its program and the public parameter. Thus in [14] the extended notion of indistinguishability — *indifferentiability* — was introduced. It was proven, that if a component $A$ is indifferentiable from $B$, then the security of any cryptosystem $C(A)$ based on $A$ is not affected when replacing $A$ by $B$. According to the authors, indifferentiability is the weakest possible property allowing for security proofs of the generic type described above. Thus, to prove the security of some cryptosystem using hash function we may prove its security in the random oracle model and then prove that hash function is indifferentiable from a random oracle within some underlying assumptions. In the current paper we assume that the base block cipher is modelled as an ideal cipher.

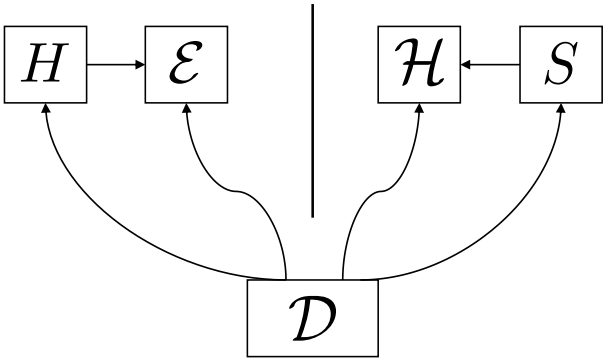



Figure 3: The indifferentiability of hash function $H$ and random oracle $\mathcal{H}$

Let us formally define what does the indifferentiability from an ideal primitive mean. We will give the definition directly for the hash function (based on the ideal cipher) and random oracle. This definition is a particular case of more general indifferentiability notion introduced in [14].

**Definition 1.** *A hash function $H$ with oracle access to an ideal cipher $\mathcal{E}$ is said to be $(T_\mathcal{D}, q_H, q_E, \varepsilon)$-indifferentiable from a random oracle $\mathcal{H}$ if there exists a simulator $S$, such that for any distinguisher $\mathcal{D}$ with binary output it holds that:*

$$\left|\Pr\left[\mathcal{D}^{H,\mathcal{E}} \to 1\right] - \Pr\left[\mathcal{D}^{\mathcal{H},S} \to 1\right]\right| < \varepsilon.$$

*The simulator has oracle access to $\mathcal{H}$. The distinguisher runs in time at most $T_{\mathcal{D}}$ and makes at most $q_H$ and $q_E$ queries to its oracles.*

The indifferentability notion is illustrated at Figure 3. The distinguisher interacts with two oracles, further we denote them by left and right oracles respectively. In the one world left oracle implements the hash function $H$ (with oracle access to the ideal cipher), while the right oracle directly implements the ideal cipher $\mathcal{E}$. In another world the left oracle implements the random oracle $\mathcal{H}$ and the right oracle is implemented by the simulator $S$. The task of the simulator is to model the ideal cipher using the oracle access to $\mathcal{H}$ so that no distinguisher could notice the difference. To achieve that, the output of $S$ should be consistent with what the distinguisher can obtain from $\mathcal{H}$. Note that the simulator does not have access to the queries of the distinguisher to $\mathcal{H}$.

## 3 Streebog indifferentiability

In this section we introduce the main result of the paper, which demonstrates that Streebog is indifferentiable from a random oracle in the ideal cipher model for the base block cipher.

At first, we discuss the choice of the underlying assumption. Indeed, the straightforward solution is to prove Streebog indifferentiability in assumption that the compression function is a random oracle. Although such proof may be constructed much easier than in the ideal cipher model, we show that the Miyaguchi-Preneel compression function cannot be modeled as a random oracle. Indeed, for this function the following condition always holds:

$$x = E^{-1}(y, h(y, x) \oplus x \oplus y).$$

Thus, the distinguisher can easily identify whether it interacts with the real compression function or the random one by making the query $(y, x)$ to the left oracle and the query $(-, y, h(y, x) \oplus x \oplus y)$ to the right oracle.

We give an indifferentiability theorem for Streebog. The full proof is provided for the Streebog variant with output size $k = 512$. For the shortened Streebog variant argumentation is completely similar. Formally, the only thing which has to be adjusted is the construction of the simulator; we will highlight the difference in the proof. The general structure of the proof and some techniques are adopted from [10, 11].

**Theorem 1.** *The hash function* Streebog *with $k = 512$ or 256 using a cipher $E : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ is $(t_D, q_H, q_E, \varepsilon)$-indifferentiable from*

*a random oracle in the ideal cipher model for $E$ for any $t_D$ with*

$$\varepsilon = \frac{q}{2^{n-3}} + \frac{(6+4n)q^2}{2^{n-4}} + \frac{q^3}{2^{n-7}},$$

*where $q = q_E + q_H(l_m + 2)$ and $l_m$ is the maximum message length (in blocks, including padding) queried by the distinguisher to its left oracle.*

*Proof.* The main goal of the proof is to show that no distinguisher can tell apart two words: in the first one, it has access to the Streebog construction using an ideal cipher as an underlying block cipher and to the ideal cipher itself; in the second one it has access to a random oracle and a simulator. The first step of the proof is to present a simulator for which it would be possible to achieve that goal.

Our simulator for the ideal cipher $\mathcal{E}$ is quite elaborate. On every distinguisher query, it tries to detect whether the distinguisher seeks to compute Streebog for some message itself. If that is the case, it chooses the reply consistently with the random oracle; otherwise, it chooses the answer randomly.

**The Simulator.** Before we proceed with the simulator itself, let us define an auxiliary function $g_0 : \{0,1\}^* \to (\{0,1\}^n, \{0,1\}^n)^*$:

$$g_0(X) = (x_1, \Delta_1) \| (x_2, \Delta_2) \| \ldots \| (x'_l \| 10^{n-1-|x'_l|}, \tilde{\Delta}_l) \| (L, 0),$$

where $L = |X|$, $l = \lfloor L/n \rfloor + 1$, $X = x_1 \| \ldots \| x'_l$, where $x_1, \ldots, x_{l-1} \in \{0,1\}^n, x'_l \in \{0,1\}^{<n}$ and $x'_l$ is an empty string if $L$ is already divisible by $n$. Clearly, if $\Sigma = \sum_{i=1}^{l-1} x_i + (x'_l \| 10^{n-1-|x'_l|})$, then $g_0(X) \| (\Sigma, 0) = g(X)$.

The simulator accepts two types of queries: either a forward ideal cipher query $(+, y, x)$, where $x \in \{0,1\}^n$ corresponds to a plaintext and $y \in \{0,1\}^n$ to a cipher key, on which it returns a ciphertext $z \in \{0,1\}^n$; or an inverse query $(-, y, z)$, on which it returns a plaintext $x$. The simulator maintains a table $T$, which contains triples $(y, x, z) \in \{0,1\}^n \times \{0,1\}^n \times \{0,1\}^n$.

*Forward query.* When the simulator gets a forward query $(+, y, x)$ it searches the table $T$ for a triple $(y, x, z)$ for some $z$. It returns $z$ if such triple exists. If there is no such triple, the simulator chooses $z$ randomly, puts the triple $(y, x, z)$ in the table and returns $z$ to the distinguisher. Additionally, in that case the simulator proceeds with the following routine. It searches the table for a sequence $(y_1, x_1, z_1), \ldots, (y_l, x_l, z_l)$ of length $l = \lfloor \mathrm{int}(x)/n \rfloor + 1$ such that:

– there exists $X$ such that $g_0(X) = (x_1, \Delta_1) \| (x_2, \Delta_2) \| \ldots \| (x_l, \tilde{\Delta}_l) \| (x, 0)$;

– it is the case that $y_1 = IV$;

– for each $i = 2, \ldots, l$, it is the case that $y_i = x_{i-1} \oplus y_{i-1} \oplus z_{i-1} \oplus \Delta_{i-1}$;

– it is the case that $y = x_l \oplus y_l \oplus z_l \oplus \tilde{\Delta}_l$.

If such sequence exists, the simulator forms a pair $(y_{l+2}, x_{l+2})$ such that $y_{l+2} = x \oplus y \oplus z$ and $x_{l+2} = \sum_{i=1}^{l-1} x_i + x'_l$, where $X = x_1 \| \ldots \| x'_l$. It is easy to see that $g(X) = (x_1, \Delta_1) \| \ldots \| (x_l, \tilde{\Delta}_l) \| (x, 0) \| (x_{l+2}, 0)$. The simulator does nothing if there already exists a triple $(y_{l+2}, x_{l+2}, z')$ for some $z'$ in the table $T$. Otherwise, it computes $z'$ to form a triple $(y_{l+2}, x_{l+2}, z')$, which will be consistent with a random oracle output on $X$, in advance. To do that it queries the random oracle to get the output $Z = \mathcal{H}(X)$, computes $z' = Z \oplus x_{l+2} \oplus y_{l+2}$ and stores the triple $(y_{l+2}, x_{l+2}, z')$ into the table $T$ [1].

*Inverse query.* On an inverse query $(-, y, z)$ the simulator acts almost similarly. It searches the table $T$ for a triple $(y, x, z)$ for some $x$. It returns $x$ if such triple exists. If there is no such triple, the simulator chooses $x$ randomly, puts the triple $(y, x, z)$ in the table and returns $x$ to the distinguisher. In that case it proceeds with completely the same routine as described above.

We will denote the number of entries in the table $T$ by $q$. It is clear that $q_E \leqslant q \leqslant 2q_E$ since for every adversarial query to $S$ at most one additional record might be added to the table $T$ besides the answer to the query itself.

**Proof of Indifferentiability.** Due to the definition of indifferentiability, if the following inequality holds for every distinguisher $\mathcal{D}$:

$$\left|\Pr\left[\mathcal{D}^{H,\mathcal{E}} \to 1\right] \Pr\left[\mathcal{D}^{\mathcal{H},S} \to 1\right]\right| \leqslant \varepsilon,$$

then the theorem follows. Hence, we have to prove that no distinguisher $\mathcal{D}$ can tell apart these two words unless with the probability $\varepsilon$. We will do that using the game hopping technique, starting in the world with the random oracle $\mathcal{H}$ and the simulator $S$ and moving through the sequence of indistinguishable games to the world with the Streebog construction and the ideal cipher $\mathcal{E}$.

*Game 1 → Game 2.* The Game 1 is the starting point, where $\mathcal{D}$ has access to the random oracle $\mathcal{H}$ and the simulator $S$. In the Game 2 we give $\mathcal{D}$ access to the relay algorithm $R_0$ instead of direct access to $\mathcal{H}$. $R_0$, in its turn, has access to the random oracle and on distinguisher's queries simply answers with $\mathcal{H}(X)$. Let us denote by $G_i$ the events that $\mathcal{D}$ returns 1 in Game $i$. It is clear that $\Pr[G_1] = \Pr[G_2]$.

[1] In $k = 256$ case the simulator first pads $Z$ with 256 randomly chosen bits and then computes $z' = Z \oplus x_{l+2} \oplus y_{l+2}$.

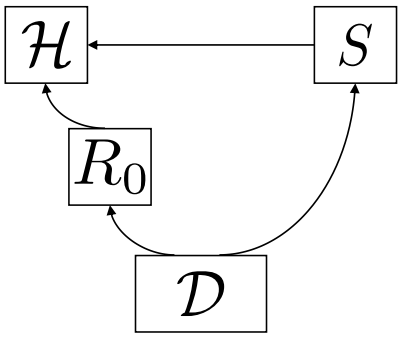



Figure 4: Game 2

*Game 2 → Game 3.* In the Game 3 we modify the simulator $S$ by introducing failure conditions. The simulator explicitly fails (i.e. returns an error symbol $\perp$) while answering the distinguisher's query, if it computes the response satisfying one of the failure conditions below. Let $S_0$ denote the modified simulator.

We introduce two types of failure conditions. Each of the conditions captures different relations between the simulator's answers, which could be exploited by the distinguisher. By failing the simulator 'gives' the distinguisher an immediate win. Our longterm goal is to show that unless the failure happens, distinguisher cannot tell apart Game 2 form the ideal cipher world. The simulator $S_0$ chooses response to the forward or inverse query similarly to the simulator $S$ and then checks the resulting triple $(y, x, z)$ for the conditions defined bellow. For each type of conditions we also provide a brief motivation behind it, i.e., how the distinguisher can exploit corresponding situations to tell apart two worlds.

**Conditions of type 1.** Conditions of type 1 are checked if the answer to the query was chosen randomly or if it is the first time the value, which was chosen by the simulator to be consistent with the random oracle and put in the table earlier, is returned to the distinguisher.

1. *Condition* $B_{11}$. It is the case that $x \oplus y \oplus z = IV$.
2. *Condition* $B_{12}$. It is the case that there exists $l \in [0, q-1]$ such that $x \oplus y \oplus z \oplus \tilde{\Delta}_l = IV$.
3. *Condition* $B_{13}$. It is the case that there exist a triple $(y', x', z') \in T$ and $i \in [1, 2^n]$ such that $x \oplus y \oplus z = x' \oplus y' \oplus z' \oplus \Delta_i$. Note that $|\{\Delta_i,\ i \in [1, 2^n]\}| = n$.

4. *Condition* $B_{14}$. It is the case that there exist a triple $(y', x', z') \in T$ and $l \in [1, q]$ such that $x \oplus y \oplus z = x' \oplus y' \oplus z' \oplus \tilde{\Delta}_l$.

5. *Condition* $B_{15}$. It is the case that there exists a triple $(y', x', z') \in T$ such that $x \oplus y \oplus z = x' \oplus y' \oplus z'$.

The type 1 conditions correspond to a situation when internal states of two Streebog computational chains of different messages collide. The distinguisher can exploit that situation in a number of ways, for example, it can force these two chains to end with the same block, which will give the same result for two different messages. From that, the distinguisher can easily distinguish the two worlds by querying its left oracle with these messages. Other bad situations which correspond to that type of conditions are analyzed in the proof of Lemma 1.

**Conditions of type 2.** Conditions of type 2 are checked if only the answer to the query was chosen by the simulator randomly (i.e., the answer was not taken from the table).

1. *Condition* $B_{21}$. It is the case that there exists a triple $(y', x', z') \in T$ such that $x \oplus y \oplus z = y'$.

2. *Condition* $B_{22}$. It is the case that there exist a triple $(y', x', z') \in T$ and $i \in [1, 2^n]$ such that $x \oplus y \oplus z = y' \oplus \Delta_i$.

3. *Condition* $B_{23}$. It is the case that there exist a triple $(y', x', z') \in T$ and $l \in [1, q]$ such that $x \oplus y \oplus z = y' \oplus \tilde{\Delta}_l$.

The conditions of type 2 correspond to a situation when some block in the computational chain is queried sometime after the query corresponding to the next block was made. In that case, that query may be made even after the query for the last block in the chain was. The distinguisher then can easily tell apart two words since the simulator did not choose the answer to the last query to be consistent with the random oracle. Notice that conditions of that type are only checked when the simulator chooses the answer randomly itself. Otherwise, the distinguisher can easily force the failure event using the random oracle – for example, it can choose an arbitrary $X$, query the random oracle for $Z = \mathcal{H}(X)$, then query the right oracle with $(+, Z, x)$ for some $x$ and finally compute the Streebog construction for $X$ using its right oracle, the simulator would fail then due to condition $B_{21}$ when answering for the last block of the computational chain. However, such a situation will not help the distinguisher since it, in some sense, corresponds to an extension

of a computational chain of some message with new blocks, which will not lead to another valid computational chain due to our prefix-free encoding $g$. Bad situations which correspond to that type of conditions are analyzed in the proof of Lemma 2.

The probability of the event that the simulator fails due to one of the failure conditions is estimated as follows:

$$\Pr[S_0 \text{ fails}] \leqslant \frac{q_E}{2^{n-1}} + \frac{(5+4n)q_E^2}{2^{n-2}} + \frac{q_E^3}{2^{n-5}}.$$

That bound directly follows from Lemma 3 with $q_S = q_E$, which is given in Appendix A. The proof of that statement is rather technical and is also provided in Appendix A.

Since Game 2 and Game 3 are different only in situations, when the simulator $S_0$ fails, it is clear that

$$|\Pr[G_2] - \Pr[G_3]| \leqslant \Pr[S_0 \text{ fails}] \leqslant \frac{q_E}{2^{n-1}} + \frac{(5+4n)q_E^2}{2^{n-2}} + \frac{q_E^3}{2^{n-5}}.$$

Now, before we proceed to the next game, our aim is to show, that unless the simulator fails, its outputs are always consistent with random oracle outputs, i.e. it does not matter if the distinguisher is computing the Streebog construction with its right oracle (maybe in some unsual way) or queries the random oracle, the results would be the same. To do that we prove two lemmas, where Lemma 2 formalizes the outlined goal.

The first lemma states that in the table $T$ there do not exist two sequences of triples, which correspond to computational chains of two different inputs, such that the last block of one chain is the first, middle or last block of the other, unless $S_0$ fails.

**Lemma 1.** *If the simulator $S_0$ does not fail, then there are no two different sequences of triples $(y_1, x_1, z_1), \ldots, (y_{l+2}, x_{l+2}, z_{l+2})$ and $(y'_1, x'_1, z'_1), \ldots, (y'_{p+2}, x'_{p+2}, z'_{p+2})$ in the table $T$ such that the following conditions hold:*

- *there exist $X$ and $X'$ such that $g(X) = (x_1, \Delta_1)\| \ldots \|(x_{l+1}, 0)\|(x_{l+2}, 0)$ and $g(X') = (x'_1, \Delta_1)\| \ldots \|(x'_{p+1}, 0)\|(x'_{p+2}, 0)$;*
- *it is the case that $y_1 = y'_1 = IV$;*
- *for each $i = 2, \ldots, l$ and $j = 2, \ldots, p$, it is the case that $y_i = x_{i-1} \oplus y_{i-1} \oplus z_{i-1} \oplus \Delta_{i-1}$ and $y'_j = x'_{j-1} \oplus y'_{j-1} \oplus z'_{j-1} \oplus \Delta_{j-1}$;*
- *it is the case that $y_{l+1} = x_l \oplus y_l \oplus z_l \oplus \tilde{\Delta}_l$ and $y'_{p+1} = x'_p \oplus y'_p \oplus z'_p \oplus \tilde{\Delta}_l$;*

– *it is the case that* $y_{l+2} = x_{l+1} \oplus y_{l+1} \oplus z_{l+1}$ *and* $y'_{p+2} = x'_{p+1} \oplus y'_{p+1} \oplus z'_{p+1}$*;*

– *there exists* $s \in [1, l+2]$ *such that* $(y_s, x_s, z_s) = (y'_{p+2}, x'_{p+2}, z'_{p+2})$

*Proof.* Let us suppose that there exist two sequences $(y_1, x_1, z_1), \ldots, (y_{l+2}, x_{l+2}, z_{l+2})$ and $(y'_1, x'_1, z'_1), \ldots, (y'_{p+2}, x'_{p+2}, z'_{p+2})$ in the table $T$, which satisfy conditions of the theorem. Then there exists the maximum $r \in [1, \min(s, p+2)]$ such that

$$(y_{s-i}, x_{s-i}, z_{s-i}) = (y'_{p-2-i}, x'_{p-2-i}, z'_{p-2-i}),\ i = 0, \ldots, r-1.$$

In other words, $r$ is the length of subsequence of equal triples which ends with $(y_s, x_s, z_s) = (y'_{p+2}, x'_{p+2}, z'_{p+2})$. We will now consider several cases depending on values of $r$ and $l$. Notice that $r \leqslant s \leqslant l+2$.

**Consider the case** $r = 1$**.** Since it is true that $(y_s, x_s, z_s) = (y'_{p+2}, x'_{p+2}, z'_{p+2})$ we can deduce that one of the following equalities has to hold:

1. if $s = 1$, then $y_s = IV$. Hence, $x'_{p+1} \oplus y'_{p+1} \oplus z'_{p+1} = y'_{p+2} = y_s = IV$;
2. if $s \in [2, l]$, then $y_s = x_{s-1} \oplus y_{s-1} \oplus z_{s-1} \oplus \Delta_{s-1}$. Hence, $x'_{p+1} \oplus y'_{p+1} \oplus z'_{p+1} = x_{s-1} \oplus y_{s-1} \oplus z_{s-1} \oplus \Delta_{s-1}$;
3. if $s = l+1$, then $y_s = x_{s-1} \oplus y_{s-1} \oplus z_{s-1} \oplus \tilde{\Delta}_{s-1}$. Hence, $x'_{p+1} \oplus y'_{p+1} \oplus z'_{p+1} = x_{s-1} \oplus y_{s-1} \oplus z_{s-1} \oplus \tilde{\Delta}_l$;
4. if $s = l+2$, then $y_s = x_{s-1} \oplus y_{s-1} \oplus z_{s-1}$. Hence, $x'_{p+1} \oplus y'_{p+1} \oplus z'_{p+1} = x_{s-1} \oplus y_{s-1} \oplus z_{s-1}$.

However, it is easy to see, that equalities above match failure conditions $B_{11}, B_{13}, B_{14}, B_{15}$ correspondingly. Hence, one of those failure conditions would have been triggered, when a forward or inverse query which corresponds to the triple $(y_{s-1}, x_{s-1}, z_{s-1})$ or $(y'_{p+1}, x'_{p+1}, z'_{p+1})$ (depending on which of them was made later) was made.

**Consider the case** $r \geqslant 2, l > 1$ **and** $r = 3, l = 1$**.** Since $r \geqslant 2$ it is easy to see, that the same inequality holds for $s$. Thereof, from $y'_{p+2} = y_s$ and the theorem statement we have that $x'_{p+1} \oplus y'_{p+1} \oplus z'_{p+1} \oplus 0 = x_{s-1} \oplus y_{s-1} \oplus z_{s-1} \oplus c$ for some $c \in \{\Delta_1, \ldots, \Delta_{l-1}, \tilde{\Delta}_l, 0\}$. However, since from $r \geqslant 2$ we have $(y_{s-1}, x_{s-1}, z_{s-1}) = (y'_{p+1}, x'_{p+1}, z'_{p+1})$, the constant $c$ has to be equal to 0. It is also easy to see that none of the values $\{\Delta_1, \ldots, \Delta_{l-1}, \tilde{\Delta}_l\}$ is equal to 0 when $l > 1$. Hence, due to the encoding $g$, it is only possible that the triple $(y_s, x_s, z_s)$ is the last one in the sequence and $s = l+2$.

Thereof, $x_{l+1} = x'_{p+1}$, where, due to the definition of $g$, $x_{l+1}$ and $x'_{p+1}$ are equal to $|X|$ and $|X'|$ correspondingly. Consequently, since by definition $l = \lfloor |X|/n \rfloor + 1$ and $p = \lfloor |X'|/n \rfloor + 1$, we have that $p = l$.

Finally, consider triples $(y_{l+2-r}, x_{l+2-r}, z_{l+2-r}) \neq (y'_{l+2-r}, x'_{l+2-r}, z'_{l+2-r})$. Notice that $r < l+2$ or else the considered sequences are equal (that excludes the $r = 3, l = 1$ case at all). Since $y_{l+2-r+1} = y'_{l+2-r+1}$ the following equality has to hold:

$$y_{l+2-r} \oplus x_{l+2-r} \oplus z_{l+2-r} \oplus c = y'_{l+2-r} \oplus x'_{l+2-r} \oplus z'_{l+2-r} \oplus c,$$

where $c$ is equal either to $\Delta_{l+2-r}$ or $\tilde{\Delta}_{l+2-r}$. However, it is easy to see that in either way the equality matches the failure condition $B_{15}$. Hence, the it would have been triggered, when a forward or inverse query which corresponds to the triple $(y_{l+2-r}, x_{l+2-r}, z_{l+2-r})$ or $(y'_{l+2-r}, x'_{l+2-r}, z'_{l+2-r})$ (depending on which of them was made later) was made.

**Consider the case $r = 2$ and $l = 1$** In this case we have, that $\tilde{\Delta}_l$ is equal to 0, hence two situations are possible. The first one is when $s = 3$, the reasoning here is completely the same as in the last case since equal triples are the two last triples in the sequences.

The second one is when $s = 2$. From that and since $r = 2$ we have that $(y_1, x_1, z_1) = (y'_{p+1}, x'_{p+1}, z'_{p+1})$. From the theorem statement, $y_1 = IV$ and $y'_{p+1} = x'_p \oplus y'_p \oplus z'_p \oplus \tilde{\Delta}_p$, thereof the following equality has to hold:

$$x'_p \oplus y'_p \oplus z'_p \oplus \tilde{\Delta}_p = IV.$$

However, it is easy to see, that the equality matches the failure condition $B_{12}$. Hence, it would have been triggered, when a forward or inverse query which corresponds to the triple $(y'_p, x'_p, z'_p)$ was made.

Now we notice that we have considered all possible pairs $(r, l)$. Hence, we can conclude that no such sequences can exist if the simulator $S_0$ does not fail. □

Now we prove, that the outputs of the simulator are consistent with the random oracle, unless it fails. To do that we show, that, if the distinguisher at some point computes the Streebog construction itself, it has to do that block-by-block with the last triple of the computational chain being consistent with the random oracle.

**Lemma 2.** *Consider any sequence of triples $(y_1, x_1, z_1), \ldots, (y_{l+2}, x_{l+2}, z_{l+2})$ from the table $T$ such that the following conditions hold:*

– *there exists $X$ such that $g(X) = (x_1, \Delta_1) \| \ldots \| (x_{l+1}, 0) \| (x_{l+2}, 0)$;*

– *it is the case that* $y_1 = IV$*;*

– *for each* $i = 2, \ldots, l$*, it is the case that* $y_i = x_{i-1} \oplus y_{i-1} \oplus z_{i-1} \oplus \Delta_{i-1}$*;*

– *it is the case that* $y_{l+1} = x_l \oplus y_l \oplus z_l \oplus \tilde{\Delta}_l$*;*

– *it is the case that* $y_{l+2} = x_{l+1} \oplus y_{l+1} \oplus z_{l+1}$*.*

*Then, if the simulator* $S_0$ *does not fail, then it must be the case the triples* $(y_1, x_1, z_1), \ldots, (y_{l+1}, x_{l+1}, z_{l+1})$ *were put in the table* $T$ *exactly in that order and answers to the corresponding queries were chosen randomly by the simulator. It also must be that the triple* $(y_{l+2}, x_{l+2}, z_{l+2})$ *was put in the table simultaneously with the triple* $(y_{l+1}, x_{l+1}, z_{l+1})$*, chosen to be consistent with the random oracle output* $\mathcal{H}(X)$*.*

*Proof.* Let us suppose that there exists $i \in [1, \ldots, l+1]$ such that the triple $(y_i, x_i, z_i)$ was put in the table as a result of the corresponding forward of inverse query, when the triple $(y_{i+1}, x_{i+1}, z_{i+1})$ already existed in the table $T$. For that pair of triples the following equality holds:

$$y_i \oplus x_i \oplus z_i \oplus c = y_{i+1},$$

where $c$ is one of the values $\{\Delta_i, \tilde{\Delta}_i, 0\}$, depending on the value of $i$. From Lemma 1 it follows, that the triple $(y_i, x_i, z_i)$ could not be the last one in the computational chain of some message $X' \neq X$. In other words, the answer to the corresponding query was not chosen to be consistent with the random oracle, but chosen randomly by the simulator. Hence, on the query corresponding to the triple $(y_i, x_i, z_i)$ one of the failure conditions of type 2 would have been triggered.

Thereby, when the query corresponding to the triple $(y_{l+1}, x_{l+1}, z_{l+1})$ is made, triples $(y_1, x_1, z_1), \ldots, (y_l, x_l, z_l)$ already exist in the table and the triple $(y_{l+2}, x_{l+2}, z_{l+2})$ does not. These triples satisfy conditions of the simulator's routine and it has to choose the triple $(y_{l+2}, x_{l+2}, z_{l+2})$ to be consistent with the random oracle and put it in the table with the triple $(y_{l+1}, x_{l+1}, z_{l+1})$. □

*Game 3 → Game 4.* In Game 4 we modify the relay algorithm $R_0$. Let $R_1$ denote the modified algorithm. $R_1$ does not have access to the random oracle. On a distinguisher query $X$ it applies the Streebog construction to $X$, using the simulator for the block cipher $E$. Notice that now at most $q_E + q_H(l_m + 2)$ queries are made to $S_0$.

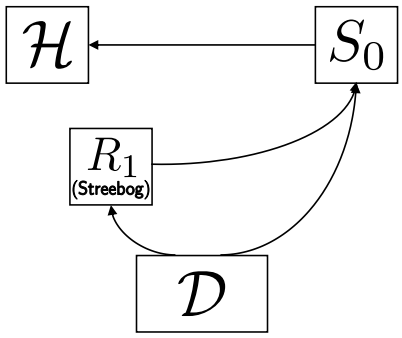



Figure 5: Game 4

Let us denote by $fail_3$ and $fail_4$ the events that the simulator fails in corresponding game. From Lemma 2 it follows that, unless the simulator does not fail, answers of the modified relay algorithm $R_1$ are exactly the outputs of the random oracle on corresponding messages, since the simulator's answers are consistent with the random oracle. Hence, if the simulator does not fail in either world, the view of the distinguisher remains unchanged from Game 3 to Game 4:

$$\Pr\left[G_3|\overline{fail_3}\right] = \Pr\left[G_4|\overline{fail_4}\right].$$

Probability of the event $fail_3$ is estimated earlier in the transition from Game 2 to Game 3. Probability of the event $fail_4$ is estimated from Lemma 3, where $q_S = q_E + q_H(l_m + 2)$. Thus, we have:

$$\begin{aligned}
|\Pr[G_3] - \Pr[G_4]| = \left|\Pr\left[G_3|\overline{fail_3}\right]\Pr\left[\overline{fail_3}\right] + \Pr[G_3|fail_3]\Pr[fail_3] - \Pr\left[G_4|\overline{fail_4}\right]\cdot\right. \\
\left.\cdot\Pr\left[\overline{fail_4}\right] + \Pr[G_4|fail_4]\Pr[fail_4]\right| \leqslant \Pr\left[G_3|\overline{fail_3}\right]\cdot\left|\Pr\left[\overline{fail_3}\right] - \Pr\left[\overline{fail_4}\right]\right| + \\
+ |\Pr[G_3|fail_3]\Pr[fail_3] - \Pr[G_4|fail_4]\Pr[fail_4]| \leqslant |\Pr[fail_4] - \Pr[fail_3]| + \\
+ |\Pr[G_3|fail_3]\Pr[fail_3] - \Pr[G_4|fail_4]\Pr[fail_4]| \leqslant \max\left(\Pr[fail_3], \Pr[fail_4]\right) + \\
+ \max\left(1\cdot\Pr[fail_3] - 0\cdot\Pr[fail_4], 0\cdot\Pr[fail_3] + 1\cdot\Pr[fail_4]\right) \leqslant \\
\leqslant 2\max\left(\Pr[fail_3], \Pr[fail_4]\right) \leqslant \frac{q_E + q_H(l_m+2)}{2^{n-1}} + \frac{(5+4n)(q_E + q_H(l_m+2))^2}{2^{n-2}} + \\
+ \frac{(q_E + q_H(l_m+2))^3}{2^{n-5}}.
\end{aligned}$$

*Game 4 → Game 5.* In Game 5 we modify the simulator. Let $S_1$ denote the modified simulator. $S_1$ does not consult the random oracle when answering the query, it still maintains a table $T$ of triples $(x, y, z)$. On a forward query $(+, y, x)$ it searches the table $T$ for a triple $(y, x, z)$ for some $z$. It returns $z$ if such triple exists. If there is no such triple, the simulator chooses $z$ randomly, puts the triple $(y, x, z)$ in the table and returns $z$ to the distinguisher. It acts similarly to answer the inverse query $(-, y, z)$, but chooses a random $x$, if there is no corresponding triple.

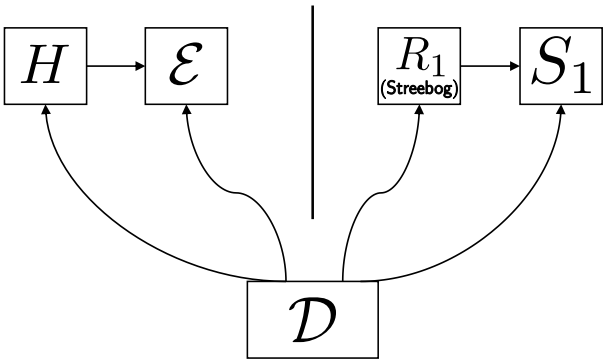



Figure 6: The ideal cipher world and Game 5

The responses of the simulators in these two games are identical, apart from the failure conditions of $S_0$. It is the case since, even when $S_0$ chooses the answer using the random oracle, all its answers look uniformly distributed to the distinguisher as it does not have a direct access to the random oracle in Game 4. Hence, the view of the distinguisher is identical in both games if the simulator does not fail in Game 4, and if in Game 5 the simulator does not give a response, which would have led to failure in Game 4. The probabilities of these events are equal since the number of queries to the simulators is the same in both games, and the distribution of the responses of the simulators is identical. Let us denote the event "$S_1$ should have failed" by $fail_5$. Hence, the following inequality holds:

$$\begin{gathered}|\Pr[G_4] - \Pr[G_5]| \leqslant \left|\Pr\left[G_4|\overline{fail_4}\right]\Pr\left[\overline{fail_4}\right] + \Pr[G_4|fail_4]\Pr[fail_4] - \Pr\left[G_5|\overline{fail_5}\right]\cdot\right. \\ \left.\cdot\Pr\left[\overline{fail_5}\right] + \Pr[G_5|fail_5]\Pr[fail_5]\right| \leqslant |\Pr[G_4|fail_4]\Pr[fail_4] + \Pr[G_5|fail_5]\Pr[fail_5]| \leqslant \\ \leqslant \Pr[fail_4] + \Pr[fail_5] = 2\Pr[fail_4] \leqslant \\ \leqslant 2\left(\frac{q_E + q_H(l_m+2)}{2^{n-1}} + \frac{(5+4n)(q_E + q_H(l_m+2))^2}{2^{n-2}} + \frac{(q_E + q_H(l_m+2))^3}{2^{n-5}}\right).\end{gathered}$$

*Game 5 → Game 6.* In the final game we replace the simulator $S_1$ with the ideal cipher $\mathcal{E}$. Since the relay algorithm $R_1$ is the Streebog construction and now it uses the ideal cipher for $E$, the Game 6 is exactly the ideal cipher model.

We now have to show that the view of the distinguisher remains almost unchanged. The outputs of the ideal cipher and the simulator $S_1$ have different distributions – the ideal cipher is a permutation for each key and $S_1$ chooses its answers randomly. Hence, the distinguisher can tell apart two games only if forward/inverse outputs of the simulator collide for the same key. The probability of that event is at most the birthday bound through all

queries. Thus, we have

$$|\Pr[G_5] - \Pr[G_6]| \leqslant \frac{(q_E + q_H(l_m + 2))^2}{2^n}.$$

Finally, combining all transitions and since Game 6 is exactly the ideal cipher model, we can deduce that

$$\left|\Pr\left[\mathcal{D}^{H,\mathcal{E}} \to 1\right] - \Pr\left[\mathcal{D}^{\mathcal{H},S} \to 1\right]\right| \leqslant \frac{q_E}{2^{n-1}} + \frac{(5+4n)q_E^2}{2^{n-2}} + \frac{q_E^3}{2^{n-5}} +$$
$$+3\left(\frac{q_E + q_H(l_m+2)}{2^{n-1}} + \frac{(5+4n)(q_E + q_H(l_m+2))^2}{2^{n-2}} + \frac{(q_E + q_H(l_m+2))^3}{2^{n-5}}\right) +$$
$$+ \frac{(q_E + q_H(l_m+2))^2}{2^n}.$$

The statement of Theorem 1 hence follows. □

## 4 Conclusion

In the paper we prove that the Streebog hash function is indifferentiable from a random oracle under the ideal cipher assumption for the underlying block cipher. It is still an open problem to determine if it is possible to prove indifferentiability of Streebog and other hash functions under idealized assumptions for even lower-level objects than a block cipher.

## References


[1] GOST R 34.11-2012. National standard of the Russian Federation. Information technology. Cryptographic data security. Hash function, 2012.

[2] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, https://www.rfc-editor.org/info/rfc7296.

[3] Rescorla, E., The Transport Layer Security (TLS) Protocol Version 1.3, RFC 8446, DOI 10.17487/RFC8446, August 2018, https://www.rfc-editor.org/info/rfc8446.

[4] Smyshlyaev, S., Ed., Alekseev, E., Griboedova, E., Babueva, A., and L. Nikiforova, GOST Cipher Suites for Transport Layer Security (TLS) Protocol Version 1.3, RFC 9367, DOI 10.17487/RFC9367, February 2023, https://www.rfc-editor.org/info/rfc9367.

[5] Smyshlyaev, S., Ed., Alekseev, E., Oshkin, I., and V. Popov, The Security Evaluated Standardized Password-Authenticated Key Exchange (SESPAKE) Protocol, RFC 8133, DOI 10.17487/RFC8133, March 2017, https://www.rfc-editor.org/info/rfc8133.

[6] Smyslov, V., Using GOST Ciphers in the Encapsulating Security Payload (ESP) and Internet Key Exchange Version 2 (IKEv2) Protocols, RFC 9227, DOI 10.17487/RFC9227, March 2022, https://www.rfc-editor.org/info/rfc9227.

[7] L. R. Akhmetzyanova, E. K. Alekseev, A. A. Babueva, S. V. Smyshlyaev, “On methods of shortening ElGamal-type signatures”, Mat. Vopr. Kriptogr., 12:2 (2021), 75–91

[8] E. K. Alekseev, S. V. Smyshlyaev, On security of the SESPAKE protocol, Prikl. Diskr. Mat., 2020, no. 50, 5–41

[9] Bellare M., Rogaway P. Random oracles are practical: A paradigm for designing efficient protocols. In Proceedings of the 1st ACM conference on Computer and communications security, pp. 62–73. 1993.

[10] Coron, J. S., Dodis, Y., Malinaud, C., Puniya, P. Merkle-Damgård revisited: How to construct a hash function. In Advances in Cryptology–CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005. Proceedings 25 (pp. 430–448). Springer Berlin Heidelberg.

[11] Coron, J. S., Dodis, Y., Malinaud, C., Puniya, P. Merkle-Damgård revisited: How to construct a hash function. Full version. https://cs.nyu.edu/~dodis/ps/merkle.pdf.

[12] Guo J., Jean J., Leurent G., Peyrin T., Wang L., "The Usage of Counter Revisited: Second-Preimage Attack on New Russian Standardized Hash Function", LNCS, Selected Areas in Cryptography – SAC 2014, 8781, ed. Joux A., Youssef A., Springer, Cham, 2014.

[13] Kiryukhin, V., 2022. Keyed Streebog is a secure PRF and MAC. Cryptology ePrint Archive.

[14] Maurer, U.M., Renner, R., Holenstein, C. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 21–39. Springer, Heidelberg (2004)

[15] Pointcheval, D., Stern, J. (1996). Security proofs for signature schemes. In Advances in Cryptology — EUROCRYPT'96: International Conference on the Theory and Application of Cryptographic Techniques Saragossa, Spain, May 12–16, 1996 Proceedings 15 (pp. 387-398). Springer Berlin Heidelberg.

[16] Schnorr, C.P. (1990). Efficient Identification and Signatures for Smart Cards. In: Brassard, G. (eds) Advances in Cryptology — CRYPTO' 89 Proceedings. CRYPTO 1989. Lecture Notes in Computer Science, vol 435. Springer, New York, NY. https://doi.org/10.1007/0-387-34805-0_22

[17] Smyshlyaev S.V., Shishkin V.A., Marshalko G.B., Rudskoy V.I., Lavrikov I.V., Overview of hash-function GOST R 34.11-2012 cryptoanalysis, Information Security Problems. Computer Systems, 2019

[18] Tessaro S, Zhu C. Short pairing-free blind signatures with exponential security. In Advances in Cryptology–EUROCRYPT 2022: 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30–June 3, 2022, Proceedings, Part II 2022 May 25 (pp. 782-811). Cham: Springer International Publishing.

[19] V. V. Vysotskaya, I. V. Chizhov, "The security of the code-based signature scheme based on the Stern identification protocol", ПДМ, 2022, no. 57, 67–90

# A Probability of the simulator's failure event

**Lemma 3.** *Let $S_0$ be a simulator defined in the proof of Theorem 1. Then the probability of the event that the simulator $S_0$ explicitly fails due to one of the failure conditions $B_{11}, \ldots, B_{23}$, defined in the proof of Theorem 1, satisfies the following bound:*

$$\Pr[S_0 \textit{ fails}] = \frac{q_S}{2^{n-1}} + \frac{(5+4n)q_S^2}{2^{n-2}} + \frac{q_S^3}{2^{n-5}},$$

*where $q_S$ is a number of queries made to the simulator.*

*Proof.* Let us denote by $q$ the maximum number of entries in the table $T$, $q_S \leqslant q \leqslant 2q_S$. To estimate the desired probability, we consider each failure condition and bound the probability that there exists a query to the simulator satisfying the condition. Let us begin with conditions of type 1.

– *Condition* $B_{11}$. It is the propability that one of at most $q$ random $n$-bit strings (where the randomness is due to either the simulator's random choice or the random oracle output) is equal to fixed $IV$. Hence,

$$\Pr[\exists\ query \text{ satisfying } B_{11}] \leqslant \frac{q}{2^n}.$$

– *Condition* $B_{12}$. It is the propability that one of at most $q$ random $n$-bit strings is equal to one of $q$ strings $IV \oplus \tilde{\Delta}_l,\ l \in [0, q-1]$.

$$\Pr[\exists\ query \text{ satisfying } B_{12}] \leqslant \frac{q^2}{2^n}.$$

– *Condition* $B_{13}$. To estimate the probability of that event we will consider three separate situations.

The first one is that there exists a query, satisfying the condition, answer to which was chosen by the simulator randomly. The probability of that situation is the propability that one of at most $q_S \leqslant q$ random $n$-bit strings is equal to one of $nq$ strings $x' \oplus y' \oplus z' \oplus \Delta_i,\ (y', x', z') \in T,\ i \in [1, 2^n]$ (recall that $|\{\Delta_i,\ i \in [1, 2^n]\}| = n$). Hence,

$$\Pr[\exists\ query \text{ satisfying } B_{12} \text{ and Situation 1}] \leqslant \frac{n \cdot q^2}{2^n}.$$

The second one is that there exists a query, satisfying the condition, answer to which was chosen by the simulator to be consistent with the random oracle ($x \oplus y \oplus z$ is exactly the random oracle output then), and the triple $(y', x', z') \in T$ was constructed independently from the random oracle (the answer to the corresponding query was chosen randomly by the simulator itself). The probability of that situation is the propability that one of at most $q_S \leqslant q$ random oracle $n$-bit outputs is equal to one of $nq$ strings $x' \oplus y' \oplus z' \oplus \Delta_i,\ (y', x', z') \in T,\ i \in [1, 2^n]$. Hence,

$$\Pr[\exists\ query \text{ satisfying } B_{12} \text{ and Situation 2}] \leqslant \frac{n \cdot q^2}{2^n}.$$

The third one is that there exists a query, satisfying the condition, answer to which was chosen by the simulator to be consistent with the random oracle, and the triple $(y', x', z') \in T$ was also constructed to be consistent with the random oracle. Then both $x \oplus y \oplus z$ and $x' \oplus y' \oplus z'$ are the random oracle outputs on different messages $X$ and $X'$ (they

are different since both triples have to be the last blocks of some computational chains and there is only one computational chain for every $X$). The probability of that situation is the propability that two random oracle outputs $Z$ and $Z'$ out of at most $q_S \leqslant q$ satisfy any of $n$ equalities $Z \oplus Z' = \Delta_i$. Hence,

$$\Pr[\exists\ query \text{ satisfying } B_{12} \text{ and Situation 3}] \leqslant \frac{n \cdot q^2}{2^n}.$$

Finally, it easy to see that

$$\Pr[\exists\ query \text{ satisfying } B_{12}] \leqslant \Pr[\exists\ query \text{ satisfying } B_{12} \text{ and Situation 1}] + \\ + \Pr[\exists\ query \text{ satisfying } B_{12} \text{ and Situation 2}] + \Pr[\exists\ query \text{ satisfying } B_{12} \text{ and Situation 3}].$$

Hence,

$$\Pr[\exists\ query \text{ satisfying } B_{13}] \leqslant 3 \cdot \frac{n \cdot q^2}{2^n}.$$

– *Condition* $B_{14}$. The probability of that event is estimated similarly to the previous one with the difference that $|\{\tilde{\Delta}_l,\ l \in [1, q]\}| = q$. Hence,

$$\Pr[\exists\ query \text{ satisfying } B_{14}] \leqslant 3 \cdot \frac{q^3}{2^n}.$$

– *Condition* $B_{15}$. The probability of that event is estimated similarly to the two previous ones:

$$\Pr[\exists\ query \text{ satisfying } B_{15}] \leqslant 3 \cdot \frac{q^2}{2^n}.$$

We proceed with conditions of type 2.

– *Condition* $B_{21}$. It is the propability that one of at most $q_S \leqslant q$ random $n$-bit strings, where the randomness is due to either the simulator's random choice or the random oracle output and independent from the distinguisher's random tape, is equal to one of $q$ strings $y',\ (y', x', z') \in T$, where all $y'$ are chosen by the distinguisher. Hence,

$$\Pr[\exists\ query \text{ satisfying } B_{21}] \leqslant \frac{q^2}{2^n}.$$

– *Condition* $B_{22}$. The probability of that event is estimated similarly to the previous one, with the difference that there are at most $nq$ different strings $y' \oplus \Delta_i,\ (y', x', z') \in T,\ i \in [1, 2^n]$. Hence,

$$\Pr[\exists\ query \text{ satisfying } B_{22}] \leqslant \frac{n \cdot q^2}{2^n}.$$

– *Condition* $B_{23}$. The probability of that event is estimated similarly to the previous ones with the difference that there are at most $q^2$ different strings $y' \oplus \tilde{\Delta}_l,\ (y', x', z') \in T,\ l \in [1, q]$. Hence,

$$\Pr[\exists\ query \text{ satisfying } B_{23}] \leqslant \frac{q^3}{2^n}.$$

Finally, we estimate the probability of the event that the simulator fails:

$$\Pr[S_0 \text{ fails}] \leqslant \Pr[\exists\ query \text{ satisfying some bad condition}] \leqslant$$
$$\leqslant \frac{q}{2^n} + \frac{(5+4n)q^2}{2^n} + \frac{4q^3}{2^n} = \frac{q_S}{2^{n-1}} + \frac{(5+4n)q_S^2}{2^{n-2}} + \frac{q_S^3}{2^{n-5}},$$

where the last inequality is due to $q \leqslant 2q_S$. □

# Alternative security models for pseudorandom functions

Kirill Tsaregorodtsev

JSC "NPK Kryptonite"
k.tsaregorodtsev@kryptonite.ru

**Abstract**

In the paper we analyze various security models for pseudorandom functions that arise in the analysis of cryptographic protocols (such as 5G-AKA). We show that these new models can be reduced to the basic PRF model for a pseudorandom function family.

**Keywords:** provable security, pseudorandom function

## Introduction

Pseudorandom functions are one of the main building blocks in cryptography and provable security analysis. Informally, a family of functions $\mathcal{F} = \{\mathcal{F}_k \mid k \in Keys\}$ is pseudorandom, if the function outputs $\mathcal{F}_k(m)$ are indistinguishable from strings of random bits, where $k$ is chosen uniformly at random from the set $Keys$, $m$ are chosen adaptively by the adversary (see Section 1.2 for more details).

In concrete security analysis of various protocols more specific properties of a family $\mathcal{F}$ are sometimes needed. For instance, in 5G-AKA protocol a set of related pseudorandom functions $f_1, \ldots, f_5, f_1^*, f_5^*$ (see Section 1.3) is used; these functions are computed using the same key $k$. The protocol must satisfy a series of properties, each of which imposes restrictions on the pseudorandom functions in question. In this paper we study the reducibility of non-standard pseudorandomness models (obtained in the context of 5G-AKA protocol) to the standard PRF model. The structure of the paper is the following.

– Main definitions and notation to be used are given in Section 1; the description of the standard PRF model is provided (see Section 1.2), as well as the motivation for the proposed new models (Section 1.3).

– PRF$^+$ model (Section 2) formalizes the following requirement: the outputs of a pseudorandom function on adaptively selected inputs must be indistinguishable from random binary strings of the appropriate length, even if the adversary has the opportunity to receive the outputs of a "real" pseudorandom function.

– UF-PRF model (Section 3) formalizes the requirement that it is impossible to forge the value of a pseudorandom function on a fresh input (similar to the models for the MAC function).

– In the context of multi-user systems it is natural to consider models for the case of $d > 1$ users; this generalization is given in Section 4.

– Sometimes in protocols involving entity authentication there are messages that depends on the shared secret key $k$ and cryptographically "binds" (part of) session transcription to the secret key, thereby authenticating the second party. At the same time it is often desirable to provide user privacy from the third parties. Hence, it is important to require the indistinguishability property of "cryptographic bindings" calculated on different keys, which leads to the LOR-PRF model (see Section 5). Note also that in the LOR-PRF model the case of $d > 1$ participants is "basic" and (generally speaking) can not be reduced to the case $d = 1$ (single participant, a degenerate case).

– In Section 6 we briefly discuss the main results of the paper and future possible directions of work.

# 1 Preliminaries

## 1.1 Notations

It this paper the following notation is used. Let $Dom$ and $Range$ be some nonempty sets, then $Funs(Dom, Range)$ is a set of all functions from $Dom$ to $Range$. Most often we will consider the set $Range = \{0,1\}^\ell$; in this case we represent the elements of $x \in Range$ as binary strings (or arrays of bits) of length $\ell$; the notation $x[i]$ in this case means the $i$-th bit of the string $x$. By $[idx_1 \colon idx_2]$, where $idx_1 \le idx_2$, $idx_1, idx_2 \in \mathbb{N}$, we denote a set of natural numbers $\{idx_1, idx_1 + 1, \ldots, idx_2\}$; $S[idx_1 \colon idx_2]$ is a binary substring of $S$ consisting of bits of $S$ with indices $[idx_1 \colon idx_2]$. If $\xi$ is a random variable (distribution), then $x \leftarrow^{\$} \xi$ means choosing a random value according to the distribution $\xi$ and assigning it to the variable $x$. We will also use the

notation $\mathbb{P}[\mathcal{A} \to 1]$ to denote the probability that the randomized algorithm $\mathcal{A}$ returns 1. An empty dictionary (associative array) is denoted as $[\,]$; if $A$ is a dictionary, then $A[s]$ is an element of $A$ with a key $s$. By $b^k$, $b \in \{0,1\}$, we denote a binary string $\underbrace{(b,\ldots,b)}_{k \text{ times}}$. The adversary $\mathcal{A}$ with an oracle access to $\mathcal{O}$ (see, e.g., [1, Section 6.3] for more details) is denoted as $\mathcal{A}^{\mathcal{O}}$. All of the adversaries are considered in some fixed model of computations (e.g., probabilistic Turing machines); by the time (complexity) of the adversary we understand the value that limits the sum of the computational time of the adversary (for example, the number of cycles of calculations) and the size of its program code. This remark is necessary to exclude situations in which some precomputed tables are written into the adversarial code to simplify the partial search.

### 1.2 PRF model

Let us consider a family of functions

$$\mathcal{F} = \{\mathcal{F}_k \in Funs(Dom, Range) \mid k \in Keys\}$$

indexed by a key $k$ from a set $Keys$. As an example of such family we can consider the block cipher "Magma" $E(k,m)$ [2] with a key length of 256 bits and block length of 64 bits:

$$\mathcal{F}_k(m) = E(k,m),\ Keys = \{0,1\}^{256},\ Dom = Range = \{0,1\}^{64},$$

or MAC function $\mathrm{MAC}(k,\cdot)$ (in that case $Dom = \{0,1\}^*$). Let us introduce the standard PRF model (see, e.g., [3, Section 3.5.1]).

**Definition 1.** *The advantage of the adversary $\mathcal{A}$ in the* PRF *model for the function family $\mathcal{F}$ is the following quantity:*

$$\mathrm{Adv}^{\mathrm{PRF}}_{\mathcal{F}}(\mathcal{A}) = \mathbb{P}\big[\mathrm{Exp}^{\text{PRF-1}}_{\mathcal{F}}(\mathcal{A}) \to 1\big] - \mathbb{P}\big[\mathrm{Exp}^{\text{PRF-0}}_{\mathcal{F}}(\mathcal{A}) \to 1\big],$$

*the pseudocode of $\mathrm{Exp}^{\text{PRF-}b}_{\mathcal{F}}$, $b \in \{0,1\}$ is given on Fig. 1.*

**Definition 2.** *Let $\mathrm{Adv}^{\mathrm{PRF}}_{\mathcal{F}}(t,q,\ell,\mu)$ be the be the maximal advantage $\mathrm{Adv}^{\mathrm{PRF}}_{\mathcal{F}}(\mathcal{A})$, where the maximum is taken over the adversaries $\mathcal{A}$ whose time complexity is at most $t$ and with the following restrictions on oracle queries: the number of queries to $\mathcal{O}_{\mathrm{prf}}$ does not exceed $q$, total length of the queries $\sum |m|$ does not exceed $\ell$, maximal query length $\max |m|$ does not exceed $\mu$.*

**Remark 1.** *If the function domain is of the form $Dom = \{0,1\}^{dlen}$ for some $dlen \in \mathbb{N}$, then automatically we have $\mu = dlen$, $\ell = q\,dlen$, hence, these parameters can be omitted, i.e., we can consider the following quantity*

$$\mathrm{Adv}^{\mathrm{PRF}}_{\mathcal{F}}(t,q) = \mathrm{Adv}^{\mathrm{PRF}}_{\mathcal{F}}(t,q,q\,dlen,dlen).$$

| $\mathrm{Exp}_{\mathcal{F}}^{\text{PRF-1}}(\mathcal{A})$ | $\mathrm{Exp}_{\mathcal{F}}^{\text{PRF-0}}(\mathcal{A})$ |
|---|---|
| $k \leftarrow^{\$} Keys$ | $Asked \leftarrow [\,]$ |
| $b' \leftarrow^{\$} \mathcal{A}^{\mathcal{O}_{\text{prf}}}$ | $b' \leftarrow^{\$} \mathcal{A}^{\mathcal{O}_{\text{prf}}}$ |
| **return** $b'$ | **return** $b'$ |
| $\mathcal{O}_{\text{prf}}(m)$ | $\mathcal{O}_{\text{prf}}(m)$ |
| **return** $\mathcal{F}_k(m)$ | **if** $Asked[m] = \perp$ |
| | $Asked[m] \leftarrow^{\$} Range$ |
| | **fi** |
| | **return** $Asked[m]$ |

Figure 1: PRF Experiment for a function family $\mathcal{F}$

## 1.3 Pseudorandom functions in 5G-AKA protocol

In this section we briefly describe the structure of 5G-AKA protocol and show how the protocol requirements are translated to the requirements for the underlying pseudorandom functions.

5G-AKA is a key agreement protocol with explicit authentication of both parties (User and Home Network) and User privacy based on the pse-shared secret key $k$. The main part of the protocol consists of three message transmissions (a more detailed description of the protocol is given in, e.g., [4, 5]):

| **User** | | **Home Network** |
|---|---|---|
| $Request$ | $\longrightarrow$ | |
| | $\longleftarrow$ | $(R, AUTN)$ |
| $RES$ or $AUTS$ | $\longrightarrow$ | |

Some fields of the transmitted messages depends on the secret key $k$. The detailed description of these fields is given in [6] (S3G functions); here we briefly describe them and point out their main purposes.

All functions of $S3G$ are based on the mapping $\mathcal{F}_k(\cdot) = \mathrm{Hash}(k\|\cdot)$ (where Hash is a hash function); the specific fields required in the protocol are obtained by calculating the 512-bit output of the $\mathcal{F}_k$ on various messages with subsequent sampling of bits (see table 1, the bit numbering order is changed compared to [6]).

Table 1: Calculation of values depending on the pre-shared secret $k$

| Value | $S3G$ function | Computation rule | Indices |
|---|---|---|---|
| $\sigma_1$ | $f_1$ | $\mathcal{F}_k(SQN \parallel RAND \parallel Const_1)$ | $[1\colon tlen]$ |
| $\sigma_2$ | $f_1^*$ | $\mathcal{F}_k(SQN_{UE} \parallel RAND \parallel Const_1)$ | $[257\colon 256 + tlen]$ |
| $RES$ | $f_2$ | $\mathcal{F}_k(RAND \parallel Const_2)$ | $[1\colon reslen]$ |
| $AK$ | $f_5$ | | $[257\colon 256 + 48]$ |
| $AK^*$ | $f_5^*$ | | $[305\colon 304 + 48]$ |
| $CK$ | $f_3$ | $\mathcal{F}_k(RAND \parallel Const_3)$ | $[1\colon klen]$ |
| $IK$ | $f_4$ | | $[257\colon 256 + klen]$ |

Constants $Const_i$ depends on the network ID; the length of the values used $tlen$, $reslen$, $klen$; the key length (in bits). $RAND$ value denotes randomness generated by the Home Network (the original 5G-AKA protocol) or jointly by both sides of the interaction (modified version).

- The $\sigma_1$, $\sigma_2$ values (part of the $AUTN$, $AUTS$ resp.) guarantee the integrity of the transmitted messages within the session; also $\sigma_1$ and $\sigma_2$ authenticate the Home Network and the User resp. by implicitly confirming the possession of the shared secret $k$.
- The $RES$ value authenticates the User and confirms the successful completion of authentication on the User's side.
- The $AK$, $AK^*$ values (used in $AUTN$, $AUTS$ resp.) serve as a pseudorandom sequence of bits used to mask the connection counters $SQN$.
- The $CK$, $IK$ values are used for the session key derivation $k_{session}$.

**Remark 2.** *We will consider the function $\mathcal{F}_k(\cdot)$ as a pseudorandom function family (see also [7] for the analysis of this assumption). For S3G functions the input always has a fixed length (depends only on the key length and the index of the function $f_i$). According to the Remark 1 we may not consider the maximum length $\mu$ and total length $\ell$ of the queries.*

The following requirements are valid for 5G-AKA protocol: the indistinguishability of the session key from a random one, even if other values and session keys are compromised; explicit authentication of participants; users privacy.

The first requirement leads to the PRF$^+$ model (see Section 2). The adversary's goal in this model is to distinguish between a segment of a pseudorandom function output and a random string (in the presence of additional

information), which corresponds to the adversary's goal in AKE protocol models: obtaining information about the session key. Learning output segments of a pseudorandom function corresponds to the adversarial ability to compromise session keys in sessions other than the one being attacked, as well as receiving the values of $\sigma_1$, $\sigma_2$, $RES$ (transmitted in plaintext) or partial information about the values of $AK$, $AK^*$.

The second property leads to the UF-PRF model (see Section 3). The adversary's goal in this model is to forge the segment of a pseudorandom function output (in the presence of additional information), which leads to the violation of the participant authentication property. As it was shown previously, the additional information obtained by the adversary corresponds to the real capabilities of the adversary in the context of AKE protocol analysis.

Each of the above models can be generalized naturally to the case of $d$ users (see Section 4): in the 5G-AKA protocol the number of users usually exceeds 1.

The latter security property leads us to the consideration of the LOR-PRF model (see Section 5), the adversary's goal in this model is to determine whether it interacts with the "left" or "right" oracle, which corresponds to adversary's goal in the privacy models for interactive protocols (see [8] and [9, section 5]). At the same time in order to exclude the possibility of trivial attacks, the adversary is not allowed to repeat messages for each specific user. In the 5G-AKA protocol the prohibition on repeating messages is implemented by adding a counter $SQN$ (number of connections) to the messages, as well as the randomness $RAND$.

# 2 PRF$^+$ model

## 2.1 Model description

In the model considered below the adversary must distinguish the output of the pseudorandom function $\mathcal{F}$ from random binary strings (similar to the problem considered in the standard model, see Section 1.2) *in the presence of additional information*: the adversary has the opportunity to obtain segments of the "real" pseudorandom function output $\mathcal{F}_k(m)[idx_1 \colon idx_2]$ on adaptively selected messages $m$ and indices $idx_1 \leq idx_2$. We make the following basic requirement for the non-triviality of the attack: the requested segments should not intersect with the segments on which the adversary is trying to distinguish the output of the function $\mathcal{F}$ from random bits.

**Remark 3.** *To simplify the model we will assume that for any two adversarial queries* $(m, idx_1, idx_2)$ *and* $(m', idx'_1, idx'_2)$ *the following condition is met: either* $m \neq m'$*, or* $[idx_1 \colon idx_2] \cap [idx'_1 \colon idx'_2] = \varnothing$*. Such restrictions do not narrow down the set of adversaries under consideration, since it is always possible to construct an adversary* $\mathcal{B}$ *that satisfies the restrictions. To do this,* $\mathcal{B}$ *will maintain an array of previously requested segments* $S[m, [idx_1 \colon idx_2]]$*, and if the query has the form* $(m, idx'_1 \colon idx'_2)$*, and part of the segment has already been requested earlier (*$[idx_1 \colon idx_2] \cap [idx'_1 \colon idx'_2] \neq \varnothing$*), then* $\mathcal{B}$ *queries the missing part of the segment on the oracle* $\mathcal{O}_{\text{prf}}$ *(see below) and combines it with the previously requested ones. The number of queries to* $\mathcal{O}_{\text{prf}}$ *in this case at most doubles. Note also that for the S3G functions the output segments do not overlap.*

**Definition 3.** *The advantage of* $\mathcal{A}$ *in* $\text{PRF}^+$ *model for the function* $\mathcal{F}$ *is:*

$$\text{Adv}^{\text{PRF}^+}_{\mathcal{F}}(\mathcal{A}) = \mathbb{P}\Big[\text{Exp}^{\text{PRF}^+\text{-}1}_{\mathcal{F}}(\mathcal{A}) \to 1\Big] - \mathbb{P}\Big[\text{Exp}^{\text{PRF}^+\text{-}0}_{\mathcal{F}}(\mathcal{A}) \to 1\Big],$$

*the pseudocode of* $\text{Exp}^{\text{PRF}^+\text{-}b}_{\mathcal{F}}$*,* $b \in \{0, 1\}$*, is given in Fig. 2.*

$\underline{\text{Exp}^{\text{PRF}^+\text{-}b}_{\mathcal{F}}(\mathcal{A})}$

$k \leftarrow^{\$} Keys$
$Asked \leftarrow [\,]$
$b' \leftarrow^{\$} \mathcal{A}^{\mathcal{O}_{\text{prf}}, \mathcal{O}^b_{\text{test}}}$
**return** $b'$

$\underline{\mathcal{O}_{\text{prf}}(m, idx_1, idx_2)}$

**if** $(Asked[m] \cap [idx_1 \colon idx_2] \neq \varnothing)$
  **return** $\perp$
**fi**
$Asked[m] \leftarrow Asked[m] \cup [idx_1 \colon idx_2]$
**return** $\mathcal{F}_k(m)[idx_1 \colon idx_2]$

$\underline{\mathcal{O}^b_{\text{test}}(m, idx_1, idx_2)}$

**if** $(Asked[m] \cap [idx_1 \colon idx_2] \neq \varnothing)$
  **return** $\perp$
**fi**
**if** $(b = 0)$
  $val \leftarrow^{\$} \{0,1\}^{idx_2 - idx_1 + 1}$
**else**
  $val \leftarrow \mathcal{F}_k(m)[idx_1 \colon idx_2]$
**fi**
$Asked[m] \leftarrow Asked[m] \cup [idx_1 \colon idx_2]$
**return** $val$

Figure 2: Pseudocode of $\text{Exp}^{\text{PRF}^+\text{-}b}_{\mathcal{F}}$

**Definition 4.** *Let* $\text{Adv}^{\text{PRF}^+}_{\mathcal{F}}(t, q_{prf}, q_{test})$ *be the maximal value among* $\text{Adv}^{\text{PRF}^+}_{\mathcal{F}}(\mathcal{A})$*, where* $\mathcal{A}$*'s time complexity does not exceed* $t$*,* $\mathcal{A}$ *makes no more than* $q_{prf}$ *queries to the* $\mathcal{O}_{\text{prf}}$ *and* $q_{test}$ *queries to* $\mathcal{O}^b_{\text{test}}$ *oracles.*

## 2.2 Analysis of $\text{PRF}^+$ model

Let us now show that $\text{PRF}^+$ model does not give any extra capabilities to the adversary compared to the standard PRF model.

**Theorem 1.** *The following inequality holds:*

$$\mathrm{Adv}^{\mathrm{PRF}^+}_{\mathcal{F}}(t, q_{prf}, q_{test}) \le 2 \cdot \mathrm{Adv}^{\mathrm{PRF}}_{\mathcal{F}}(t + q_{prf} + q_{test}, q_{prf} + q_{test}).$$

*Proof.* By $\mathcal{A}^{\mathcal{O}_1,\mathcal{O}_2}$ we denote an adversary $\mathcal{A}$ in the $\mathrm{PRF}^+$ model interacting with oracles $\mathcal{O}_1$ and $\mathcal{O}_2$. Then by definition:

$$\mathrm{Adv}^{\mathrm{PRF}^+}_{\mathcal{F}}(\mathcal{A}) = \mathbb{P}\Big[\mathcal{A}^{\mathcal{O}_{\mathrm{prf}},\mathcal{O}^1_{\mathrm{test}}} \to 1\Big] - \mathbb{P}\Big[\mathcal{A}^{\mathcal{O}_{\mathrm{prf}},\mathcal{O}^0_{\mathrm{test}}} \to 1\Big] =$$
$$= \mathbb{P}\Big[\mathcal{A}^{\mathcal{O}_{\mathrm{prf}},\mathcal{O}^1_{\mathrm{test}}} \to 1\Big] - \mathbb{P}\Big[\mathcal{A}^{\$,\mathcal{O}^0_{\mathrm{test}}} \to 1\Big] + \mathbb{P}\Big[\mathcal{A}^{\$,\mathcal{O}^0_{\mathrm{test}}} \to 1\Big] - \mathbb{P}\Big[\mathcal{A}^{\mathcal{O}_{\mathrm{prf}},\mathcal{O}^0_{\mathrm{test}}} \to 1\Big],$$

where by \$ we denote an oracle returning random binary strings as the response. Let us estimate these two summands.

The first one can be upper bounded by $\mathrm{Adv}^{\mathrm{PRF}}_{\mathcal{F}}(t+q_{prf}+q_{test}, q_{prf}+q_{test})$, because it is possible to construct $\mathcal{B}$ in PRF model that uses $\mathcal{A}$ as a subroutine and redirects all requests (both to $\mathcal{O}_{\mathrm{prf}}$ and to $\mathcal{O}^b_{\mathrm{test}}$) to its own oracle $\mathcal{O}^b_{\mathrm{prf}}$ (and also maintains an array $Asked$):

– if $b = 1$, then $\mathcal{B}$ models $\mathrm{Exp}^{\mathrm{PRF}^+\text{-}1}$, i.e. $\mathcal{O}_{\mathrm{prf}}$ and $\mathcal{O}^1_{\mathrm{test}}$ oracles;

– if $b = 0$, then $\mathcal{B}$ models \$ and $\mathcal{O}^0_{\mathrm{test}}$ oracles.

$\mathcal{B}$ returns the same bit as $\mathcal{A}$. Time complexity of $\mathcal{B}$ exceeds time complexity of $\mathcal{A}$ by no more than $q_{prf} + q_{test}$ (we assume that simulating one query takes one step of computation).

The second summand can be bounded by $\mathrm{Adv}^{\mathrm{PRF}}_{\mathcal{F}}(t + q_{prf} + q_{test}, q_{prf})$, because we can construct adversary $\mathcal{B}$ in PRF model that redirects all $\mathcal{A}$'s $\mathcal{O}_{\mathrm{prf}}$ oracle queries to its own $\mathcal{O}^b_{\mathrm{prf}}$ oracle, queries to the $\mathcal{O}^0_{\mathrm{test}}$ can be simulated (i.e., returns random binary strings).

– if $b = 1$, then $\mathcal{B}$ models $\mathrm{Exp}^{\mathrm{PRF}^+\text{-}1}$, i.e. $\mathcal{O}_{\mathrm{prf}}$ and $\mathcal{O}^0_{\mathrm{test}}$ oracles;

– if $b = 0$, then $\mathcal{B}$ models \$ and $Test^0$ oracles.

$\mathcal{B}$ returns the same bit as $\mathcal{A}$. Time complexity of $\mathcal{B}$ can be evaluated similarly. Combining the estimates, we obtain a statement of the theorem. □

## 3 UF-PRF model

In the UF-PRF model considered below an adversary must predict a segment of the $\mathcal{F}_k(m)$ value; the length of the segment to be predicted is $tlen$ bits. The adversary has an opportunity to receive segments of the values of $\mathcal{F}_k(m)[idx_1 \colon id_2]$ on adaptively selected messages $m$ and indices $idx_1 \le$

$idx_2$. The main requirement for the non-triviality of the attack: the requested segments should not intersect with the segments on which the adversary is trying to forge the output of $\mathcal{F}$.

**Definition 5.** *The advantage of $\mathcal{A}$ in* UF-PRF *model for the function $\mathcal{F}$ is:*

$$\mathrm{Adv}_{\mathcal{F}}^{\text{UF-PRF}}(\mathcal{A}) = \mathbb{P}\big[\mathrm{Exp}_{\mathcal{F}}^{\text{UF-PRF}}(\mathcal{A}) \to 1\big],$$

*the pseudocode of* $\mathrm{Exp}_{\mathcal{F}}^{\text{UF-PRF}}$ *is given in Fig. 3.*

$\underline{\mathrm{Exp}_{\mathcal{F}}^{\text{UF-PRF}}(\mathcal{A})}$

$k \leftarrow^{\$} Keys$
$Asked \leftarrow [\,]$
$win \leftarrow false$
$\mathcal{A}^{\mathcal{O}_{\text{prf}},\mathcal{O}_{\text{vfy}}}$
**return** $win$

$\underline{\mathcal{O}_{\text{prf}}(m, idx_1, idx_2)}$

$Asked[m] \leftarrow Asked[m] \cup [idx_1 \colon idx_2]$
**return** $\mathcal{F}_k(m)[idx_1 \colon idx_2]$

$\underline{\mathcal{O}_{\text{vfy}}(m, \tau, i)}$

$val \leftarrow \mathcal{F}_k(m)[i \colon i + tlen - 1]$
$res \leftarrow (\tau = val)$
**if** $(Asked[m] \cap [i \colon i + tlen - 1] = \varnothing)$
  $win \leftarrow win \vee res$
**fi**
**return** $res$

Figure 3: Pseudocode of $\mathrm{Exp}_{\mathcal{F}}^{\text{UF-PRF}}$

**Definition 6.** *Let* $\mathrm{Adv}_{\mathcal{F}}^{\text{UF-PRF}}(t, q_{prf}, q_{vfy}, tlen)$ *be the maximal value among* $\mathrm{Adv}_{\mathcal{F}}^{\text{UF-PRF}}(\mathcal{A})$*, where $\mathcal{A}$'s time complexity does not exceed $t$, the length of the segment to be predicted is $tlen$, $\mathcal{A}$ makes no more than $q_{prf}$ queries to the $\mathcal{O}_{\text{prf}}$ and $q_{vfy}$ queries to $\mathcal{O}_{\text{vfy}}$ oracles.*

### 3.1 Analysis of UF-PRF model

The following result is intuitive: to distinguish a segment of a pseudorandom function from random bits is, generally speaking, much easier than to predict it completely (see also a similar result for the MAC functions [10]).

**Theorem 2.** *The following inequality holds:*

$$\mathrm{Adv}_{\mathcal{F}}^{\text{UF-PRF}}(t, q_{prf}, q_{vfy}, tlen) \leq \\ \leq \mathrm{Adv}_{\mathcal{F}}^{\text{PRF}^+}(t + q_{prf} + q_{vfy}, q_{prf}, q_{vfy}) + \frac{q_{vfy}}{2^{tlen}}.$$

*Proof.* Let $\mathcal{A}$ be an adversary in the UF-PRF model. We construct the adversary $\mathcal{B}$ in the $\text{PRF}^+$ model as follows:

– $\mathcal{A}$'s queries to $\mathcal{O}_{\text{prf}}$ the adversary $\mathcal{B}$ redirects to its own oracle $\mathcal{O}_{\text{prf}}$ unchanged;

– $\mathcal{A}$'s queries to $\mathcal{O}_{\text{vfy}}$ of the form $(m, \tau, i)$ are processed by $\mathcal{B}$ as follows:
  – $\mathcal{B}$ queries $val \leftarrow^{\$} \mathcal{O}^b_{\text{test}}(m, i, i + tlen - 1)$;
  – if $\tau = val$, then $\mathcal{B}$ returns 1 to $\mathcal{A}$, otherwise returns 0.

At the end of the experiment $\mathcal{B}$ returns 1 if and only if the value 1 was returned by $\mathcal{B}$ on one of the queries to $\mathcal{O}_{\text{vfy}}$. If $b = 1$, then $\mathcal{O}^1_{\text{test}}$ returns "real" segments of $\mathcal{F}$ outputs, hence, $\mathcal{B}$ ideally simulated the $\text{Exp}^{\text{UF-PRF}}_{\mathcal{F}}$ experiment, i.e. $\mathbb{P}\left[\text{Exp}^{\text{UF-PRF}}_{\mathcal{F}}(\mathcal{A}) \to 1\right] = \mathbb{P}\left[\text{Exp}^{\text{PRF}^+\text{-1}}_{\mathcal{F}}(\mathcal{B}) \to 1\right]$.

If $b = 0$, then $\mathcal{B}$ can return 1 by a chance (it happens if $\mathcal{A}$ accidentally guesses the segment returned by $\mathcal{O}^0_{\text{test}}$ correctly at least once). The probability of this event can be bounded from above as $\frac{q_{vfy}}{2^{tlen}}$, because each of the equalities $\tau = val$ holds with probability $\frac{1}{2^{tlen}}$. □

**Remark 4.** *The resulting estimate is reasonable, because if $q_{vfy} = 2^{tlen}$, the adversary $\mathcal{A}$ can win in* UF-PRF *experiment with probability 1 by simply going through all possible segments of length tlen.*

**Remark 5.** *By proposition 1 we can reduce the* $\text{PRF}^+$ *model to the* PRF *and obtain the following estimate:*

$$\text{Adv}^{\text{UF-PRF}}(t, q_{prf}, q_{vfy}, tlen) \leq \\ \leq 2 \cdot \text{Adv}^{\text{PRF}}(t + 2\,(q_{prf} + q_{vfy}), q_{prf} + q_{vfy}) + \frac{q_{vfy}}{2^{tlen}}.$$

## 4 $\text{PRF}^+$ and UF-PRF models with $d$ users

In this section we generalize $\text{PRF}^+$ and UF-PRF models to the case of $d$ users (we denote them as $\text{PRF}^+(d)$ and UF-PRF$(d)$ resp.). We consider the oracles $\mathcal{O}_{\text{prf}}$ and $\mathcal{O}_{\text{vfy}}/\mathcal{O}^b_{\text{test}}$ with an additional input $i \in \{1, \ldots, d\}$: in this case messages inside the oracles will be processed on the key $k_i$. We also assume that the keys of the participants $k_1, \ldots, k_d$ are independent and identically distributed. It can be shown that models with $d$ users can be reduced to the corresponding single-user models: informally speaking, due to the independence of the keys, the values depending on the keys $k_1, \ldots, k_{i-1}, k_{i+1}, \ldots, k_d$ do not give any information about the key $k_i$.

**Definition 7.** *Let* $\text{Adv}^{\text{PRF}^+}_{\mathcal{F}}(t, Q_{prf}, Q_{test}; d)$ *be the maximal value among* $\text{Adv}^{\text{PRF}^+(d)}_{\mathcal{F}}(\mathcal{A})$*, where $\mathcal{A}$'s time complexity does not exceed $t$, $\mathcal{A}$ has the following restrictions on oracle queries ($1 \leq i \leq d$):*

– *number of queries of the type* $(i, \ldots)$ *to* $\mathcal{O}_{\text{prf}}$ *oracle does not exceed* $Q_{prf}[i]$*;*

– *number of queries of the type* $(i, \ldots)$ *to* $\mathcal{O}^b_{\text{test}}$ *oracle does not exceed* $Q_{test}[i]$*.*

**Definition 8.** *Let* $\text{Adv}^{\text{UF-PRF}}_{\mathcal{F}}(t, Q_{prf}, Q_{vfy}, tlen; d)$ *be the maximal value among* $\text{Adv}^{\text{UF-PRF}(d)}_{\mathcal{F}}(\mathcal{A})$*, where* $\mathcal{A}$*'s time complexity does not exceed* $t$*,* $\mathcal{A}$ *has the following restrictions on oracle queries* $(1 \le i \le d)$*:*

– *number of queries of the type* $(i, \ldots)$ *to* $\mathcal{O}_{\text{prf}}$ *oracle does not exceed* $Q_{prf}[i]$*;*

– *number of queries of the type* $(i, \ldots)$ *to* $\mathcal{O}_{\text{vfy}}$ *oracle does not exceed* $Q_{vfy}[i]$*;*

– *the length of the segment to be predicted is* $tlen$*.*

The following two statements are simple consequences of the hybrid argument.

**Theorem 3.** *The following inequality holds:*

$$\text{Adv}^{\text{PRF}^+(d)}_{\mathcal{F}}(t, Q_{prf}, Q_{test}; d) \le \\ \le d \cdot \text{Adv}^{\text{PRF}^+}_{\mathcal{F}}(t + T, \max_i Q_{prf}[i], \max_i Q_{test}[i]),$$

*where* $T = \sum_i (Q_{prf}[i] + Q_{test}[i])$*.*

*Proof.* Since the keys $k_i$ are independent, we can use a hybrid argument. Namely, we build a series of adversaries $\mathcal{B}_i$, each of which generates keys $k_j$, $j \ne i$, redirects queries of the form $(i, \ldots)$ to its own oracles $\mathcal{O}_{\text{prf}}$ and $\mathcal{O}^b_{\text{test}}$, and models the rest as follows:

– queries of the form $(j, \ldots)$, $j < i$ to $\mathcal{O}^b_{\text{test}}$ the adversary $\mathcal{B}_i$ processes according to the definition of $\mathcal{O}^0_{\text{test}}$ oracle on key $k_j$;

– queries of the form $(j, \ldots)$, $j > i$ to $\mathcal{O}^b_{\text{test}}$ the adversary $\mathcal{B}_i$ processes according to the definition of $\mathcal{O}^1_{\text{test}}$ oracle on key $k_j$;

– queries of the form $(j, \ldots)$, $j \ne i$ to $\mathcal{O}_{\text{prf}}$ the adversary $\mathcal{B}_i$ processes according to the definition of $\mathcal{O}_{\text{prf}}$ oracle on the key $k_j$.

| $(id < i, m, idx_1, idx_2)$ | $(id = i, m, idx_1, idx_2)$ | $(id > i, m, idx_1, idx_2)$ |
|---|---|---|
| Simulating $\mathcal{O}_{\text{prf}}$ and $\mathcal{O}^0_{\text{test}}$ | Redirecting queries to $\mathcal{O}_{\text{prf}}$ and $\mathcal{O}^b_{\text{test}}$ | Simulating $\mathcal{O}_{\text{prf}}$ and $\mathcal{O}^1_{\text{test}}$ |

Time $T$ needed to simulate the experiment can be estimated as follows (we assume here that processing of one query takes one unit of time):

$$T \leq \sum_i \left(Q_{prf}[i] + Q_{test}[i]\right).$$

The advantage of each of the $\mathcal{B}_i$ can be bounded from above by

$$\mathrm{Adv}_{\mathcal{F}}^{\mathrm{PRF}^+}(t+T, \max_i Q_{prf}[i], \max_i Q_{test}[i]),$$

hence the result. □

**Theorem 4.** *The following inequality holds:*

$$\mathrm{Adv}_{\mathcal{F}}^{\text{UF-PRF}}(t, Q_{prf}, Q_{vfy}, tlen; d) \leq \\ \leq d \cdot \mathrm{Adv}_{\mathcal{F}}^{\text{UF-PRF}}(t+T, \max_i Q_{prf}[i], \max_i Q_{vfy}[i], tlen),$$

*where* $T = \sum_i \left(Q_{prf}[i] + Q_{vfy}[i]\right)$.

*Proof.* Analogously to the theorem above we can use the following form of hybrid argument. The adversary $\mathcal{B}$ chooses random key index $i \leftarrow^{\$} \{1, \ldots, d\}$ for which it redirects queries to its own oracles and simulates oracle answers on the rest of the keys by itself. □

**Remark 6.** *Let us note that the degradation of the estimate by d times when considering an Experiment with d participants is in some sense inevitable: for instance, in the paper [11, Section 2.3] it was shown that when one considers a MAC scheme for d users, there is an adversary who effectively uses the fact that there are more than one participant and is able to increase the basic advantage up to the factor of d.*

## 5 LOR-PRF model

In LOR-PRF model the adversary has to determine on which of the keys ("left" $k_{i_0}$ or "right" $k_{i_1}$) and which message ("left" $m_0$ or "right" $m_1$) is processed by the oracle $\mathcal{O}_{\mathrm{lor}}^b$. At the same time to exclude the possibility of trivial attacks the adversary is not allowed to repeat messages for each fixed key $k_i$.

**Remark 7.** *Without this kind of restrictions the adversary's task becomes trivial: it is enough to query two pairs of messages* $(m, i_0, m, i_1)$ *and* $(m, i_0, m', i_1)$*, where* $m' \neq m$*. If the answers are different, then the oracle processes "right" messages, otherwise "left" are processed.*

**Definition 9.** *The advantage of $\mathcal{A}$ in* LOR-PRF *model (with $d$ users) for the function $\mathcal{F}$ is:*

$$\mathrm{Adv}^{\text{LOR-PRF}}_{\mathcal{F}}(\mathcal{A}) = \mathbb{P}\big[\mathrm{Exp}^{\text{LOR-PRF-1}}_{\mathcal{F}}(\mathcal{A}) \to 1\big] - \mathbb{P}\big[\mathrm{Exp}^{\text{LOR-PRF-0}}_{\mathcal{F}}(\mathcal{A}) \to 1\big],$$

*the pseudocode of* $\mathrm{Exp}^{\text{LOR-PRF-}b}_{\mathcal{F}}$, $b \in \{0,1\}$, *is given in Fig. 4.*

| $\mathrm{Exp}^{\text{LOR-PRF-}b}_{\mathcal{F}}(\mathcal{A})$ | $\mathcal{O}^b_{\text{lor}}(m_0, i_0, m_1, i_1)$ |
|---|---|
| **for** $i \in \{1, \ldots, d\}$ | **if** $(m_0 \in Msg[i_0]) \vee (m_1 \in Msg[i_1])$ |
| $k_i \xleftarrow{\$} Keys$ | **return** $\perp$ |
| **endfor** | **fi** |
| $Msg \leftarrow [\,]$ | $Msg[i_0] \leftarrow Msg[i_0] \cup \{m_0\}$ |
| $b' \xleftarrow{\$} \mathcal{A}^{\mathcal{O}^b_{\text{lor}}}$ | $Msg[i_1] \leftarrow Msg[i_1] \cup \{m_1\}$ |
| **return** $b'$ | **return** $\mathcal{F}_{k_{i_b}}(m_b)$ |

Figure 4: Pseudocode of $\mathrm{Exp}^{\text{LOR-PRF-}b}_{\mathcal{F}}$

**Definition 10.** *Let* $\mathrm{Adv}^{\text{LOR-PRF}}_{\mathcal{F}}(t, Q; d)$ *be the maximal value among* $\mathrm{Adv}^{\text{LOR-PRF}}_{\mathcal{F}}(\mathcal{A})$ *in* LOR-PRF *Experiment with $d$ users, where $\mathcal{A}$'s time complexity does not exceed $t$, the number of queries to $\mathcal{O}^b_{\text{lor}}$ oracle on the key $k_i$ (either as "left", or as "right", i.e., queries of the form $(\cdot, i, \cdot, \cdot)$ or $(\cdot, \cdot, \cdot, i)$) does not exceed $Q[i]$.*

## 5.1 Analysis of LOR-PRF model

In this section we show that the LOR-PRF model introduced above does not give any additional opportunities to the adversary compared with the standard PRF model (see Section 1.2).

**Theorem 5.** *The following inequality holds:*

$$\mathrm{Adv}^{\text{LOR-PRF}}_{\mathcal{F}}(t, Q; d) \le 2d \cdot \mathrm{Adv}^{\text{PRF}}_{\mathcal{F}}(t + d + \sum_i Q[i], \max_i Q[i]).$$

*Proof.* Let $\mathcal{A}$ be an adversary in LOR-PRF model. By $\mathcal{A}^b$, $b \in \{0,1\}$, we denote the adversary $\mathcal{A}$ that interacts with the oracle $\mathcal{O}^b_{\text{lor}}$ in the experiment LOR-PRF. We introduce a series of adversaries $\mathcal{B}^{b_1,\ldots,b_d}_{b_0}(\mathcal{A})$, which will process the inputs $(m_0, i_0, m_1, i_1)$ as follows:

- if $b_0 = 0$, $b_{i_0} = 0$: return a random string of appropriate length;
- if $b_0 = 0$, $b_{i_0} = 1$: return $\mathcal{F}_{k_{i_0}}(m_0)$;
- if $b_0 = 1$, $b_{i_1} = 0$: return a random string of appropriate length;

– if $b_0 = 1$, $b_{i_0} = 1$: return $\mathcal{F}_{k_{i_1}}(m_1)$;

Essentially, the bit $b_0$ sets whether "left" or "right" messages are processed by the adversary $\mathcal{B}$, the bit $b_i$, $i \in \{1, \ldots, d\}$ specifies what will be used as the $i$-th function: a truly random or pseudorandom function. The adversaries $\mathcal{B}_{b_0}^{b_1,\ldots,b_d}(\mathcal{A})$ return the same bit as $\mathcal{A}$. The advantage of $\mathcal{A}$ in LOR-PRF model can be written as follows:

$$\mathrm{Adv}_{\mathcal{F}}^{\text{LOR-PRF}}(\mathcal{A}) = \mathbb{P}\big[\mathcal{A}^1 \to 1\big] - \mathbb{P}\big[\mathcal{A}^0 \to 1\big] = \\ = \mathbb{P}\Big[\mathcal{B}_1^{1^d}(\mathcal{A}) \to 1\Big] - \mathbb{P}\Big[\mathcal{B}_0^{1^d}(\mathcal{A}) \to 1\Big].$$

Let us subtract and add a set of identical terms:

$$\mathbb{P}\Big[\mathcal{B}_1^{1^d}(\mathcal{A}) \to 1\Big] - \mathbb{P}\Big[\mathcal{B}_0^{1^d}(\mathcal{A}) \to 1\Big] = \\ \Big(\mathbb{P}\Big[\mathcal{B}_1^{1^d} \to 1\Big] - \mathbb{P}\Big[\mathcal{B}_1^{1^{d-1}0} \to 1\Big]\Big) + \Big(\mathbb{P}\Big[\mathcal{B}_1^{1^{d-1}0} \to 1\Big] - \mathbb{P}\Big[\mathcal{B}_1^{1^{d-2}0^2} \to 1\Big]\Big) + \ldots \\ \ldots + \Big(\mathbb{P}\Big[\mathcal{B}_1^{10^{d-1}} \to 1\Big] - \mathbb{P}\Big[\mathcal{B}_1^{0^d} \to 1\Big]\Big) + \Big(\mathbb{P}\Big[\mathcal{B}_1^{0^d} \to 1\Big] - \mathbb{P}\Big[\mathcal{B}_0^{0^d} \to 1\Big]\Big) + \\ + \Big(\mathbb{P}\Big[\mathcal{B}_0^{0^d} \to 1\Big] - \mathbb{P}\Big[\mathcal{B}_0^{0^{d-1}1} \to 1\Big]\Big) + \Big(\mathbb{P}\Big[\mathcal{B}_0^{0^{d-1}1} \to 1\Big] - \mathbb{P}\Big[\mathcal{B}_0^{0^{d-2}11} \to 1\Big]\Big) + \ldots \\ \ldots + \Big(\mathbb{P}\Big[\mathcal{B}_0^{1^{d-1}0} \to 1\Big] - \mathbb{P}\Big[\mathcal{B}_0^{1^d} \to 1\Big]\Big).$$

Note that each term of the form

$$\Big(\mathbb{P}\Big[\mathcal{B}_1^{1^{d-i}0^i} \to 1\Big] - \mathbb{P}\Big[\mathcal{B}_1^{1^{d-i-1}0^{i+1}} \to 1\Big]\Big)$$

is upper bounded by the value $\mathrm{Adv}_{\mathcal{F}}^{\text{PRF}}(t + d + \sum_i Q[i], \max_i Q[i])$, since it is possible to construct the adversary $\mathcal{C}$ in PRF model, which processes only the "right" messages $m_1$; queries for the first $d - i - 1$ indices are processed on randomly selected keys $k_j$, $1 \le j \le d - i - 1$, queries under the key $k_{d-i}$ the adversary $\mathcal{C}$ redirects to its own oracle $\mathcal{O}_{\text{prf}}^b$, queries for the last $i$ indices are generated randomly equiprobably (lazy generation of a random function). The time required to simulate the experiment does not exceed $d + \sum_i Q[i]$ (key generation and query processing). Analogously, we can bound the value

$$\Big(\mathbb{P}\Big[\mathcal{B}_0^{0^{d-i}1^i} \to 1\Big] - \mathbb{P}\Big[\mathcal{B}_0^{0^{d-i-1}1^{i+1}} \to 1\Big]\Big)$$

by $\mathrm{Adv}_{\mathcal{F}}^{\text{PRF}}(t + d + \sum_i Q[i], \max_i Q[i])$.

The remaining summand $\Big(\mathbb{P}\Big[\mathcal{B}_1^{0^d} \to 1\Big] - \mathbb{P}\Big[\mathcal{B}_0^{0^d} \to 1\Big]\Big)$ corresponds to the situation when all queries to the oracle under any of the keys $k_i$ must return random strings, which means that the oracle's responses are statistically independent of the bit $b_0$, and the difference in question is 0. Collecting together all the inequalities obtained, we get a statement of the theorem. □

# 6 Conclusion

In this paper we consider a series of models ($\mathrm{PRF}^+$ with one and $d$ user(s), UF-PRF with one and $d$ user(s), LOR-PRF with $d$ users) that formalize different security properties of pseudorandom functions. We prove that the security in these models follows from the security of a function family in a standard PRF model. Hence, the adversary gives no extra power compared to the standard assumptions. The results obtained in the paper may be used to estimate the security properties of 5G-AKA protocol (and its variations).

# References


[1] M. Sipser, *Introduction to the Theory of Computation (3rd Edition)*, Cengage Learning, 2012, 504 pp.

[2] *GOST 34.12-2018. Information technology. Cryptographic data security. Block ciphers*, 2018, In Russian.

[3] J. Katz, Y. Lindell, *Introduction to modern cryptography*, CRC press, Boca Raton, Florida, 2020, 626 pp.

[4] 3GPP, *Technical specification (TS). Security architecture and procedures for 5G System (3GPP TS 33.501 version 17.5.0 Release 17).*, 2022.

[5] V. Belsky, A. Drynkin, S. Davydov, "A subscriber's Privacy on the 5G Radio Interface", *International Journal of Open Information Technologies*, **9**:7 (2021), 32–54, In Russian.

[6] *Recommendations for standardization "IT.KZI. Cryptographic algorithms for generating encryption keys and authentication vectors intended for implementation in hardware security modules for use in mobile radiotelephone communication"*, 2023, In Russian.

[7] V. Kiryukhin, "Keyed Streebog is a secure PRF and MAC", 2022, `https://eprint.iacr.org/2022/972`.

[8] J. Hermans, A. Pashalidis, F. Vercauteren, B. Preneel, "A new RFID privacy model", European symposium on research in computer security, 2011, 568–587.

[9] A. Koutsos, "The 5G-AKA Authentication Protocol Privacy", 2019 IEEE European Symposium on Security and Privacy (EuroS&P), 2019, 464–479.

[10] M. Bellare, O. Goldreich, A. Mityagin, *The Power of Verification Queries in Message Authentication and Authenticated Encryption*, Cryptology ePrint Archive, Report 2004/309, 2004 `eprint.iacr.org/2004/309.pdf`.

[11] S. Chatterjee, A. Menezes, P. Sarkar, "Another look at tightness", Selected Areas in Cryptography: 18th International Workshop, SAC 2011, Toronto, ON, Canada, 2012, 293–319.

# Quantum and postquantum

# A simple quantum circuit for the attack which shows vulnerability of quantum cryptography with phase-time coding

Dmitry Kronberg

Steklov Mathematical Institute of Russian Academy of Sciences, Russia
dmitry.kronberg@gmail.com

**Abstract**

The key feature of quantum cryptography is the possibility of security proof against any actions of the eavesdropper. Nevertheless, security proof may be very complex and include errors, which may cause the overall vulnerability, i.e., partial or full knowledge of the final key by the eavesdropper. Here, we propose a quantum circuit for the attack on a two-parametric quantum key distribution protocol with phase-time encoding, which demonstrates its vulnerability. This attack requires just three additional qubits and is shown to be relatively simple. We discuss the errors in the security proof which made this attack possible.

**Keywords:** quantum cryptography, quantum eavesdropping, quantum computing

## 1 Introduction

The main theoretical element for quantum key distribution protocol is its security proof, i.e., the expression for the secret key rate as a function of the initial and observed parameters, and the proof of the fact that the key with the corresponding length after privacy amplification step is secure according to the security parameter, see, e.g., [1, 2]. The eavesdropper may perform any actions which do not contradict the laws of quantum mechanics, but for correct security proof the key still remains secure.

In [3], it was shown, that security proof for the quantum key distribution protocol with phase-time encoding [4, 5] contains errors. The two main errors were the following:

– Only a particular case of transformation performed by the eavesdropper was considered at the most general one, hence some effective attacks were not taken into account;

– The use of the GLLP approach [6], which was designed for BB84 protocol and cannot be applied to other protocols without proper reasoning.

As an example, in [3] an attack was proposed which reveals critical vulnerability of the protocol: for any non-zero level of losses, the eavesdropper may know the whole key, while the legitimate users think that the key is completely secure, i.e., the key is secure according to the secret key rate expression. The complexity of this attack was not studied in [3], because in original security proof [4, 5] no practical restrictions were imposed on the eavesdropper.

Here, we propose a quantum circuit for the attack from [3] in single-photon case. Below, we show that three additional qubits are enough to solve this task. We also need four Toffoli gates and two CNOT operations.

## 2 A brief protocol and attack description

The latest version of the protocol with phase-time coding [4, 5] utilizes three time slots $\{|1\rangle, |2\rangle, |3\rangle\}$, uses two bases (the "left" one L and the "right" one R), and four states:

$$\begin{aligned} |0_L\rangle = |L_+\rangle &= \frac{1}{\sqrt{2}}(|1\rangle + |2\rangle), \qquad |0_R\rangle = |R_+\rangle = \frac{1}{\sqrt{2}}(|2\rangle + |3\rangle), \\ |1_L\rangle = |L_-\rangle &= \frac{1}{\sqrt{2}}(|1\rangle - |2\rangle), \qquad |1_R\rangle = |R_-\rangle = \frac{1}{\sqrt{2}}(|2\rangle - |3\rangle). \end{aligned} \tag{1}$$

The eavesdropper's task is to obtain bit information (0 or 1) from every basis without introducing large disturbance, and the detailed description of the attack can be found in [3], where the performance of the attack is compared with secret key rate expression of the protocol.

Let us briefly describe the two main stages for this attack in single-photon case:

1. The eavesdropper discriminates between the "left" and "right" bases, using a postselective (i.e., having an inconclusive outcome "?") observable given by the following positive operator-valued measure (POVM):

$$M_L = \frac{1}{2}\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad M_R = \frac{1}{2}\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad M_? = \frac{1}{2}\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{2}$$

   This measurement provides the basis information ("L" of "R") with *a posteriori* correct decision probability $\frac{2}{3}$, which is significantly larger than the probability $\frac{1}{2}$ obtained by simple guessing. In case of correct outcome, this measurement does not affect the state, otherwise the state collapses to $|2\rangle$.

2. Using the basis information, the eavesdropper performs a partial copying of the bit information, which can be described as the transformation

$$|0\rangle \otimes |\psi_0\rangle \to |0\rangle \otimes |\psi_0\rangle,$$

$$|1\rangle \otimes |\psi_0\rangle \to |1\rangle \otimes |\psi_1\rangle,$$

where the pure states $|\psi_0\rangle$ and $|\psi_1\rangle$ in the eavesdropper's memory are generally not orthogonal. Using such a transformation whether to "left" or to "right" states, the eavesdropper gets the key information without introducing large disturbance to the states at the receiver side.

## 3 Quantum circuit

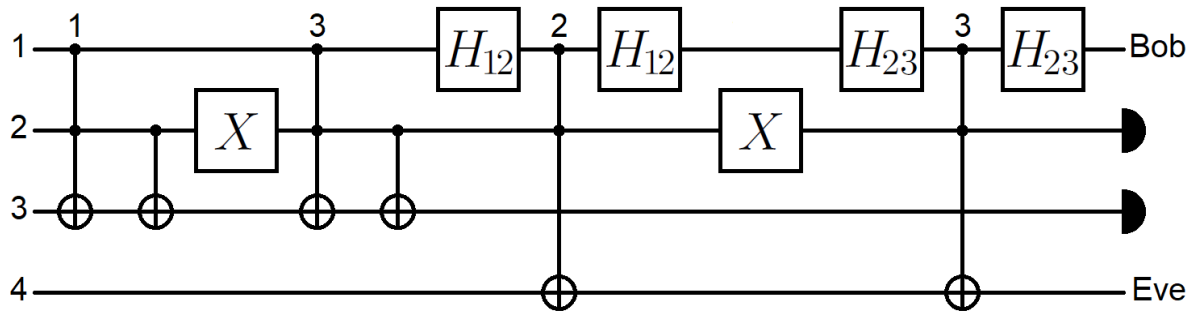



Figure 1: Quantum circuit for the proposed eavesdropping strategy

The quantum circuit for the proposed attack is shows at the Fig. 1. Let us describe the main elements of this scheme. It consists of the following single-photon gates: bit flip X, and Hadamard gate $H_{ij}$ for indices $i$ and $j$ of qutrit state, i.e.,

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad H_{12} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 & 0 \\ 1 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad H_{23} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & -1 \end{pmatrix}. \quad (3)$$

This circuit also contains two CNOT gates, and four Toffoli gates, for which the effect on qutrit state is specified with additional numbers. These are identity operations for every basic states except for the following:

$$Tof_1: \begin{array}{l} |1\rangle|1\rangle|0\rangle \to |1\rangle|1\rangle|1\rangle, \\ |1\rangle|1\rangle|1\rangle \to |1\rangle|1\rangle|0\rangle, \end{array} \quad (4)$$

$$Tof_2: \begin{array}{l} |2\rangle|1\rangle|0\rangle \to |2\rangle|1\rangle|1\rangle, \\ |2\rangle|1\rangle|1\rangle \to |2\rangle|1\rangle|0\rangle, \end{array} \quad (5)$$

$$Tof_3: \begin{array}{l} |3\rangle|1\rangle|0\rangle \to |3\rangle|1\rangle|1\rangle, \\ |3\rangle|1\rangle|1\rangle \to |3\rangle|1\rangle|0\rangle. \end{array} \tag{6}$$

The input elements of this scheme are the initial qutrit state (line 1), and auxiliary states $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, $|0\rangle$, and $\cos\varphi|0\rangle + \sin\varphi|1\rangle$ (lines 2, 3 and 4 correspondingly).

The key element of the attack is postselective measurement (2), which provides basis information, and may also yield inconclusive outcome "?". Let us show that the first five elements of the circuit correspond to this measurement. Let us first consider the action of the first two elements when the state of the "right" basis R is given on input:

$$\begin{aligned} |\pm R\rangle_1|+\rangle_2|0\rangle_3 \xrightarrow{\text{Tof}_1} |\pm R\rangle_1|+\rangle_2|0\rangle_3 \xrightarrow{\text{CNOT}} & |\pm R\rangle_1|0\rangle_2|0\rangle_3 : (\varnothing, \quad \frac{1}{2}) \\ & |\pm R\rangle_1|1\rangle_2|1\rangle_3 : (R, \quad \frac{1}{2}), \end{aligned}$$

Let us specify the notations. After the CNOT gate, the output at the lines 2 and 3 may be $|0\rangle_2|0\rangle_3$, which we consider as "nothing happened", and it occurs with probability $\frac{1}{2}$. These elements may also provide $|1\rangle_2|1\rangle_3$ as the output with the same probability, and this outcome we refer as "R basis". For the "left" basis states, the following happens:

$$\begin{aligned} |\pm L\rangle_1|+\rangle_2|0\rangle_3 \xrightarrow{\text{Tof}_1} |\pm L\rangle_1|0\rangle_2|0\rangle_3 \xrightarrow{\text{CNOT}} & |\pm L\rangle_1|0\rangle_2|0\rangle_3 : (\varnothing, \quad \frac{1}{2}), \\ |1\rangle_1|1\rangle_2|1\rangle_3 \xrightarrow{\text{CNOT}} & |1\rangle_1|1\rangle_2|0\rangle_3 : (?, \quad \frac{1}{4}), \\ |2\rangle_1|1\rangle_2|0\rangle_3 \xrightarrow{\text{CNOT}} & |2\rangle_1|1\rangle_2|1\rangle_3 : (R, \quad \frac{1}{4}). \end{aligned}$$

Here, $|0\rangle_2|0\rangle_3$ and $|1\rangle_2|1\rangle_3$ outcomes are also referred as "nothing happened" and "R basis" correspondingly, but the probability of the latter is $\frac{1}{4}$, and the third outcome, $|1\rangle_2|0\rangle_3$, is also possible with probability $\frac{1}{4}$, which is regarded as the inconclusive result.

After that, we flip the qubit at the second line, and when "nothing happened" after the first two operations, the next operations will be performed with the value $|1\rangle_2$ in the second line. Hence, the second line is responsible for the probabilistic application of the transformation.

After the five elements of the scheme, it is straightforward to see the

following:

$$\begin{aligned}
|\pm L\rangle_1|+\rangle_2|0\rangle_3 \rightarrow & |\pm L\rangle_1|1\rangle_2|1\rangle_3 : (L, \quad \frac{1}{2}) \\
& |1\rangle_1|0\rangle_2|0\rangle_3 : (?, \quad \frac{1}{4}), \\
& |2\rangle_1|0\rangle_2|1\rangle_3 : (R, \quad \frac{1}{4}), \\
|\pm R\rangle_1|+\rangle_2|0\rangle_3 \rightarrow & |\pm R\rangle_1|0\rangle_2|1\rangle_3 : (R, \quad \frac{1}{2}) \\
& |2\rangle_1|1\rangle_2|1\rangle_3 : (L, \quad \frac{1}{4}), \\
& |3\rangle_1|1\rangle_2|0\rangle_3 : (?, \quad \frac{1}{4})
\end{aligned} \tag{7}$$

Hence, these elements do implement the observable (2), as the corresponding probabilities and output states are exactly the same. Recall that this transformation is a sort of non-demolition basis measurement which provides basis information with *a posteriori* error rate $\frac{1}{3}$ (according to Bayes' theorem), and does not changes the states in case of correct outcome.

As one can see from (7), the line 2 corresponds to the basis outcome: $|1\rangle_2$ for the "left" basis, and $|0\rangle_2$ for the "right" basis. The line 3 indicates success or failure: $|0\rangle_3$ for inconclusive result, and $|1\rangle_3$ for success. Note that in case of failure, the basis information is irrelevant, as the bit values got lost, and the eavesdropper blocks the corresponding photons.

After obtaining basis information, the eavesdropper performs the corresponding partial information extraction. Hence, the measurement scheme depends on the line 2, which stores the basis information. This partial extraction of information needs three elements: two Hadamard gates and one Toffoli gate. Their action, when the observed basis is L (i.e., the state at line 2 is $|1\rangle_2$), reads

$$\begin{aligned}
&|+L\rangle_1|1\rangle_2|\varphi_0\rangle_4 \xrightarrow{H_{12}} |0\rangle_1|1\rangle_2|\varphi_0\rangle_4 \xrightarrow{\mathrm{Tof}_2} |0\rangle_1|1\rangle_2|\varphi_0\rangle_4 \xrightarrow{H_{12}} |+L\rangle_1|1\rangle_2|\varphi_0\rangle_4 \\
&|-L\rangle_1|1\rangle_2|\varphi_0\rangle_4 \xrightarrow{H_{12}} |1\rangle_1|1\rangle_2|\varphi_0\rangle_4 \xrightarrow{\mathrm{Tof}_2} |1\rangle_1|1\rangle_2|\varphi_1\rangle_4 \xrightarrow{H_{12}} |-L\rangle_1|1\rangle_2|\varphi_1\rangle_4,
\end{aligned} \tag{8}$$

where

$$\begin{aligned}
|\varphi_0\rangle &= \cos\varphi|0\rangle + \sin\varphi|1\rangle, \\
|\varphi_1\rangle &= \sin\varphi|0\rangle + \cos\varphi|1\rangle,
\end{aligned}$$

hence this operation makes the states at line 4 distinguishable, and these states store partial information about the bit value. The three last elements

to the same, when the observed basis value is R. Here, we use the parameter $\varphi = \frac{\pi}{4} - \gamma \in [0, \frac{\pi}{4}]$ for simplicity, where $\gamma$ is the original attack parameter, see [3]. $\varphi = 0$ corresponds to full knowledge, while $\varphi = \frac{\pi}{4}$ extracts no information.

After applying the whole quantum circuit, the eavesdropper can obtain the bit information from line 4. The eavesdropper makes a decision after measuring the output of the line 3, since this line indicates success or failure for basis discrimination. When this value is 0, the photon should be blocked. The single-photon attack needs blocking a quarter of the pulses.

As shown in [3], after setting a proper value of the attack parameter $\varphi$, the eavesdropper may obtain the whole information about the key, while the legitimate users would think that the key is completely secure, hence this attack reveals a critical vulnerability of the protocol.

## 4 Conclusion

We have proposed a quantum circuit for an attack which demonstrates the vulnerability of the QKD protocol with phase-time encoding. This scheme works for a single-photon case, and is relatively simple: it requires just three additional qubits, and besides single-photon operations it uses four Toffoli gates and two CNOT operations. We do not claim the optimality for this circuit, hence a simpler circuit with lower number of extra qubits and/or operations may in principle exist. Note that the corresponding attack may be also not optimal, since the optimality is not claimed in [3].

The main error of security proof given in [4, 5], as noted in [3], is that the unitary operation which was considered as the most general one, is actually not the most general. It does not include the case of the eavesdropper who introduces losses to the channel. As is it widely known in quantum mechanics, the class of operations which may be done with non-unit success probability, is significantly broader than the class of deterministic quantum operations. Therefore, the attack from [3] directly uses the possibility to introduce the losses to the channel in case of failure, and this possibility allows the eavesdropper to perform effective postselective discrimination between the bases.

We note that the more general protocol description [5] includes the case of lossy channel, but these losses are taken into account with GLLP approach, when the fraction of single-photon pulses which contribute the key is estimated by using decoy state method. But the GLLP approach was designed for BB84 protocol, where blocking some of single-photon pulses does not help the eavesdropper to design more effective attacks. It is not the case for the

protocol with phase-time coding, hence the use of the GLLP approach is incorrect and also causes vulnerability.

A possible countermeasure against the described attack could be the use of larger number of bases. Recall that the earlier version of the protocol with phase-time coding [7, 8] included four bases: two "left" and two "right", with configuration within each pair similar to BB84 (see Ref. [7]) or SARG04 (see Ref. [8]). Nevertheless, for this modification the aforementioned security proof issues still remain, and the complete security proof should use the correct dimension of the receiver space and should not use the GLLP approach without proper additional reasoning.

## References

[1] Portmann C., Renner R., "Security in quantum cryptography", *Reviews of Modern Physics*, **94**:2 (2022), 025008.

[2] Trushechkin A. S., "On the operational meaning and practical aspects of using the security parameter in quantum key distribution", *Quantum Electronics*, **50**:5 (2020), 426—439.

[3] Kronberg D. A., "Vulnerability of quantum cryptography with phase–time coding under attenuation conditions", *Teoreticheskaya i Matematicheskaya Fizika*, **214**:1 (2023), 140–152.

[4] Molotkov S. N., "Tight finite-key analysis for two-parametric quantum key distribution", *Laser Physics Letters*, **16**:3 (2019), 035203.

[5] Molotkov S. N., "Robustness of quantum cryptography systems with phase–time coding against active probing attacks", *Journal of Experimental and Theoretical Physics*, **131** (2020), 877–894.

[6] Gottesman D., Lo H. K., Lutkenhaus N., Preskill J., "Security of quantum key distribution with imperfect devices", International Symposium on Information Theory, 2004.

[7] Molotkov S. N., "Cryptographic robustness of a quantum cryptography system using phase-time coding", *Journal of Experimental and Theoretical Physics*, **106** (2008), 1–16.

[8] Kronberg D. A., Molotkov S. N., "Security of a two-parameter quantum cryptography system using time-shifted states against photon-number splitting attacks", *Journal of Experimental and Theoretical Physics*, **109** (2009), 557–584.

# The McEliece–type Cryptosystem based on $D$–codes

Yu. Kosolapov and E. Lelyuk

Southern Federal University, Russia
yvkosolapov@sfedu.ru, lelukevgeniy@mail.ru

**Abstract**

Tensor product codes are often used in data transmission systems to protect against errors. In this article we study the McEliece–type cryptosystem based on $D$–code construction, which is one generalization of tensor product codes. The object of research is the resistance of this cryptosystem to attacks on the key and ciphertext. We propose a combined cryptanalytic method, which in some cases, using structural analysis, significantly increases the success probability of an attack on ciphertext using information set decoding (ISD-attack). The developed analysis method is applied to $D$–codes based on binary Reed–Muller codes. For the parameters of these $D$–codes corresponding to weak keys, the success probability of the combined attack is estimated, and for $D$–codes corresponding to keys that are guaranteed to be resistant to this attack, the success probability of ISD-attack is estimated.

**Keywords:** McEliece–type cryptosystem, tensor product, $D$–codes, security analysis, Schur-Hadamard product

## 1 Introduction

The Classic McEliece code–based cryptosystem [1] is one of the contenders for the asymmetric encryption standard selected as part of the NIST PQC competition [2]. This cryptosystem is based on Goppa codes, which are a subclass of alternate codes. The main disadvantage of this cryptosystem is the key size. Attempts to use Reed–Solomon codes [3], Reed–Muller codes [4], algebraic-geometric codes [5], low–density parity–check codes [6] for reducing the key size have not been successful, since structural attacks on the corresponding cryptosystems were found (see [7]–[11]). Currently for some classes of Goppa codes, structural attacks on the corresponding cryptosystems have also been found [12, 13]. In addition, an effective structural attack has also been found for one class of subspace subcodes of Reed–Solomon codes, which are also alternate [14]. These results do not exclude the appearance in the future of effective structural attacks on the Classic McEliece cryptosystem

based on other classes of Goppa codes. Therefore, despite the available secure schemes, the problem of finding other efficiently decodable codes that provide high security of code–based cryptosystems is a relevant problem.

One way to obtain new codes is to use code constructions based on known codes (base codes). We note that the use of such code constructions as the combination of codes [15], the direct sum of codes, the transition from field extensions to basic fields [16] did not allow to increase the security [17, 18]. Nevertheless, code constructions are promising, since they make it possible to construct new efficiently decodable codes based on known codes. In general, new codes belong to a class that differs from the class of base codes, i.e. have a different structure (algebraic and/or combinatorial), so structural attacks on cryptosystems based on base codes are not directly applicable to cryptosystems based on new codes. Note that many structural attacks are based on the use of the Schur–Hadamard product [9, 17, 18], so it is important to estimate the security of cryptosystems based on new constructions to such attacks.

An important example of a code construction is the tensor product codes, as it is widely used in telecommunication for error correction (see patents [19]–[21]). In this paper, we study a McEliece–type cryptosystem based on $D$–codes, which are one of the generalizations of the tensor product codes. Namely, we consider $D$–codes based on families of Reed–Muller codes. Based on new and earlier results obtained by the authors regarding the properties of $D$–codes [22], the requirements for $D$–codes (including the tensor product codes) are determined, under which the security of the cryptosystem is guaranteed to structural attacks based on the Schur-Hadamard product as well as to the information set decoding attack. Parameters of $D$–codes based on binary Reed–Muller codes, which correspond to the strong keys of the cryptosystem, are given.

## 2 Preliminaries

Let $\mathbb{F}_q^n$ be a vector space over the Galois field $\mathbb{F}_q$. The zero vector of this space is denoted by $\mathbf{0}$. For a vector $\mathbf{x}(\in \mathbb{F}_q^n)$ the set of its nonzero coordinates is called the support of the vector $\mathbf{x}$ and is denoted by $\mathrm{supp}(\mathbf{x})$. The weight $\mathrm{wt}(\mathbf{x})$ of the vector $\mathbf{x}$ is defined as $|\mathrm{supp}(\mathbf{x})|$. Here and below, the symbol $|A|$ denotes the cardinality of the set $A$. For a $(k \times n)$–matrix $M$ and a set $\tau \subseteq \{1, ..., n\}$ the projection of the matrix $M$ onto the set $\tau$ is called $(k \times |\tau|)$–matrix $\tau(M)$ composed of columns of matrix $M$ with indices from $\tau$ in natural order. For a set $U \subset \mathbb{F}_q^n$, denote by $\mathcal{L}(U)$ its linear span. By the

tensor product $A \otimes B$ of the matrices $A = (a_{i,j})$ and $B$ of size $k_1 \times n_1$ and $k_2 \times n_2$, respectively, we mean $(k_1 k_2 \times n_1 n_2)$–matrix of the following form:

$$\begin{pmatrix} a_{1,1}B & \cdots & a_{1,n_1}B \\ \vdots & \ddots & \vdots \\ a_{k_1,1}B & \cdots & a_{k_1,n_1}B \end{pmatrix}.$$

A linear $[n, k, d]_q$–code $C$ is a subspace of the space $\mathbb{F}_q^n$ of dimension $k$, with the minimum code distance $d = \min\{\mathrm{wt}(\mathbf{c}) \mid \mathbf{c} \in C \setminus \{\mathbf{0}\}\}$. We denote the generator matrix of the code $C$ by $G_C$, i.e. $C = \mathcal{L}(G_C)$. The code dual to $C$, as in [23, 24, 22], will be denoted $\overline{C}$ for convenience. An information set of a code $C$ is a set $\tau \subset \{1, ..., n\}$ of cardinality $k$ such that $\mathrm{rank}(\tau(G_C)) = k$. Tensor product $C_1 \otimes C_2$ of two $[n_i, k_i, d_i]_q$–codes $C_i (\subset \mathbb{F}_q^{n_i})$, where $i \in \{1, 2\}$, can be defined as $\mathcal{L}(G_{C_1} \otimes G_{C_2})$. It is known that $C_1 \otimes C_2$ is an $[n_1 n_2, k_1 k_2, d_1 d_2]_q$–code ([25], n. 6.2.3.). The power $s$ (in the sense of the Schur–Hadamard product) of a code $C \subset \mathbb{F}_q^n$ is a code $C^s$ defined as follows [26]:

$$C^s = \mathcal{L}\big(\{\mathbf{c}_1 \star ... \star \mathbf{c}_s | \forall \mathbf{c}_1, ..., \mathbf{c}_s \in C\}\big),$$

where $\mathbf{x} \star \mathbf{y} = (x_1 y_1, ..., x_n y_n)$. It is known (see [27]) that

$$(C_1 \otimes C_2)^s = C_1^s \otimes C_2^s. \tag{1}$$

Recall that a $[n, k]_q$–code $C$ is called *decomposable* [28] if for any of its generator matrix $G_C$ there exists a nondegenerate $(k \times k)$–matrix $M$ and permutation $(n \times n)$–matrix $Q$, such that $MG_CQ = \mathrm{diag}(A_1, ..., A_r), \mathrm{rank}(A_i) \geqslant 1, r \geqslant 2$.

One generalization of the tensor product construction is the $D$–code construction, whose properties are considered in [23, 24, 22]. Let's briefly define these codes. Let $C_1(k_i) \subseteq \mathbb{F}_q^{n_1}$, $k_i \geqslant 0$, $C_2(l_i) \subseteq \mathbb{F}_q^{n_2}$, $l_i \geqslant 0$, $i = 1, ..., s$; codes $C_1(k_i)$, $C_2(l_i)$ will be called as base codes. Let's define the code

$$\overline{C(D)} = \sum_{i=1}^{s} \overline{C_1(k_i)} \otimes \overline{C_2(l_i)}. \tag{2}$$

If for $s \geqslant 2$ and $i < s$ in the introduced notation we have $\overline{C_1(k_i)} \subset \overline{C_1(k_{i+1})}$, $\overline{C_2(l_{i+1})} \subset \overline{C_2(l_i)}$, and the conditions from [24] are also satisfied, then $\overline{C(D)}$ is a $D$–code and it can be decoded. Note that, according to [23], $D$ is a subset of the set of pairs of non-negative integers and defines the set of pairs $(k_i, l_i)$ from (2). Some properties of powers of $D$–codes and their decomposability are studied in [22].

One of the important classes of codes is the class of binary $[n, k, d]_2$ Reed–Muller codes $\mathrm{RM}(r, m)$ [29], where $n = 2^m$, $k = \sum_{i=0}^{r} \binom{r}{i}$, $d = 2^{m-r}$ and $\mathrm{RM}(r, m) = \mathrm{RM}(m, m) = \mathbb{F}_2^n$ for $r > m$. It is known that Reed–Muller codes for $r < m$ are indecomposable [17], and it's shown in [9] that

$$\mathrm{RM}(r, m)^s = \mathrm{RM}(rs, m). \tag{3}$$

The linear $[n, k, d]_q$–code $C$ is considered as the basis of the asymmetric code–based cryptosystem [2]. In the system proposed by R. McEliece in [30], the secret key is the triple $(C, S, P)$, where $S$ is a random nondegenerate $(k \times k)$–matrix, $P$ is a random permutation $(n \times n)$–matrix. The public key of this cryptosystem is the pair $(\tilde{G}, t)$, where $t = \lfloor (d-1)/2 \rfloor$,

$$\tilde{G} = SG_C P. \tag{4}$$

The information vector $\mathbf{m}(\in \mathbb{F}_q^k)$ is encrypted according to the rule $\mathbf{c} = \mathbf{m}\tilde{G} + \mathbf{e}$, where the vector $\mathbf{e}$ is usually chosen randomly with equal probability among vectors of weight $t$. The secret key is used for decryption: $\mathbf{m} = S^{-1}\tau(G_C)^{-1}\tau(\mathsf{Dec}_C(\mathbf{c}P^{-1}))$, where $\mathsf{Dec}_C : \mathbb{F}_q^n \to C$ is an efficient decoder for the code $C$, and $\tau$ is any information set. We denote the McEliece-type cryptosystem on a code $C$ by $\mathrm{McE}(C)$.

The $D$–code can be efficiently decoded, so a McEliece system can be built on its basis. Since a $D$–code generally belongs to a class that differs from the classes of base codes, structural attacks on McEliece cryptosystems based on base codes are not directly applicable to cryptosystems based on $D$–codes. In particular, if the base codes are generalized Reed–Solomon codes, then the Sidelnikov–Shestakov attack is not applicable. When the base codes are binary Reed–Muller codes, then in the general case the Minder-Shokrollahi and Chizhov-Borodin attacks are not applicable. Therefore, the problem of analyzing the structural security of the $\mathrm{McE}(\overline{C(D)})$ cryptosystem is relevant. Often, when analyzing structural security, the Schur–Hadamard product is used, which in some cases allows reducing the cryptanalysis of a system based on the code of an unexplored structure to the analysis of a well-known code cryptosystem [9, 17, 18]. In [22] using the Schur–Hadamard product, the results of the structural strength of $\mathrm{McE}(\overline{C(D)})$ are obtained.

In the next sections, based on the results of [22], we construct a cryptanalytic framework for McEliece-type system on $D$-codes. To simplify the analysis, the error vector in encryption rule is generated as follows. Each bit of the vector $\mathbf{e}$ takes the value 1 with probability $t/n$ and 0 with probability $1 - t/n$. Within the framework of such an error generation model, it is

possible to check the weight of the vector $\mathbf{e}$ during encryption and, if necessary, repeat the generation process. Then we can assume that $\mathrm{wt}(\mathbf{e}) = t$, i.e. the weight of the error vector is within the correcting capability of the code. Note that this method of error generation is used in [31] also to simplify the analysis.

## 3 Analysis of the security of McEliece–type cryptosystems based on the subcode of the direct sum of codes

Let $n_1, n_2 \in \mathbb{N}$, $n = n_1 n_2$, $C_2$ be a $[n_2, k_2, d_2]$–code, $C$ be a $[n, k, d]$–code $C$, such that

$$C \subset \mathbb{F}_q^{n_1} \otimes C_2, \quad \mathrm{rank}(\tau_i(G_C)) = k_2, \tag{5}$$

$$\tau_i = \{(i-1)n_2 + 1, ..., in_2\}, \quad i = 1, ..., n_1. \tag{6}$$

The submatrix $\tau_i(G_C)$ is called the block of the matrix $G_C$ with number $i$. From (6) we have that $G_C$ can be represented as follows

$$G_C = M\mathrm{diag}(G_{C_2}, ..., G_{C_2}) = (M_1 G_{C_2} \parallel ... \parallel M_{n_1} G_{C_2}),$$

where $M = (M_1 \parallel ... \parallel M_{n_1})$ is the $(k \times n_1 \dim(C_2))$–matrix and $\mathrm{rank}(M_i) = k_2$ for $i = 1, ..., n_1$.

Let's consider $\mathrm{McE}(C)$. If for natural $s \geqslant 2$

$$C^s = \mathbb{F}_q^{n_1} \otimes C_2^s \neq \mathbb{F}_q^{n_1 n_2} \tag{7}$$

and $C_2^s$ is indecomposable, then one can apply the attack algorithm to $\mathrm{McE}(C)$ from [22] (see the proof of Theorem 6), that finds such a permutation matrix $\Pi$, that

$$\tilde{G}\Pi = (\hat{S}_1 G_{C_2} \parallel ... \parallel \hat{S}_{n_1} G_{C_2}) = (\hat{G}_1 \parallel ... \parallel \hat{G}_{n_1}) = \hat{G}, \tag{8}$$

where $\hat{S}_i$ is a $(k \times k_2)$–matrix of rank $k_2$. Denote this algorithm AttackDKey.

**Remark 1.** *If $C^{s-1}$ is an indecomposable code and $C^s = \mathbb{F}_q^{n_1 n_2}$, then the* AttackDKey *algorithm is not applicable to the cryptosystem* $\mathrm{McE}(C)$.

Let $\mathbf{z} = \mathbf{m}\tilde{G} + \mathbf{e}$ be the received ciphertext, where $\mathbf{m}$ is the information message and $\mathbf{e}$ is error vector. Then

$$\begin{aligned} \mathbf{z}\Pi &= \mathbf{m}\hat{G} + \mathbf{e}\Pi \\ &= [c_1 \parallel c_2 \parallel ... \parallel c_{n_1}] + [\hat{e}_1 \parallel \hat{e}_2 \parallel ... \parallel \hat{e}_{n_1}] \\ &= [\hat{z}_1 \parallel \hat{z}_2 \parallel ... \parallel \hat{z}_{n_1}] = \hat{\mathbf{z}},\ c_i \in C_2. \end{aligned}$$

Denote by $\hat{\mathbf{c}} = [\hat{c}_1 \parallel ... \parallel \hat{c}_{n_1}]$ the vector obtained from the vector $\hat{\mathbf{z}}$ by applying the decoder of code $C_2$ to each block $\hat{z}_i$. For convenience, the decoder that takes $\hat{\mathbf{z}}$ as input and returns $\hat{\mathbf{c}}$ is denoted by DecoderSum. Then the AttackDCipher attack can be applied to the ciphertext $\mathbf{z}$ (see the algorithm 1), which consists in applying the AttackDKey algorithm to find the matrix $\Pi$ (step 1), decoding the vector $\mathbf{z}\Pi$ using the DecoderSum algorithm (step 3) and then applying the information set decoding algorithm to the resulting vector (steps 6–15). If the attack is successful, the AttackDCipher algorithm will return the value $\hat{\mathbf{m}} = \mathbf{m}$, otherwise it will return the value $\hat{\mathbf{m}} = \perp$.

**Algorithm 1** AttackDCipher

**Require:** $\tilde{G}$, $\mathbf{z}$ , $p$, $I_{max}$
**Ensure:** $\hat{\mathbf{m}}$
$\Pi :=$ AttackDKey$(\tilde{G})$.
$\hat{G} := \tilde{G}\Pi$, $\hat{\mathbf{z}} := \mathbf{z}\Pi$.
$\hat{\mathbf{c}} :=$ DecoderSum$(\hat{\mathbf{z}})$.
Choose from $\{1, ..., n_1\}$ the minimal value $K_p$ such that the matrix composed of $K_p$ randomly selected submatrices $\hat{G}_i$ of the matrix $\hat{G}$, will have rank $k$ with probability greater than $p$.
$i := 0$
**repeat**
  Randomly choose $K_p$ blocks $\hat{c}_i$, denote the set of numbers of coordinates in the chosen blocks by $T$.
  **if** $\mathrm{rank}(T(\hat{G})) = k$, **then**
    choose $\tau \subset T$ such that $|\tau| = k$, $\mathrm{rank}(\tau(\hat{G})) = k$,
  **else**
    Go to step 7.
  **end if**
  $\hat{\mathbf{m}} := \tau(\hat{\mathbf{c}})\tau(\hat{G})^{-1}$.
  $i := i + 1$
**until** $\hat{\mathbf{z}} - \hat{\mathbf{m}}\hat{G} > t$ и $i < I_{max}$.
**if** $i = I_{max}$ и $\hat{\mathbf{z}} - \hat{\mathbf{m}}\hat{G} > t$, **then**
  $\hat{\mathbf{m}} := \perp$.
**end if**
**return** $\hat{\mathbf{m}}$.

**Remark 2.** *The parameter $p$ of the* AttackDCipher *algorithm affects the choice of $K_p$. The more $p$, the more $K_p$, i.e. if we what to find a matrix of a given rank at the step 8 with higher probability then we should increase the number of blocks to choose. On the other hand, the larger the value of $K_p$, the less likely it is that there will be no bad blocks among the selected blocks. Since at the step 4 of the algorithm for a given $p$ it is necessary to carry out at least $1/p$ experiments to check the next chosen value $K_p$, it makes sense to choose the value of the parameter $p$ based on the computational capabilities*

*of the attacker.*

**Remark 3.** *The value $K_p$ that found at the step 4 may turn out to be too large to find $K_p$ error-free blocks $\hat{c}_i$. It means that the check $\hat{\mathbf{z}} - \hat{\mathbf{m}}\hat{G} > t$ will always be true. To prevent looping of the algorithm, the $I_{max}$ parameter is introduced. The value of $I_{max}$ must be chosen in a reasonable way, that is, so that it does not affect the probability of success of the attack. Further, when assessing the probability of success of the attack, the recommendations for choosing the value of the $I_{max}$ parameter are refined.*

Let's analysis the probability of the attack AttackDCipher success. The analysis is based on the estimate of the number $N_g$ of error-free blocks $\hat{c}_i$ in the vector obtained after applying the decoder DecoderSum. If $wt(\hat{e}_i) \leqslant t_2 = \lfloor (d_2-1)/2 \rfloor$ then vector $\hat{z}_i$ will be decoded correctly, in this case block $\hat{z}_i$ will be called «good», otherwise, i.e. for $wt(\hat{e}_i) > t_2$, the block $\hat{z}_i$ will be called «bad». Let's consider variants of distribution of $t = \lfloor (d-1)/2 \rfloor$ errors over $n_1$ blocks. The largest number of «bad» blocks is achieved when each such block has exactly $t_2+1$ errors. With other variant of the distribution of errors between coordinates, the number of «bad» blocks can only be less. Then the maximum number of bad blocks is equal to $\lfloor \lfloor (d-1)/2 \rfloor / \lfloor (d_2-1)/2+1 \rfloor \rfloor$, and the minimum number of good ones, respectively

$$N_g^{min} = n_1 - \lfloor (d-1)/(d_2+1) \rfloor. \tag{9}$$

Let's estimate the average number of good blocks. Within the framework of the considered error generation model, each of $t$ errors falls into the $i$–th block with a probability $1/n_1$, $i = 1, ..., n_1$, i.e. the probability of an error in the $i$–th block follows the Bernoulli distribution with the parameter $t/n_1$. Then the average number of errors in a block is $t/n_1$, and if $t/n_1 <= t_2 + 1$, then if the weight of a bad block is more than $t_2 + 1$, the probability of such a block is reduced. Therefore, the average number of bad blocks will also decrease. If

$$t/n_1 > t_2 + 1, \tag{10}$$

then all blocks are bad on average and the attack AttackDCipher is not applicable. Then, further in the analysis, we will assume that the weight of a bad block is equal to $t_2+1$. Let $A_i$ be the property that the $i$–th block of the vector is bad, $i = 1, ..., n_1$. Let $C_r(n_1, n_2, t_1, t_2, t)$ be the number of error vectors that have exactly $r$ bad blocks. Then, according to the inclusion–exclusion formula [32] (see Chap. 2, formula (1.9)) we have

$$C_r(n_1, n_2, t_1, t_2, t) = \sum_{k=r}^{n_1} (-1)^{k-r} \binom{k}{r} S_k,$$

where $S_k = \sum_{1 \leqslant i_1 < \ldots < i_k \leqslant n_1} M(A_{i_1}, ..., A_{i_k}) = \binom{n_1}{k}\left(\binom{n_2}{t_2+1}\right)^k \binom{(n_1-k)n_2}{t-(t_2+1)k}$, a $M(A_{i_1}, ..., A_{i_k})$ — number of vectors with fixed properties $A_{i_1}, ..., A_{i_k}$, i.e. having bad blocks in numbers $i_1, i_2, ..., i_k$. Then the probability $Q_r$ that there will be exactly $r$ bad blocks in the error vector is equal to

$$Q_r = \frac{C_r(n_1, n_2, t_1, t_2, t)}{\binom{n}{t}}, \tag{11}$$

and the average number of good blocks $N_g^{avg}$ is calculated by the formula

$$N_g^{avg} = \lfloor n_1 - \sum_{r=0}^{n_1} r \cdot Q_r \rfloor = \lfloor n_1 - \sum_{r=1}^{n_1 - N_g^{min}} r \cdot Q_r \rfloor. \tag{12}$$

According to remark 2, the probability $P_{attack}(p)$ of the success of the attack depends on the parameter $p$ of the AttackDCipher algorithm, since it actually affects the value of $K_p$, which in turn affects the probability fulfillment of conditions at the steps 8,15. Then the probability $P_{attack}(p)$ is estimated by the product $P_{attack}(p) \geqslant P_1(p) \cdot P_2(p)$, where $P_1(p) = \binom{N_g}{K_p} / \binom{n_1}{K_p}$ — the probability of choosing $K_p$ good blocks from $n_1$ blocks at the step 7 of the algorithm 1, and $P_2(p)$ — is the probability that $K_p n_2$ selected columns form a matrix of rank $k$. Further, for convenience, the dependence on $p$ will be omitted in the notation of probabilities.

Strictly speaking, the probability $P_{attack}$ also depends on the parameter $I_{max}$ of the AttackDCipher algorithm. In particular, choosing a sufficiently small $I_{max}$ leads to an early termination of the attack with an error. To reduce this dependence, it makes sense to choose $I_{max} \approx 1/(P_1 \cdot P_2)$.

Since the minimum number of good blocks is $N_g^{min}$ and the average is $N_g^{avg}$, then $P_1^{min} = \binom{N_g^{min}}{K_p} / \binom{n_1}{K_p}$, $P_1^{avg} = \binom{N_g^{avg}}{K_p} / \binom{n_1}{K_p}$. So we have

$$P_{attack}^{min} \geqslant P_1^{min} \cdot P_2, \quad P_{attack}^{avg} \geqslant P_1^{avg} \cdot P_2. \tag{13}$$

# 4 Security of McE($C$) based on $D$–code $C$

In this section we analyze the security of cryptosystem McE($C$) that based on $D$–code $C$ using the results of the previous section. First, we consider tensor product code as a special case of $D$–code, and then we consider the general case when $D$–code has the form (2).

## 4.1 Case $C = C_1 \otimes C_2$

Let's consider the cryptosystem McE($C$) based on $[n, k, d]$–code $C = C_1 \otimes C_2$, where $C_i$ — $[n_i, k_i, d_i]$–code, $n = n_1 n_2$, $k = k_1 k_2$, $d = d_1 d_2$, $t =$

$\lfloor (d_1d_2-1)/2 \rfloor$, $t_i = \lfloor (d_i-1)/2 \rfloor$ for $i = 1, 2$. For code $C$ inequality (10) is not true, because $t \leqslant (d_1d_2-1)/2 \leqslant (n_1d_1-1)/2$ и $t_2+1 = \lfloor (d_2-1)/2 \rfloor + 1 = \lfloor (d_2+1)/2 \rfloor$, whence $t/n_1 \leqslant (d_2 - 1/n_1)/2 \leqslant d_2/2 \leqslant \lfloor (d_2+1)/2 \rfloor = t_2+1$. Also note that the conditions from (6) are satisfied for the code $C$. Therefore, if $C^s = \mathbb{F}_q^{n_1} \otimes C_2^s$ and $C_2^s$ is an indecomposable code, then the cryptosystem McE($C$) with a public matrix of the form (4) can be attacked by method from the section 3. In particular, the permutation matrix $\Pi$ can be found at the step 1 of the algorithm 1 and the matrix $\hat{G}$ of the form (8) can be calculated at the step 2 of the same algorithm. The minimum value of $K_p$ for $p = 0$ that required at the step 4 can be found analytically for the $C = C_1 \otimes C_2$. Namely, let $\omega \subset \{1, ..., n_1\}$ be the set of block numbers $\hat{G}_i$. Denote by $T$ the set of numbers of the $|\omega| n_2$ columns of the matrix $\hat{G}$ corresponding to $\omega$. It is known from [22] that the permutation matrix $\Pi$ is such that the matrix $Q = P\Pi$ permutes the blocks $\tau_i(G_C)$ and the columns inside these blocks in the matrix $G_C$. That is, $Q$ can be represented as $Q = (W \otimes I_{n_2})\mathrm{diag}(V_1, ..., V_{n_1})$, where $W$ is the permutation $(n_1 \times n_1)$–matrix, and $V_i$ are permutation $(n_2 \times n_2)$–matrices. Then

$$\begin{aligned}
\mathrm{rank}(T(\hat{G})) &= \mathrm{rank}(T(S(G_{C_1} \otimes G_{C_2})Q)) = \mathrm{rank}(T((G_{C_1} \otimes G_{C_2})Q)) \\
&= \mathrm{rank}(T((G_{C_1} \otimes G_{C_2})(W \otimes I_{n_2})\mathrm{diag}(V_1, ..., V_{n_1}))) \\
&= \mathrm{rank}(T((G_{C_1}W) \otimes G_{C_2})\mathrm{diag}(V_1, ..., V_{n_1})) \\
&= \mathrm{rank}(\omega(G_{C_1}W))\mathrm{rank}(G_{C_2}) = \mathrm{rank}(\omega(G_{C_1}W))k_2.
\end{aligned}$$

Since in the algorithm 1 at the step 7 blocks are chosen randomly, the probability of the event $\mathrm{rank}(\omega(G_{C_1}W)) = k_1$ is equal to the probability of the event $\mathrm{rank}(\tau(G_{C_1})) = k_1$ for $|\tau| = |\omega|$. Therefore, he probability $P_2$ that $\mathrm{rank}(T(\hat{G})) = k_1k_2$ is equal to the probability of the event $\mathrm{rank}(\tau(G_{C_1})) = k_1$ for a randomly chosen $\tau$. With $|\tau| < k_1$ the probability of this event is equal to zero, and already at $|\tau| = k_1$ for most codes this probability is non-negligible [33]. Therefore, by setting the value of parameter $p$ equal to zero in the algorithm 1, $K_p = k_1$ will be chosen at the step 4.

Lets' consider the case when $C_i$ is the $[n_i, k_i, d_i]_2$ Reed–Muller code $\mathrm{RM}(r_i, m_i)$, $i = 1, 2$. For the code $C = C_1 \otimes C_2$ in [34] an efficient majority–logical decoder is constructed and it is proposed to use $C$ in the McEliece cryptosystem McE($C$). Security to attacks on the key and ciphertext was studied in [34], [35]. Let us show how the attack constructed above makes it possible to refine the security of McE($C$) to attacks on the ciphertext.

According to (1), (3), for $r_1 = \lceil m_1/2 \rceil, ..., m_1$, $r_2 = 0, ..., \lceil m_2/2 \rceil - 1$ we have $C^2 = \mathbb{F}_2^{n_1} \otimes C_2^2$. Also $C_2^2$ is indecomposable, so McE($C$) can be attacked with AttackDCipher. In the table 1 for some cases with $m_1, m_2 \in \{7, 8\}$ the

estimate of the success probability of this attack is presented. To do this, the values of $P_{attack}^{min}$ and $P_{attack}^{avg}$ are calculated according to (13), while $P_2$ is estimated experimentally by calculating the rank of 10000 pseudo-randomly selected $(k_1 \times K_p)$–submatrices of the generator matrix of the code $C_1$ and calculation of the fraction of matrices of rank $k_1$. The table 1 also contains the calculation of the probability $P_{ISD}$ of the success of an attack on a ciphertext by information set decoding, which is calculated by the formula $P_{ISD} = \binom{n-t}{k} / \binom{n}{k}$. For each code $C = C_1 \otimes C_2$, $P_{attack}^{min}$ and $P_{attack}^{avg}$ are evaluated for different values of $p$ in order to analyze the behaviour of these probabilities.

Table 1: AttackDCipher attack success probability for the tensor product of Reed–Muller codes

| $\mathrm{RM}(r_1, m_1) \otimes \mathrm{RM}(r_2, m_2)$ | $p$ | $K_p$ | $P_{ISD}$ | $P_{attack}^{min}$ | $P_{attack}^{avg}$ |
|---|---|---|---|---|---|
| $\mathrm{RM}(4,7) \otimes \mathrm{RM}(3,7)$ | 0.1 | 99 | 2.379E-14 | 2.883E-06 | 3.956E-02 |
| $\mathrm{RM}(4,7) \otimes \mathrm{RM}(3,7)$ | 0.3 | 100 | 2.379E-14 | 4.904E-06 | 8.564E-02 |
| $\mathrm{RM}(4,7) \otimes \mathrm{RM}(3,7)$ | 0.5 | 101 | 2.379E-14 | 5.427E-06 | 1.219E-01 |
| $\mathrm{RM}(4,7) \otimes \mathrm{RM}(3,7)$ | 0.7 | 102 | 2.379E-14 | 5.017E-06 | 1.464E-01 |
| $\mathrm{RM}(5,7) \otimes \mathrm{RM}(3,7)$ | 0.3 | 120 | 1.577E-09 | 5.945E-05 | 2.265E-02 |
| $\mathrm{RM}(5,7) \otimes \mathrm{RM}(3,7)$ | 0.6 | 121 | 1.577E-09 | 7.046E-05 | 3.758E-02 |
| $\mathrm{RM}(5,7) \otimes \mathrm{RM}(3,7)$ | 0.8 | 122 | 1.577E-09 | 5.109E-05 | 4.088E-02 |
| $\mathrm{RM}(6,7) \otimes \mathrm{RM}(3,7)$ | 0.9 | 127 | 1.716E-05 | 7.812E-03 | 7.812E-03 |
| $\mathrm{RM}(4,8) \otimes \mathrm{RM}(3,8)$ | 0.2 | 163 | 4.868E-30 | 2.603E-08 | 8.101E-02 |
| $\mathrm{RM}(4,8) \otimes \mathrm{RM}(3,8)$ | 0.4 | 164 | 4.868E-30 | 4.690E-08 | 1.721E-01 |
| $\mathrm{RM}(4,8) \otimes \mathrm{RM}(3,8)$ | 0.6 | 165 | 4.868E-30 | 5.445E-08 | 2.362E-01 |
| $\mathrm{RM}(4,8) \otimes \mathrm{RM}(3,8)$ | 0.7 | 166 | 4.868E-30 | 5.304E-08 | 2.725E-01 |
| $\mathrm{RM}(5,8) \otimes \mathrm{RM}(3,8)$ | 0.1 | 219 | 1.931E-21 | 1.488E-07 | 2.749E-02 |
| $\mathrm{RM}(5,8) \otimes \mathrm{RM}(3,8)$ | 0.4 | 220 | 1.931E-21 | 2.725E-07 | 6.043E-02 |
| $\mathrm{RM}(5,8) \otimes \mathrm{RM}(3,8)$ | 0.6 | 221 | 1.931E-21 | 3.090E-07 | 8.269E-02 |
| $\mathrm{RM}(5,8) \otimes \mathrm{RM}(3,8)$ | 0.7 | 222 | 1.931E-21 | 2.993E-07 | 9.726E-02 |
| $\mathrm{RM}(6,8) \otimes \mathrm{RM}(3,8)$ | 0.3 | 247 | 9.941E-13 | 1.019E-05 | 1.179E-02 |
| $\mathrm{RM}(6,8) \otimes \mathrm{RM}(3,8)$ | 0.6 | 248 | 9.941E-13 | 1.303E-05 | 2.010E-02 |
| $\mathrm{RM}(6,8) \otimes \mathrm{RM}(3,8)$ | 0.8 | 249 | 9.941E-13 | 1.063E-05 | 2.296E-02 |
| $\mathrm{RM}(7,8) \otimes \mathrm{RM}(3,8)$ | 0.9 | 255 | 5.694E-07 | 3.906E-03 | 3.906E-03 |
| $\mathrm{RM}(4,8) \otimes \mathrm{RM}(3,7)$ | 0.2 | 163 | 4.412E-22 | 2.575E-08 | 8.014E-02 |
| $\mathrm{RM}(4,8) \otimes \mathrm{RM}(3,7)$ | 0.4 | 164 | 4.412E-22 | 4.611E-08 | 1.692E-01 |
| $\mathrm{RM}(4,8) \otimes \mathrm{RM}(3,7)$ | 0.6 | 165 | 4.412E-22 | 5.396E-08 | 2.341E-01 |
| $\mathrm{RM}(4,8) \otimes \mathrm{RM}(3,7)$ | 0.7 | 166 | 4.412E-22 | 5.394E-08 | 2.771E-01 |
| $\mathrm{RM}(4,8) \otimes \mathrm{RM}(2,8)$ | 0.2 | 163 | 2.788E-22 | 2.551E-08 | 7.938E-02 |
| $\mathrm{RM}(4,8) \otimes \mathrm{RM}(2,8)$ | 0.4 | 164 | 2.788E-22 | 4.645E-08 | 1.705E-01 |
| $\mathrm{RM}(4,8) \otimes \mathrm{RM}(2,8)$ | 0.6 | 165 | 2.788E-22 | 5.392E-08 | 2.339E-01 |
| $\mathrm{RM}(4,8) \otimes \mathrm{RM}(2,8)$ | 0.7 | 166 | 2.788E-22 | 5.320E-08 | 2.733E-01 |

The table 2 contains for codes from the table 1 the calculation of $N_g^{min}$

and $N_g^{avg}$ values using the formulas (9) and (12) respectively, calculation the probability $Q_{N_g^{avg}}$ of appearing $N_g^{avg}$ good blocks, as well as the probability $Q_{N_g^{min}}$ of a situation where the number of good blocks is minimal, according to the formula (11).

Table 2: Number of good blocks for the tensor product of Reed–Muller codes

| $\mathrm{RM}(r_1, m_1) \otimes \mathrm{RM}(r_2, m_2)$ | $n_1$ | $N_g^{min}$ | $N_g^{avg}$ | $Q_{N_g^{min}}$ | $Q_{N_g^{avg}}$ |
|---|---|---|---|---|---|
| $\mathrm{RM}(4,7) \otimes \mathrm{RM}(3,7)$ | 128 | 121 | 127 | 3.387E-57 | 3.676E-06 |
| $\mathrm{RM}(5,7) \otimes \mathrm{RM}(3,7)$ | 128 | 125 | 127 | 1.012E-29 | 9.481E-09 |
| $\mathrm{RM}(6,7) \otimes \mathrm{RM}(3,7)$ | 128 | 127 | 127 | 8.701E-12 | 8.701E-12 |
| $\mathrm{RM}(4,8) \otimes \mathrm{RM}(3,8)$ | 256 | 241 | 255 | 1.415E-265 | 1.833E-12 |
| $\mathrm{RM}(5,8) \otimes \mathrm{RM}(3,8)$ | 256 | 249 | 255 | 8.543E-151 | 2.538E-17 |
| $\mathrm{RM}(6,8) \otimes \mathrm{RM}(3,8)$ | 256 | 253 | 255 | 2.381E-75 | 1.440E-22 |
| $\mathrm{RM}(7,8) \otimes \mathrm{RM}(3,8)$ | 256 | 255 | 255 | 1.329E-28 | 1.329E-28 |
| $\mathrm{RM}(4,8) \otimes \mathrm{RM}(3,7)$ | 256 | 241 | 255 | 9.978E-127 | 9.616E-06 |
| $\mathrm{RM}(4,8) \otimes \mathrm{RM}(2,8)$ | 256 | 241 | 255 | 2.294E-547 | 3.743E-26 |

According to the table 1, for the considered codes, the probability of decryption using the constructed AttackDCipher attack is much greater than the probability of decryption by information set decoding, even in the worst case of the errors distribution for the attacker. Note that earlier in [35] (see Table 3) the codes

$$\mathrm{RM}(4,8) \otimes \mathrm{RM}(3,7), \mathrm{RM}(4,8) \otimes \mathrm{RM}(3,8), \mathrm{RM}(4,8) \otimes \mathrm{RM}(2,8) \qquad (14)$$

were considered resistant.

According to the table 1, choosing $p$, which results in $K_p$ being greater than $k_1$, allows increasing the probability of attack success due to a significant increase in the probability of $P_2$ and a slight decrease in the probabilities of $P_1^{min}$ and $P_1^{avg}$. In particular, the table lists the values of $K_p$ from $k_1$ to the first such value at which the probability of $P_{attack}^{min}$ decreases. It can also be seen that as $r_1$ grows, the probability of attack success increases.

Also note that when calculating $N_g^{avg}$ using the formula (12) rounding down occurs to estimate the probability $P_{attack}^{avg}$. According to the table 2 for the codes considered in the table 1 we have $N_g^{avg} = n_1 - 1$, i.e. the number of bad blocks is 1. However, even in this case the probability $Q_{N_g^{avg}}$ is rather small. This means that actually the number of bad blocks is equal to zero with a high probability and the probability of $P_{attack}^{avg}$ given in the table 1 is much higher.

For cryptosystem $\mathrm{McE}(C)$, $C = \mathrm{RM}(r_1, m_1) \otimes \mathrm{RM}(r_2, m_2)$, according to (1), (3) for $r_1 \geqslant \lceil m_1/2 \rceil$ and $r_2 \geqslant \lceil m_2/2 \rceil$ the condition (7) is not satisfied,

namely $C^2 = \mathbb{F}_2^{n_1 n_2}$. Note that for $\lceil m_1/3 \rceil \leqslant r_1 < \lceil m_1/2 \rceil$ and $\lceil m_2/3 \rceil \leqslant r_2 < \lceil m_2/2 \rceil$ the code $C^2$ is indecomposable as a tensor product of two indecomposable codes, and $C^3 = \mathbb{F}_2^{n_1 n_2}$. Therefore, in these cases, according to remark 1, the AttackDKey algorithm is not applicable and, accordingly, the AttackDCipher attack is not applicable. Therefore, the cryptosystem based on codes from table 3 of [35], other than (14), is resistant to the AttackDCipher attack.

### 4.2 General case $C = \overline{C(D)}$

Let's consider the cryptosystem McE($\overline{C(D)}$). Since $\overline{C(D)}$ has the form (2), then $\operatorname{rank}(\tau_i(G_{\overline{C(D)}})) = \dim(\overline{C_2(l_1)})$ for $\tau_i = \{(i-1)n_2+1, ..., in_2\}, i = 1, ..., n_1$. Moreover, from $\overline{C_2(l_1)} \supset \overline{C_2(l_2)} \supset ... \supset \overline{C_2(l_s)}$ follows $\overline{C(D)} \subset \mathbb{F}_q^{n_1} \otimes \overline{C_2(l_1)}$. Therefore, the conditions (6) are satisfied for the code $\overline{C(D)}$.

In [22] conditions under which (7) is true for the case when $C_1(k_i), C_2(l_i)$ are Reed–Muller codes are found. That is, $\overline{C(D)}^v = \mathbb{F}_q^{n_1} \otimes \overline{C_2(l_1)}^v \neq \mathbb{F}_q^{n_1 n_2}$. Therefore, the AttackDCipher attack can be applied to such codes. The table 3 gives an estimate of the probability $P_{attack}$ of the success of the attack AttackDCipher for $[n, k, d]$–code $\overline{C(D)}$ based on Reed–Muller codes $\mathrm{RM}(r, m)$ of length $n_1 = 2^{m_1} = n_2 = 2^{m_2}$, for $m_1 = m_2 = 8$. Namely, the first column of the table contains the $D$–code notation, where $[[r_1^1, r_1^2], [r_2^1, r_2^2], ...] = \mathrm{RM}(r_1^1, 8) \otimes \mathrm{RM}(r_1^2, 8) + \mathrm{RM}(r_2^1, 8) \otimes \mathrm{RM}(r_2^2, 8) + ....$ Also, for each $D$–code, the value $K_p$ for the step 4 of the algorithm 1 is experimentally found for $p = 0.01$. It can be seen that the probability $P_{attack\_avg}$ of finding an informational message using AttackDCipher is significantly greater than the probability $P_{ISD}$ of a successful attack on a ciphertext based on information set decoding. According to the formula (9), the minimum number of good blocks for codes from the table 3 is 241. Note that in this case the probability $P_{attack\_min}$ of success of the attack for some codes from the table 3 may be equal to zero. However, according to the table 4, the probability of such an event is negligible.

Table 3: AttackDCipher attack success probability for $D$–codes based on Reed–Muller codes

| D–code | $p$ | $K_p$ | $P_{ISD}$ | $P_{attack\_min}$ | $P_{attack\_avg}$ |
|---|---|---|---|---|---|
| [[4, 3], [5, 2]] | 0.01 | 219 | 1.013E-34 | 1.117E-16 | 0.145 |
| [[4, 3], [5, 2], [6, 1]] | 0.01 | 247 | 2.646E-35 | 0 | 0.011 |
| [[4, 3], [5, 2], [6, 1], [7, 0]] | 0.01 | 255 | 2.536E-35 | 0 | 0.004 |

Table 4: Probability of occurrence of bad blocks for $D$–codes from table 3

| $N_g$ | $n_1 - N_g$ | $Q_{N_g^{avg}}$ | $N_g$ | $n_1 - N_g$ | $Q_{N_g^{avg}}$ |
|---|---|---|---|---|---|
| 256 | 0 | 0.999 | 248 | 8 | 3.850E-113 |
| 255 | 1 | 1.832E-12 | 247 | 9 | 1.724E-130 |
| 254 | 2 | 6.462E-25 | 246 | 10 | 7.830E-149 |
| 253 | 3 | 5.416E-38 | 245 | 11 | 2.466E-168 |
| 252 | 4 | 1.111E-51 | 244 | 12 | 3.142E-189 |
| 251 | 5 | 5.377E-66 | 243 | 13 | 7.092E-212 |
| 250 | 6 | 5.656E-81 | 242 | 14 | 6.895E-237 |
| 249 | 7 | 1.149E-96 | 241 | 15 | 1.415E-265 |

In [22] the conditions on a $D$-code on Reed-Muller codes are found under which the cryptosystem McE($\overline{C(D)}$) is guaranteed to be resistant to a structural attack AttackDKey and, accordingly, to the AttackDCipher attack. In particular, in Theorems 3, 4, 5 of [22] conditions under which $\overline{C(D)}^2 = \mathbb{F}_2^{m_1 n_2}$ are obtained. The table 5 lists $D$-codes on Reed–Muller codes for $m_1 = m_2 = 8$ that resistant to AttackDKey and AttackDCipher attacks. The first column of the table contains the $D$–code notation, which is completely analogous to the representation from the table 3. The table also presents the parameters of the corresponding codes and an estimate of the probability $P_{ISD}$ of the success of the attack based on information set decoding. The $D$–codes were chosen according to the following rule. Among $D$–codes of the form (2), which have guaranteed resistance to the above attacks, for each $s = 2, ..., 9$ the code with the highest resistance to the information set decoding attack was chosen.

According to the results presented in the table 5, an increase in the number of addends in a $D$–code makes it possible to increase the resistance to information set decoding attack by increasing the dimension and preserving the minimum code distance. However, note that, in accordance with Lemma 2 from [22], code number 1 from the table 5 is the Reed–Muller code RM$(8, 16)$, and code number 2 its subcode of codimension 1. According to [9], the cryptosystem McE(RM$(r, m)$) is not resistant to structural attacks, and it's shown in [36] that subcodes of Reed–Muller codes of codimension 1 are also not resistant to structural attacks. Therefore, despite the resistance to AttackDKey and AttackDCipher attacks, $D$–codes 1 and 2 from the table 5 cannot be used in the McEliece–type cryptosystem.

The table 6 presents a comparison of McEliece type cryptosystems based on Goppa codes, Reed–Muller codes and $D$–codes based on Reed–Muller

Table 5: $D$–codes, resistant to AttackDKey and AttackDCipher attacks

| No. | $D$–code | $k$ | $d$ | $P_{ISD}$ |
|---|---|---|---|---|
| 1* | $[[0,8],[1,7],[2,6],[3,5],[4,4],[5,3],[6,2],[7,1],[8,0]]$ | 39203 | 256 | 1.715E-51 |
| 2* | $[[0,8],[1,7],[2,6],[3,5],[4,4],[5,3],[6,2],[7,1]]$ | 39202 | 256 | 1.723E-51 |
| 3 | $[[1,7],[2,6],[3,5],[4,4],[5,3],[6,2],[7,1]]$ | 39201 | 256 | 1.732E-51 |
| 4 | $[[1,7],[2,6],[3,5],[4,4],[5,3],[6,2]]$ | 39129 | 256 | 2.458E-51 |
| 5 | $[[2,6],[3,5],[4,4],[5,3],[6,2]]$ | 39057 | 256 | 3.486E-51 |
| 6 | $[[2,6],[3,5],[4,4],[5,3]]$ | 38021 | 256 | 4.796E-49 |
| 7 | $[[3,5],[4,4],[5,3]]$ | 36985 | 256 | 5.498E-47 |
| 8 | $[[2,5],[4,3]]$ | 19821 | 512 | 7.279E-41 |
| 9 | $[[4,4]]$ | 26569 | 256 | 1.156E-29 |

Table 6: Comparison of the characteristics of McEliece–type cryptosystems

| Code | Goppa code | Reed–Muller code | | $D$–code | |
|---|---|---|---|---|---|
| $[n,k,d]$ | $[3488, 2720, \geqslant 129]$ | $[65536, 14893, 1024]$ | $[65536, 39203, 256]$ | $[65536, 19821, 512]$ | $[65536, 39201, 256]$ |
| Size of publ. key | 1.13Mb | 116.35Mb | 306.27Mb | 154.85Mb | 306.25Mb |
| $k/n$ | $\approx 0.78$ | $\approx 0.23$ | $\approx 0.6$ | $\approx 0.3$ | $\approx 0.6$ |
| Decoder | Patterson decoding | Reed decoding | | majority–logical decoding | |
| $t = \lfloor (d-1)/2 \rfloor$ | 64 | 511 | 127 | 255 | 127 |
| $P_{ISD}$ | $2^{-142.8}$ | $2^{-192.62}$ | $2^{-169.37}$ | $2^{-136.16}$ | $2^{-169.37}$ |
| Structural attacks | $-$ | $+$ | | $-$ | |

codes (codes with numbers 3 and 8 from the table 5). The parameters of the selected Goppa code correspond to the mceliece348864 cryptosystem from the specification [37]. Note that this cryptosystem is among the finalists of the NIST competition for a post-quantum asymmetric encryption algorithm. The table shows that the cryptosystem based on $D$–codes in the case of using the majority–logical decoder has a much larger public key size than in the mceliece348864 system, with comparable security. However, we note that the cryptosystem based on $D$–codes is close in characteristics to the system based on Reed–Muller codes when using decoders with guaranteed error correction. But a significant advantage of a cryptosystem on $D$–codes is that the existing effective structural attacks from [8] and [9] on a system based on Reed–Muller codes are not directly applicable to a system on $D$–codes. Moreover, according to [22], in some cases a $D$–code is a subcode of some Reed–Muller code. This means that decoders operating beyond half the

code distance can be applied to $D$–codes (see the review [29]), in particular, the Sidelnikov–Pershakov decoder [38], its modifications and Dumer's list decoder, as well as a decoder for low-density codes [39] and a permutation decoder [40]. This can reduce the key size while maintaining security level. At the same time, the results of the analysis of the structural security of the system based on $D$–codes, obtained in this work, are also applicable when using other decoders. Thus, this work seems to be the basis for further research of the cryptosystem based on $D$–codes.

# References


[1] Bernstein D. J., Chou T., Lange T., von Maurich I., Misoczki R., Niederhagen R., ..., Wang W., "Classic McEliece: conservative code-based cryptography", *NIST submissions*, 2017.

[2] "Post-Quantum Cryptography", http://csrc.nist.gov/projects/post-quantum-cryptography, *National Institute of Standards and Technology (NIST).*

[3] Niederreiter, H., "Knapsack-type cryptosystems and algebraic coding theory", *Prob. Contr. Inform. Theory*, **15**:2 (1986), 157–166.

[4] Sidel'nikov, V. M., "Open coding based on Reed–Muller binary codes", *Diskretnaya matematika*, **6**:2 (1994), 3–20.

[5] Janwa H., Moreno O., "McEliece public key cryptosystems using algebraic-geometric codes", *Designs, Codes and Cryptography*, **8**:3 (1996), 293–307.

[6] Baldi M., Bianchi M., Chiaraluce F., "Security and complexity of the McEliece cryptosystem based on quasi-cyclic low-density parity-check codes", *IET Information Security*, **7**:3 (2013), 212–220.

[7] Sidel'nikov V. M., Shestakov S. O., "O sisteme shifrovaniya, postroennoj na osnove obobshchennyh kodov Rida–Solomona", *Diskretnaya matematika*, **4**:3 (1992), 57–63.

[8] Minder L., Shokrollahi A., "Cryptanalysis of the Sidelnikov cryptosystem", *In Advances in Cryptology-EUROCRYPT 2007: 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Barcelona, Spain, May 20-24, 2007. Springer Berlin Heidelberg*, **26** (2007), 347–360.

[9] Borodin M. A., Chizhov I. V., "Effective attack on the McEliece cryptosystem based on Reed-Muller codes", *Discrete Mathematics and Applications*, **24**:5 (2014), 273–280.

[10] Couvreur A., Márquez-Corbella I., Pellikaan R., "Cryptanalysis of McEliece cryptosystem based on algebraic geometry codes and their subcodes", *IEEE Transactions on Information Theory*, **63**:8 (2017), 5404–5418.

[11] Fabšič T., Hromada V., Stankovski P., Zajac P., Guo Q., Johansson T., "A reaction attack on the QC-LDPC McEliece cryptosystem. In Post-Quantum Cryptography: 8th International Workshop, PQCrypto 2017, Utrecht, The Netherlands, June 26-28, 2017, Proceedings 8", *Springer International Publishing*, 2017, 51-68.

[12] Couvreur A., Otmani A., Tillich J. P., "Polynomial time attack on wild McEliece over quadratic extensions.", *IEEE Transactions on Information Theory*, **63**:1 (2016), 404–427.

[13] Elbro F., Majenz C., "An Algebraic Attack Against McEliece-like Cryptosystems Based on BCH Codes", *Cryptology ePrint Archive*, 2022.

[14] Couvreur A., Lequesne M., "On the Security of Subspace Subcodes of Reed–Solomon Codes for Public Key Encryption", *IEEE Transactions on Information Theory*, 2021.

[15] Egorova E., Kabatiansky G., Krouk E., Tavernier C., "A new code-based public-key cryptosystem resistant to quantum computer attacks. In Journal of Physics: Conference Series", *IOP Publishing*, **1163**:1, 012061.

[16] Zyablov V. V., Ivanov F. I., Kruk E. A., Sidorenko V. R., "O novyh zadachah v asimmetrichnoj kriptografii, osnovannoj na pomekhoustojchivom kodirovanii", *Problemy peredachi informacii*, **58**:2 (2022), 92–111.

[17] Deundyak V. M., Kosolapov Y. V., Maystrenko I. A., "On the decipherment of sidel'nikov-type cryptosystems. In Code-Based Cryptography: 8th International Workshop, CBCrypto 2020, Zagreb, Croatia, May 9–10, 2020, Revised Selected Papers", *Springer International Publishing*, 2020, 20–40.

[18] Vedenev K., Kosolapov Y., "Cryptanalysis of Ivanov-Krouk-Zyablov cryptosystem", *In Code-Based Cryptography Workshop. Springer, Cham*, 2023, 137–153.

[19] Xu J., Chaichanavong P., Burd G., Wu Z., "U.S. Patent No. 7,861,131", *Washington, DC: U.S. Patent and Trademark Office*, 2010.

[20] Twitto M., Ari M. B., Dor A., Erez E., Kong J. J., Shany Y., "U.S. Patent No. 10,333,554", *Washington, DC: U.S. Patent and Trademark Office*, 2019.

[21] Yeo E., Yadav M. K., Chaichanavong P., Burd G., "U.S. Patent No. 8,321,769", *Washington, DC: U.S. Patent and Trademark Office*, 2012.

[22] Kosolapov Yu. V., Lelyuk E. A., "O strukturnoj stojkosti kriptosistemy tipa Mak-Elisa na summe tenzornyh proizvedenij binarnyh kodov Rida-Mallera", *PDM*, **57** (2022), 22—39.

[23] Kasami T., Lin S., "On the Construction of a Class of Majority-Logic Decodable Codes", *IEEE Transactions on Information Theory*, **IT-17**:5 (1971), 600–610.

[24] Deundyak V. M., Lelyuk E. A., "A graph-theoretical method for decoding some group MLD-codes", *Journal of Applied and Industrial Mathematics*, **14**:2 (2020), 265–280.

[25] Morelos-Zaragoza R. H., *The Art of Error Correcting Coding*, John Wiley & Sons, Ltd., 2006.

[26] Randriambololona H., "On products and powers of linear codes under componentwise multiplication", *Algorithmic Arithmetic Geometry Coding Theory*, **637** (2015), 3—78.

[27] Deundyak V. M., Kosolapov Yu. V., "O nekotoryh svojstvah proizvedeniya Shura-Adamara dlya linejnyh kodov i ih prilozheniyah", *Prikladnaya diskretnaya matematika*, **50** (2020), 72-–86.

[28] D. Slepian, "Some further theory of group codes", *Bell Syst. Tech. J.*, **39**:5 (1960), 1219—1252.

[29] Abbe E., Shpilka A., Ye M., "Reed–Muller codes: Theory and algorithms", *IEEE Transactions on Information Theory*, **67**:6 (2020), 3251–3277.

[30] McEliece R. J., "A Public-Key Cryptosystem Based On Algebraic Coding Theory", *DSN Progress Report*, 1978, 42–44.

[31] Dottling N., Dowsley R., Muller-Quade J., Nascimento A.C.A., "A CCA secure variant of the McEliece cryptosystem", *IEEE Trans. Inf. Theory*, **58** (2012), 6672—6680.

[32] Sachkov V. N., *Vvedenie v kombinatornye metody diskretnoj matematiki*, MCNMO, 2004.

[33] R. Brent, S. Gao, A. Lauder, "Random Krylov Spaces over Finite Fields", *SIAM Journal on Discrete Mathematics*, **16**:2 (2002), 276-–287.

[34] Deundyak V. M., Kosolapov Yu. V., Lelyuk E. A., "Decoding the Tensor Product of MLD Codes and Applications for Code Cryptosystems", *Automatic Control and Computer Sciences*, **52**:7 (2018), 647–657.

[35] Deundyak V.M., Kosolapov Yu.V., "Ispol'zovanie tenzornogo proizvedeniya kodov Rida-Mallera v asimmetrichnoj kriptosisteme tipa Mak-Elisa i analiz ee stojkosti k atakam na shifrogrammu", *Vychislitel'nye tekhnologii*, **22**:4 (2017), 43–60.

[36] Borodin M. A., Chizhov I. V., "Klassifikaciya proizvedenij Adamara podkodov korazmernosti 1 kodov Rida-Mallera", *Diskretnaya matematika*, **32**:1 (2020), 115–134.

[37] Bernstein, Daniel J., et al., "Classic McEliece: conservative code-based cryptography: cryptosystem specification", *NIST submissions*, 2022.

[38] Sidel'nikov V. M., Pershakov A. S., "Dekodirovanie kodov Rida–Mallera pri bol'shom chisle oshibok", *Probl. peredachi inform.*, **28**:3 (1992), 80–94.

[39] Geiselhart M., Elkelesh A., Ebada M., Cammerer S., Ten Brink S., "Iterative Reed–Muller Decoding", *In 2021 11th International Symposium on Topics in Coding (ISTC). IEEE*, 2021, 1–5.

[40] Kamenev M., Kameneva Y., Kurmaev O., Maevskiy A., "A new permutation decoding method for Reed-Muller codes", *In 2019 IEEE International Symposium on Information Theory (ISIT). IEEE*, 2019, 26–30.

# Mathematical aspects

# Properties of generalized bent functions and decomposition of Boolean bent functions

Aleksandr Kutsenko

Sobolev Institute of Mathematics, Novosibirsk, Russia
Novosibirsk State University, Novosibirsk, Russia
alexandrkutsenko@bk.ru

**Abstract**

Functions of the form $\mathbb{F}_2^n \to \mathbb{Z}_q$, where $q \geqslant 3$ is a positive integer, with flat generalized Walsh–Hadamard spectrum are called generalized bent (gbent) functions. Gbent functions for which it is possible to define a dual gbent function are called regular. Within the Boolean functions the duality mapping is the unique known isometric mapping that preserves bentness and is not an element of the group of automorphisms of the set of bent functions but for the generalized case the study of its properties is an open problem. A regular gbent function is said to be self-dual if it coincides with its dual, in this case its polyphase vector is an eigenvector of the Sylvester–Hadamard matrix.

We completely characterize affine gbent functions and investigate the conditions of regularity for them. The form of the dual gbent function of affine gbent function is obtained and it appears that it is also affine. We show that affine gbent functions cannot be self-dual. The form of the dual functions is used to prove that the duality mapping is an isometry of the set of regular affine gbent functions. It is proved that the sets of quaternary self-dual and anti-self-dual gbent functions are metrically regular within the Lee distance, that provides a non-trivial example of pair of sets of quaternary vectors with such property.

We generalize known results («On subfunctions of self-dual bent functions», Proc. of CTCrypt2021, p. 210–239) regarding decomposition of self-dual Boolean bent functions of the form $(f_0, f_1, f_2, f_3)$ and adapt them for the case of an arbitrary bent function. The general forms of the Gram matrix of subfunctions for a bent function and its dual one are studied. By using the obtained form of the Gram matrix we prove that if characteristic vectors of subfunctions of a bent function are linearly dependent, then all these subfunctions are bent functions. It is also interesting since the subfunctions of its dual bent function are bent as well.

**Keywords:** Generalized bent function, self-dual bent, metrical regularity, Lee distance, duality mapping

## 1 Introduction

The study of Boolean functions having strong cryptographic properties is the domain of current interest. The notion of Boolean bent function was

introduced by Rothaus in 1976 [39]. But it is known that these functions were also studied in the Soviet Union in 1960s, see [24, 52]. Due to maximal nonlinearity and flat Walsh–Hadamard spectrum they have a number of applications in cryptography and coding theory. These functions are not balanced and their algebraic degree can not exceed $n/2$, so rather than direct usage in ciphers they can be used in a field of obtaining constructions of (vectorial) Boolean functions with strong cryptographic properties by analytical methods or evolutionary programming and heuristics. There are examples of their direct usage in ciphers, in particular, they were used as building blocks of stream (Grain, 2004) and block (CAST, 1997) ciphers. But despite the long history of research in this area there are still many open problems, in particular, problems regarding cardinality of the set of bent functions, design of new construction and studying new properties. One can find more details on bent functions in books [52, 33] and survey [6].

For every bent Boolean function its dual bent function, that describes the signs of its Walsh–Hadamard coeffecients, is uniquely defined. The duality mapping is the unique known isometric mapping of the set of bent functions into itself that preserves bentness and is not an element of the group of their group of automorphisms [4]. Self-dual bent functions form a remarkable class of bent functions since they have the direct relation to their dual bent functions and in terms of mappings can be considered as fixed points of the duality mapping. These functions were explored by Carlet et al. in 2010 in work [5], where a number of constructions and properties were given and the classification for small number of variables was provided. Later the classification, constructions and properties of self-dual Boolean bent functions were extensively studied, see [17, 19, 11, 32, 29, 26, 48]. The overview of the known metrical properties of self-dual bent functions can be found in [22].

Bent functions were initially generalized by Kumar et al. in 1985 by considering functions of the form $\mathbb{Z}_q^n \to \mathbb{Z}_q$ with corresponding definition of bentness, see [20]. Bent functions from a finite Abelian group into a finite Abelian group were studied in 1997 by Logachev, Sal'nikov, Yashchenko in [28] and in 2002 by Solodovnikov [46]. Having applications of functions from $\mathbb{F}_2^n$ to $\mathbb{Z}_4$ in code-division multiple access (CDMA) systems, Schmidt in [41] (initially appeared in preprint from 2006) generalized the notion of bentness for functions of the form $\mathbb{F}_2^n \to \mathbb{Z}_q$, where $q \geqslant 2$ is a positive integer and studied these functions for the case $q = 4$. The considered functions are named generalized bent (gbent) functions. A comprehensive survey on existing generalizations of bent functions can be found in [50].

Gbent functions, at least for quaternary case, can be used for the construc-

tion of optimal and nearly optimal codebooks, as was shown in [12, 53]. The connection between concepts of strong regularity of (edge-weighted) Cayley graph associated to a generalized Boolean function and gbent functions was pointed in [38]. Note that the generalization of this type deals with *generalized Boolean functions* that are essentially related to the $q$-ary generalization $\mathrm{RM}_q(r, n)$ of the classic Reed–Muller codes, that allows to obtain linear codes with special properties, see [8, 36, 40]. A close connection between generalized Boolean functions and vectorial Boolean functions was shown in [35]. Note that for generalized case the choice of metrics and definition of distance significantly affects the considered properties. In mentioned papers for generalized Reed–Muller codes the Hamming, Lee and Euclidean distances were considered.

In recent years generalized bent functions obtained much attention, in particular, for the case $q = 2^k$, $k \geqslant 2$. Starting from quaternary case [45], different constructions and properties of generalized bent functions were obtained for $q = 4, 8$ in [47] and even $q$ in [43, 13, 14]. The questions of existence were arisen in [27, 25]. The complete characterization of generalized bent functions from different perspectives was recently presented in works [49, 15, 34] and, in particular, the connection between gbent functions and affine spaces of Boolean bent functions or semi-bent functions (depending on the parity of $n$) with additional properties was discovered [15].

The action of the duality mapping on bent functions within generalizations is increasingly nontrivial. It is defined only for the part of bent functions from corresponding generalization which are called *regular*. Nevertheless it is known that gbent functions with values in $\mathbb{Z}_{2^k}$, $k \geqslant 2$, which is one of the most interesting cases, are regular for both $n$ even and odd except the case $q = 4$ and odd $n$ [30].

The extension of the concept of self-duality for different generalizations of bent functions was made in several papers. The classification of quadratic self-dual bent functions of the form $\mathbb{F}_p^n \to \mathbb{F}_p$, $p$ odd prime, was made by Hou in [18]. Çeşmelioğlu et al. in paper [7] studied self-duality for bent functions within the same generalization type and in [9] proposed two concepts of self-duality for vectorial bent functions, self-duality and weak self-duality. In 2018, Sok et al. studied quaternary self-dual bent functions of the form $\mathbb{F}_2^n \to \mathbb{Z}_4$ from the viewpoints of existence, construction and symmetry [44]. The relation between sign functions of quaternary self-dual bent function in $n$ variables and two self-dual bent functions in $n$ variables was found. Based on this it was proved that there are no quaternary self-dual bent functions in odd number of variables. Self-dual bent sequences associated to

complex Hadamard matrices were studied by Shi et al. in [42].

Every generalized Boolean function in $n$ variables can be uniquely represented as a sum modulo $q$ of a certain number of (component) Boolean functions in $n$ variables with coefficients that are powers of 2, see [47]. For the case $q = 2^k$ this decomposition was completely studied for even $n$ in paper [15] and odd $n$ in [34]. Regarding self-duality one can show that component functions of every self-dual gbent function are self-dual bent up to some fixed shift that is every self-dual gbent function generates a set of self-dual Boolean bent functions. Therefore, the study of constructions and properties, including cardinality, of self-dual gbent functions can also be interesting from the perspective of obtaining some new results for self-dual Boolean bent functions.

In current work we study special classes of generalized bent functions and also obtain new results regarding the decomposition of bent functions. The paper is organized as follows. Section 2 contains the necessary notation. In section 3 affine generalized bent functions are characterized, the question of their regularity and the desription of their component Boolean functions are studied in sections 3.2 and 3.3, correspondingly. In Section 3.4 we touch upon the question of isometric mappings and the duality mapping. Section 4 is devoted to the study of metrical properties of the set of quaternary self-dual bent functions, namely the functions which are maximally distant from them. In Section 5 the Gram matrices are studied for the case of an arbitrary bent function with arespect to the decomposition of its vector of values or characteristic vector. The known results on self-dual Boolean bent functions are generalized for the set of all bent functions. Conclusion is in Section 6.

## 2 Notation

Let $\mathbb{F}_2^n$ be a set of binary vectors of length $n$. For $x, y \in \mathbb{F}_2^n$ denote $\langle x, y\rangle = \bigoplus_{i=1}^{n} x_i y_i$, where the sign $\oplus$ denotes a sum modulo 2. A *generalized Boolean function* $f$ in $n$ variables is any map from $\mathbb{F}_2^n$ to $\mathbb{Z}_q$, the integers modulo $q$. The set of generalized Boolean functions in $n$ variables is denoted by $\mathcal{GF}_n^q$, for the case $q = 2$ we use notation $\mathcal{F}_n$.

Let $\omega = e^{2\pi i/q}$. A *polyphase vector* (sequence) of $f \in \mathcal{GF}_n^q$ is a complex-valued vector

$$F = \omega^f = \left(\omega^{f_0}, \omega^{f_1}, \ldots, \omega^{f_{2^n-1}}\right)$$

of length $2^n$, where $(f_0, f_1, \ldots, f_{2^n-1})$ is a vector of values of the function $f$. The vector $F$ can be seen as a $q$-ary phase-shift-keying constellation. For

Boolean case $q = 2$ we use notion *characteristic vector*. Any (generalized) Boolean function in $n$ variables can be uniquely represented via the multivariate polynomial over $\mathbb{Z}_q$:

$$f(x_1, x_2, \ldots, x_n) = \bigoplus_{i_1, i_2, \ldots i_n \in \mathbb{F}_2} a_{i_1 i_2 \ldots i_n} x_1^{i_1} x_2^{i_2} \cdot \ldots \cdot x_n^{i_n},$$

where $a_y \in \mathbb{Z}_q$ for all $y \in \mathbb{F}_2^n$, see [8]. Here we use the agreement $0^0 = 1$. This representation is called the *algebraic normal form* (ANF) of the (generalized) Boolean function $f$. For the case $q = 2$ it is called *Zhegalkin polynomial*. The *degree* $\deg(f)$ of the function $f$ is the maximal degree (number of terms) of the monimial from its algebraic normal form that has nonzero coefficient. If $\deg(f) \leqslant 1$, the function is called *affine*. If $\deg(f) = 2$, the function is said to be *quadratic*.

The *Hamming weight* $\mathrm{wt}_H(x)$ of the vector $x \in \mathbb{F}_2^n$ is the number of nonzero coordinates of $x$. The *Hamming distance* $\mathrm{dist}_H(f, g)$ between generalized Boolean functions $f, g$ in $n$ variables is the cardinality of the set $\{x \in \mathbb{F}_2^n | f(x) \neq g(x)\}$. The Lee weight of the element $x \in \mathbb{Z}_q$ is $\mathrm{wt}_L(x) = \min\{x, q - x\}$. The Lee weight of generalized Boolean function is the sum of Lee weights of all its values:

$$\mathrm{wt}_L(f) = \sum_{x \in \mathbb{F}_2^n} \mathrm{wt}_L(f(x)).$$

The *Lee distance* $\mathrm{dist}_L(f, g)$ between $f, g \in \mathcal{GF}_n^q$ is equal to $\mathrm{wt}_L(f - g)$, where the operation " $-$ " is considered over the ring $\mathbb{Z}_q$. For Boolean case $q = 2$ the Hamming distance coincides with the Lee distance.

The *Walsh–Hadamard transform* of $f \in \mathcal{F}_n$ is the integer function:

$$W_f(y) = \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x) \oplus \langle x, y \rangle}.$$

A Boolean function $f$ in $n$ variables is said to be *bent* if

$$|W_f(y)| = 2^{n/2}$$

for all $y \in \mathbb{F}_2^n$ [39].

The *(generalized) Walsh–Hadamard transform* of $f \in \mathcal{GF}_n^q$ is the complex valued function:

$$H_f(y) = \sum_{x \in \mathbb{F}_2^n} \omega^{f(x)} (-1)^{\langle x, y \rangle}.$$

A generalized Boolean function $f$ in $n$ variables is said to be *generalized bent* (gbent) if

$$|H_f(y)| = 2^{n/2},$$

for all $y \in \mathbb{F}_2^n$ [41] (initially appeared in preprint from 2006). If there exists such $\widetilde{f} \in \mathcal{GF}_n^q$ that $H_f(y) = \omega^{\widetilde{f}(y)} 2^{n/2}$ for any $y \in \mathbb{F}_2^n$, the gbent function $f$ is said to be *regular* and $\widetilde{f}$ is called its *dual*. Note that $\widetilde{f}$ is generalized bent as well. If there exists such $\widetilde{f} \in \mathcal{GF}_n^q$ that for any $y \in \mathbb{F}_2^n$ it holds $H_f(y) = \zeta \omega^{\widetilde{f}(y)} 2^{n/2}$, where $\zeta \in \mathbb{C}$ and $|\zeta| = 1$, the gbent function $f$ is said to be *weakly regular*.

A regular gbent function $f$ is said to be *self-dual* if $f = \widetilde{f}$, and *anti-self-dual* if $f = \widetilde{f} + q/2$. Consequently, it is the case only for even $q$. So, throughout this paper we assume that $q$ is a positive even integer. Corresponding sets of gbent functions are denoted by $\mathcal{SB}_n^{q,+}$ and $\mathcal{SB}_n^{q,-}$, respectively. For $q = 2$ we denote them by $\mathcal{SB}_n^+$ and $\mathcal{SB}_n^-$

## 3 Affine gbent functions and their duals

It is well known that due to the maximal nonlinearity Boolean bent functions can not be affine, but in case of generalized Boolean functions the situation is not so trivial. Gbent functions of the form

$$f(x) = \sum_{j=1}^{n} \lambda_j x_j + \lambda_0, \quad x \in \mathbb{F}_2^n, \tag{1}$$

where $\lambda_0, \lambda_1, \ldots, \lambda_n \in \mathbb{Z}_q$ are referred to as *affine* functions. They were studied by Singh in [43] for the case when $q$ is divisible by 4 and it was shown that if $\lambda_j \in \{\frac{q}{4}, \frac{3q}{4}\}$ for any $j = 1, 2 \ldots, n$, these functions are gbent. Also, necessary and sufficient conditions for the bentness of affine generalized Boolean functions were obtained, namely

$$\prod_{j=1}^{n} \left(1 + (-1)^{y_j} \cos \frac{2\pi\lambda_j}{q}\right) = 1 \text{ for any } y \in \mathbb{F}_2^n.$$

For affine generalized Boolean functions one can obtain more convenient of the Walsh–Hadamard coefficients (see also [43])

$$\begin{aligned} H_f(y) &= \sum_{x \in \mathbb{F}_2^n} \omega^{f(x)} (-1)^{\langle x,y \rangle} = \omega^{\lambda_0} \sum_{x \in \mathbb{F}_2^n} \omega^{\sum\limits_{j=1}^{n} \lambda_j x_j + \frac{q}{2}\langle x,y \rangle} \\ &= \omega^{\lambda_0} \prod_{j=1}^{n} \sum_{x_j \in \mathbb{F}_2} \omega^{\lambda_j x_j + \frac{q}{2} y_j x_j} = \omega^{\lambda_0} \prod_{j=1}^{n} \left(1 + \omega^{\frac{q}{2} y_j + \lambda_j}\right) \end{aligned}$$

for any $y \in \mathbb{F}_2^n$.

### 3.1 Non-existence of affine self-dual gbent functions

The next result shows the non-existence of self-dual gbent functions within the class of affine functions.

**Proposition 1.** *There are no affine self-dual gbent functions.*

*Proof.* Let $f$ be an affine gbent function in $n$ variables (for the case $q$ not divisible by 4 if such exists, otherwise the result follows) of the form (1). It is self-dual if and only if

$$H_f(y) = \omega^{\lambda_0} \prod_{j=1}^{n} \left(1 + \omega^{\frac{q}{2} y_j + \lambda_j}\right) = 2^{n/2} \omega^{\sum\limits_{j=1}^{n} \lambda_j y_j + \lambda_0}.$$

for every $y \in \mathbb{F}_2^n$. Denote

$$\begin{aligned} \widehat{y} &= (y_1, y_2, \ldots, y_{n-1}) \in \mathbb{F}_2^{n-1}, \\ P_{n-1}(\widehat{y}) &= \left(1 + \omega^{\frac{q}{2} y_1 + \lambda_1}\right)\left(1 + \omega^{\frac{q}{2} y_2 + \lambda_2}\right) \cdot \ldots \cdot \left(1 + \omega^{\frac{q}{2} y_{n-1} + \lambda_{n-1}}\right), \\ a_{n-1}(\widehat{y}) &= \lambda_1 y_1 + \lambda_2 y_2 + \ldots + \lambda_{n-1} y_{n-1}. \end{aligned}$$

Then for any $y \in \mathbb{F}_2^n$ such that $y_n = 0$ it holds

$$P_{n-1}(\widehat{y})\left(1 + \omega^{\lambda_n}\right) = 2^{n/2} \omega^{a_{n-1}(\widehat{y})},$$

and for any $y \in \mathbb{F}_2^n$ such that $y_n = 1$:

$$P_{n-1}(\widehat{y})\left(1 + \omega^{\frac{q}{2} + \lambda_n}\right) = 2^{n/2} \omega^{a_{n-1}(\widehat{y}) + \lambda_n}.$$

So, for any $\widehat{y} \in \mathbb{F}_2^{n-1}$ consider the system

$$\begin{cases} P_{n-1}(\widehat{y})\left(1 + \omega^{\lambda_n}\right) = 2^{n/2} \omega^{a_{n-1}(\widehat{y})}, \\ P_{n-1}(\widehat{y})\left(1 - \omega^{\lambda_n}\right) = 2^{n/2} \omega^{a_{n-1}(\widehat{y}) + \lambda_n}. \end{cases}$$

It is equivalent to

$$\begin{cases} P_{n-1}(\widehat{y})\left(1 + \omega^{\lambda_n}\right) = 2^{n/2} \omega^{a_{n-1}(\widehat{y})}, \\ P_{n-1}(\widehat{y})\left(1 - \omega^{\lambda_n}\right) = P_{n-1}(\widehat{y})\left(1 + \omega^{\lambda_n}\right) \cdot \omega^{\lambda_n}. \end{cases}$$

Thus, we obtain the relation

$$P_{n-1}(\widehat{y})\left(1 - \omega^{\lambda_n}\right) = P_{n-1}(\widehat{y})\left(1 + \omega^{\lambda_n}\right) \cdot \omega^{\lambda_n},$$

and note that $P_{n-1}(\widehat{y}) \neq 0$ since for any $y \in \mathbb{F}_2^n$ the number $|P_{n-1}(\widehat{y})|$ is a divisor of $2^{n/2}$. That is we have an equation $1 - \omega^{\lambda_n} = \omega^{\lambda_n} + \left(\omega^{\lambda_n}\right)^2$, but its solutions are $\left(-1 \pm \sqrt{2}\right)$. The norm of every of these numbers is not 1 therefore $\omega^{\lambda_n}$ can not be a solution. □

### 3.2 Characterization of affine gbent functions and their duals

In the next result we prove that the values of coefficients, proposed in [43], are also necessary for affine generalized Boolean function to be gbent.

**Theorem 1.** *Affine generalized Boolean function of the form (1) is gbent if and only if* $q \equiv 0 \pmod 4$ *and* $\lambda_j \in \left\{\frac{q}{4}, \frac{3q}{4}\right\}$ *for any* $j = 1, 2, \ldots, n$.

*Proof.* The sufficiency was shown in [43]. For necessity note, again, that the function (1) is gbent if

$$|H_f(y)| = \left|\omega^{\lambda_0} \prod_{j=1}^{n} \left(1 + \omega^{\frac{q}{2} y_j + \lambda_j}\right)\right| = 2^{n/2}$$

for any $y \in \mathbb{F}_2^n$. Consider this relation for two arguments $u, v \in \mathbb{F}_2^n$ that are distinct in the coordinate with number $k \in \{1, 2, \ldots, n\}$ only. From $|H_f(u)| = |H_f(v)|$ we obtain the relation

$$\left|1 + \omega^{\lambda_k}\right| = \left|1 - \omega^{\lambda_k}\right|.$$

The only possibility for $\omega^{\lambda_k}$ is to be equal either $i$ or $-i$. In both cases we have to possess the condition $q \equiv 0 \pmod 4$ and $\lambda_k \in \left\{\frac{q}{4}, \frac{3q}{4}\right\}$. Since $k$ was chosen randomly, we have $\lambda_j \in \left\{\frac{q}{4}, \frac{3q}{4}\right\}$ for any $j = 1, 2 \ldots, n$. □

Consequently, further we will assume that $q \equiv 0 \pmod 4$. We can immediately deduce the exact number of such functions.

**Corollary 1.** *The number of affine gbent functions is equal to* $q \cdot 2^n$.

In [41] the quaternary function

$$f(x_1, x_2, \ldots, x_n) = x_1 + x_2 + \ldots + x_n$$

and its shifts were considered for obtaining a constant-amplitude code. It follows that for quaternary case it is gbent, moreover, it is regular but only for even $n$. A very similar construction of real-valued bent functions was proposed in [31] (Theorem 2).

The next result answers the question of the regularity of affine gbent function and provides the form of its dual, if exists.

**Theorem 2.** *Affine gbent function (1) is regular if at least one of conditions is satisfied:*

*1)* $n$ *is even;*

*2)* $q \equiv 0 \pmod 8$,

*and its dual gbent is equal to*

$$\widetilde{f}(x) = \sum_{j=1}^{n} (q - \lambda_j)\, x_j + \left( \lambda_0 + \frac{3q}{4} n + \frac{3}{2} \sum_{k=1}^{n} \lambda_k \right).$$

*If $n$ is odd and $q \equiv 4 \pmod 8$, gbent function is weakly regular, its dual is equal to*

$$\widetilde{f}(x) = \sum_{j=1}^{n} (q - \lambda_j)\, x_j + \left( \lambda_0 + \frac{3q}{4} n \right)$$

*and* $\zeta = \exp\left( \frac{3\pi i}{q} \sum\limits_{k=1}^{n} \lambda_k \right)$.

*Proof.* By using Theorem 1, denote $\lambda_j = \frac{q}{4} + \frac{q}{2} b_j$, where $b_j \in \mathbb{F}_2$, $j = 1, 2, \ldots, n$, are coordinates of binary vector $b \in \mathbb{F}_2^n$. For any $y \in \mathbb{F}_2^n$ we have

$$\begin{aligned}
H_f(y) &= \omega^{\lambda_0} \prod_{j=1}^{n} \left( 1 + \omega^{\frac{q}{2} y_j + \lambda_j} \right) = \omega^{\lambda_0} \prod_{j=1}^{n} \left( 1 + \omega^{\frac{q}{2} y_j + \frac{q}{4} + \frac{q}{2} b_j} \right) \\
&= \omega^{\lambda_0} (1 - i)^{\mathrm{wt}(y \oplus b)} (1 + i)^{n - \mathrm{wt}(y \oplus b)} \\
&= 2^{n/2} \omega^{\lambda_0} \exp\left( \frac{7\pi i}{4} \mathrm{wt}(y \oplus b) + \frac{\pi i}{4} (n - \mathrm{wt}(y \oplus b)) \right) \\
&= 2^{n/2} \omega^{\lambda_0} \exp\left( -\frac{\pi i}{2} n + \frac{3\pi i}{2} \sum_{j=1}^{n} y_j + \frac{3\pi i}{q} \sum_{k=1}^{n} \lambda_k - 3\pi i \sum_{l=1}^{n} y_l b_l \right) \\
&= 2^{n/2} \omega^{\lambda_0 + \frac{3q}{4} n} \exp\left( \pi i \sum_{j=1}^{n} y_j + \frac{3\pi i}{q} \sum_{k=1}^{n} \lambda_k + \frac{2\pi i}{q} \sum_{l=1}^{n} \lambda_l \right) \\
&= 2^{n/2} \omega^{\lambda_0 + \frac{3q}{4} n + \frac{q}{2} \sum\limits_{j=1}^{n} y_j + \sum\limits_{l=1}^{n} \lambda_l y_l} \exp\left( \frac{3\pi i}{q} \sum_{k=1}^{n} \lambda_k \right).
\end{aligned}$$

Here we use the fact that $a + \frac{q}{2} \equiv q - a \pmod q$ for $a \in \left\{ \frac{q}{4}, \frac{3q}{4} \right\}$, then

$$H_f(y) = 2^{n/2} \omega^{\lambda_0 + \frac{3q}{4} n + \sum\limits_{j=1}^{n} (q - \lambda_j) y_j} \exp\left( \frac{3\pi i}{q} \sum_{k=1}^{n} \lambda_k \right).$$

If $n$ is even or $q \equiv 0 \pmod 8$, then the number $\sum\limits_{k=1}^{n} \lambda_k$ is an even integer and we can write

$$e^{\frac{3\pi i}{q} \sum\limits_{k=1}^{n} \lambda_k} = \omega^{\frac{3}{2} \sum\limits_{k=1}^{n} \lambda_k}.$$

If $n$ is odd and $q \equiv 4 \pmod 8$, then $\sum\limits_{k=1}^{n} \lambda_k$ is an odd integer so we introduce $\zeta = \exp\left(\frac{3\pi i}{q} \sum\limits_{k=1}^{n} \lambda_k\right)$. Thus, finally we have

$$H_f(y) = 2^{n/2} \zeta \omega^{\sum\limits_{j=1}^{n} (q-\lambda_j)x_j + \left(\lambda_0 + \frac{3q}{4}\right)},$$

hence $f$ is regular if $n$ is even or $q \equiv 0 \pmod 8$, in this case its dual gbent is equal to

$$\widetilde{f}(x) = \sum_{j=1}^{n} (q - \lambda_j)\, x_j + \left(\lambda_0 + \frac{3q}{4} n + \frac{3}{2} \sum_{k=1}^{n} \lambda_k\right).$$

If $n$ is odd and $q \equiv 4 \pmod 8$, gbent function is weakly regular, its dual is equal to

$$\widetilde{f}(x) = \sum_{j=1}^{n} (q - \lambda_j)\, x_j + \left(\lambda_0 + \frac{3q}{4} n\right).$$

□

It follows that the dual of (1), if exists, is also affine. Its coefficients are the ones of $f$ that are reflected with a respect to $q$. At the same time the coefficient $\lambda_0$ is changed in another way. From this it also follows the non-existence of affine self-dual gbent functions.

We can also note an interesting fact

**Corollary 2.** *The polyphase vector $\omega^{\widetilde{f}}$ of the dual gbent function $\widetilde{f}$ of affine gbent function $f$ from (1) is equal to the complex conjugation of $\omega^f$ up to the global phase* $\exp\left(\frac{3\pi i}{2} + \frac{3\pi i}{q} \sum\limits_{k=1}^{n} \lambda_k\right)$.

It means the the duality mapping, acting on polyphase vectors of gbent functions, coincides with the conjugation up to the global phase, that depends on coefficients of affine gbent function.

### 3.3 Component Boolean functions of affine gbent functions and their duals

We can describe component functions from Theorem 2 for affine functions and its dual one. For this we consider the following auxiliary fact

**Lemma 1.** *For any $x \in \mathbb{F}_2^n$ it holds*

$$\bigoplus_{j=1}^{n} x_j = \mathrm{wt}(x) + \sum_{k=2}^{n} (-1)^{k-1} 2^{k-1} \sum_{1 \leqslant r_1 < r_2 < \ldots < r_k \leqslant n} x_{r_1} x_{r_2} \cdot \ldots \cdot x_{r_k}.$$

*Proof.* We are to use mathematical induction. The base of induction $n = 2$ is obvious. Show that if it holds for $n = m$, then it also holds for $n = m + 1$. We have

$$\begin{aligned}
\bigoplus_{j=1}^{m+1} x_j &= \bigoplus_{j=1}^{m} x_j \oplus x_{m+1} = \left( \bigoplus_{j=1}^{m} x_j \right) + x_{m+1} - 2 \left( \bigoplus_{j=1}^{m} x_j \right) x_{m+1} \\
&= \sum_{j=1}^{m} x_j + \sum_{k=2}^{m} (-1)^{k-1} 2^{k-1} \sum_{1 \leqslant r_1 < r_2 < \ldots < r_k \leqslant m} x_{r_1} x_{r_2} \cdot \ldots \cdot x_{r_k} + x_{m+1} \\
&- 2 \left( \sum_{l=1}^{m} x_l + \sum_{k=2}^{m} (-1)^{k-1} 2^{k-1} \sum_{1 \leqslant r_1 < r_2 < \ldots < r_k \leqslant m} x_{r_1} x_{r_2} \cdot \ldots \cdot x_{r_k} \right) x_{m+1} \\
&= \sum_{j=1}^{m+1} x_j + \sum_{k=2}^{m} (-1)^{k-1} 2^{k-1} \sum_{1 \leqslant r_1 < r_2 < \ldots < r_k \leqslant m+1} x_{r_1} x_{r_2} \cdot \ldots \cdot x_{r_k} \\
&- 2(-1)^{m-1} 2^{m-1} x_1 x_2 \cdot \ldots \cdot x_m x_{m+1} \\
&= \sum_{j=1}^{m+1} x_j + \sum_{k=2}^{m+1} (-1)^{k-1} 2^{k-1} \sum_{1 \leqslant r_1 < r_2 < \ldots < r_k \leqslant m+1} x_{r_1} x_{r_2} \cdot \ldots \cdot x_{r_k},
\end{aligned}$$

from which the result follows. □

The following Proposition describes Boolean decomposition of non-constant part of the function (1) and its dual.

**Proposition 2.** *If $f$ is an affine gbent function in $n$ variables of the form (1), then*

$$\begin{aligned}
f(x) &= \lambda_0 + \frac{q}{4} \bigoplus_{j=1}^{n} x_j + \frac{q}{2} \left( \bigoplus_{h=1}^{n} b_h x_h \oplus \bigoplus_{1 \leqslant r < s \leqslant n} x_r x_s \right) \\
\widetilde{f}(x) &= \widetilde{\lambda}_0 + \frac{q}{4} \bigoplus_{j=1}^{n} x_j + \frac{q}{2} \left( \bigoplus_{l=1}^{n} x_l \oplus \bigoplus_{h=1}^{n} b_h x_h \oplus \bigoplus_{1 \leqslant r < s \leqslant n} x_r x_s \right),
\end{aligned}$$

*where $b_j = \frac{2\lambda_j}{q} - \frac{1}{2}$ are binary coefficients for $j = 1, 2, \ldots, n$, and $\widetilde{\lambda}_0 = \lambda_0 + \frac{3q}{4} + \frac{3}{2} \sum_{k=1}^{n} \lambda_k$.*

*Proof.* Consider the function $f$ in the following form

$$f(x) = \sum_{j=1}^{n} \lambda_j x_j + \lambda_0 = \sum_{j=1}^{n} \left(\frac{q}{4} + \frac{q}{2} b_j\right) x_j + \lambda_0 = \frac{q}{4}\mathrm{wt}(x) + \frac{q}{2}\left(\bigoplus_{j=1}^{n} b_j x_j\right) + \lambda_0,$$

where $b_j \in \mathbb{F}_2$ for $j = 1, 2, \ldots, n$. From Lemma 1 it follows that for any $x \in \mathbb{F}_2^n$ it holds

$$\mathrm{wt}(x) = \bigoplus_{j=1}^{n} x_j - \sum_{k=2}^{n} (-1)^{k-1} 2^{k-1} \sum_{1 \leqslant r_1 < r_2 < \ldots < r_k \leqslant n} x_{r_1} x_{r_2} \cdot \ldots \cdot x_{r_k}.$$

In the sum above all the terms with $k \geqslant 3$, being multiplied by $q/4$, are vanished modulo $q$. It means that only the term with $k = 2$ makes sense, that is

$$\frac{q}{4}\mathrm{wt}(x) \equiv \bigoplus_{j=1}^{n} x_j - 2 \sum_{1 \leqslant r < s \leqslant n} x_r x_s \pmod{q}.$$

Then we have

$$\begin{aligned} f(x) &= \lambda_0 + \frac{q}{4}\left(\bigoplus_{j=1}^{n} x_j - 2 \sum_{1 \leqslant r < s \leqslant n} x_r x_s\right) + \frac{q}{2}\left(\bigoplus_{j=1}^{n} b_j x_j\right) \\ &= \lambda_0 + \frac{q}{4} \bigoplus_{j=1}^{n} x_j + \frac{q}{2}\left(\bigoplus_{h=1}^{n} b_h x_h \oplus \bigoplus_{1 \leqslant r < s \leqslant n} x_r x_s\right). \end{aligned}$$

For the dual gbent function binary coefficients are equal to negation of $b_j \in \mathbb{F}_2$ for $j = 1, 2, \ldots, n$. Therefore,

$$\widetilde{f}(x) = \widetilde{\lambda}_0 + \frac{q}{4} \bigoplus_{j=1}^{n} x_j + \frac{q}{2}\left(\bigoplus_{l=1}^{n} x_l \oplus \bigoplus_{h=1}^{n} b_h x_h \oplus \bigoplus_{1 \leqslant r < s \leqslant n} x_r x_s\right),$$

where $\widetilde{\lambda}_0 = \lambda_0 + \frac{3q}{4} + \frac{3}{2} \sum_{k=1}^{n} \lambda_k$. □

So we can describe all component Boolean functions of the affine gbent function and its dual. We specify the case, when $q$ is a power of 2, divisible by 8.

**Theorem 3.** *Let $q = 2^k$, where $k \geqslant 3$. The components Boolean functions $a_0, a_1, \ldots, a_{n-1} \in \mathcal{F}_n$ of affine gbent function of the form (1) are equal*

*to*

$$\begin{aligned}
a_0(x) &= c_0,\\
a_1(x) &= c_1,\\
&\vdots\\
a_{k-3}(x) &= c_{k-3},\\
a_{k-2}(x) &= c_{k-2} \oplus \bigoplus_{j=1}^{n} x_j,\\
a_{k-1}(x) &= c_{k-1} \oplus \bigoplus_{h=1}^{n} b_h x_h \oplus \bigoplus_{1\leqslant r<s\leqslant n} x_r x_s,
\end{aligned}$$

*where* $x \in \mathbb{F}_2^n$ *and vector* $c$ *is the binary representation of the element* $\lambda$.

*The components Boolean functions* $b_0, b_1, \ldots, b_{n-1} \in \mathcal{F}_n$ *of the dual to affine gbent function of the form (1) are equal to*

$$\begin{aligned}
b_0(x) &= \widetilde{c}_0\\
b_1(x) &= \widetilde{c}_1\\
&\vdots\\
b_{k-3}(x) &= \widetilde{c}_{k-3}\\
b_{k-2}(x) &= \widetilde{c}_{k-2} \oplus \bigoplus_{j=1}^{n} x_j\\
b_{k-1}(x) &= \widetilde{c}_{k-1} \oplus \bigoplus_{l=1}^{n} x_l \oplus \bigoplus_{h=1}^{n} b_h x_h \oplus \bigoplus_{1\leqslant r<s\leqslant n} x_r x_s,
\end{aligned}$$

*where* $x \in \mathbb{F}_2^n$ *and vector* $\widetilde{c}$ *is the binary representation of the element* $\widetilde{\lambda}_0 = \lambda_0 + \frac{3q}{4}n + \frac{3}{2}\sum_{k=1}^{n} \lambda_k$.

Note that quadratic parts of $a_{k-1}, b_{k-1}$ are, so called, elementary symmetric functions of degree 2. Dillon in [10] pointed that they are bent functions.

### 3.4 The duality mapping is isometric on the set of affine gbent functions

It is well-known that the duality mapping, being defined on Boolean bent functions, is isometry, that is it preserves the Hamming distance between any pair of bent functions [4]. It has a great interest in a scope of bent functions since it is the only known isometric mapping of the set of bent functions that is not an element of its group of automorphisms.

Within the set of generalized bent functions the duality mapping is defined on gbent functions that are called regular. In what follows we assume that either $n$ is even number or $n$ is odd and $q \equiv 0 \pmod 8$.

**Theorem 4.** *Within the Lee distance the duality mapping is an isometry of the set of regular affine gbent functions.*

*Proof.* Combining Theorems 1 and 2, we are to choose two regular affine gbent functions in $n$ variables and analyze the Lee distance between them and the distance between their dual ones. Let these functions be

$$f(x) = \sum_{j=1}^{n} \lambda_j x_j + \lambda_0, \qquad g(x) = \sum_{j=1}^{n} \mu_j x_j + \mu_0,$$

where $x \in \mathbb{F}_2^n$, $\lambda_j, \mu_j \in \left\{\frac{q}{4}, \frac{3q}{4}\right\}$ for $j = 1, 2, \ldots, n$ and $\lambda_0, \mu_0 \in \mathbb{Z}_q$. Then $\mathrm{dist}_L(f, g) = \mathrm{wt}_L(\Delta)$, where

$$\Delta(x) = \sum_{j=1}^{n} (\lambda_j - \mu_j)\, x_j + (\lambda_0 - \mu_0)\,, \quad x \in \mathbb{F}_2^n.$$

The duals of the considered gbent functions are

$$\widetilde{f}(x) = \sum_{j=1}^{n} (q - \lambda_j)\, x_j + \left(\lambda_0 + \frac{3q}{4} n + \frac{3}{2} \sum_{k=1}^{n} \lambda_k\right), \quad x \in \mathbb{F}_2^n,$$
$$\widetilde{g}(x) = \sum_{j=1}^{n} (q - \mu_j)\, x_j + \left(\mu_0 + \frac{3q}{4} n + \frac{3}{2} \sum_{k=1}^{n} \mu_k\right), \quad x \in \mathbb{F}_2^n.$$

The Lee distance between them is $\mathrm{dist}_L\big(\widetilde{f}, \widetilde{g}\big) = \mathrm{wt}_L\big(\Delta'\big)$, where

$$\Delta'(x) = \sum_{j=1}^{n} (\lambda_j - \mu_j)\, x_j + (\lambda_0 - \mu_0) + \frac{3}{2} \sum_{k=1}^{n} (\lambda_k - \mu_k)\,, \quad x \in \mathbb{F}_2^n.$$

Let's analyze the difference $\Delta - \Delta'$ between distances in details. Denote

$$N_1 = \{j \geqslant 1 : \lambda_j - \mu_j = q/2\}\,,$$
$$N_2 = \{j \geqslant 1 : \lambda_j - \mu_j = -q/2\}\,,$$
$$N_3 = \{j \geqslant 1 : \lambda_j - \mu_j = 0\}\,.$$

It is clear that $N_1 + N_2 + N_3 = n$ and these three numbers cover all posssible variants of differences between the coefficients of $f$ and $g$. If $N_1 = N_2 = 0$,

we have $\mathrm{dist}_L(f,g) \in \{0, q\cdot 2^{n-1}\}$ and it holds $\mathrm{dist}_L(f,g) = \mathrm{dist}_L(\widetilde{f},\widetilde{g})$. If at least one of $N_1, N_2$ is non-zero, we have

$$\mathrm{dist}_L(f,g) = 0\cdot 2^{n-1} + \frac{q}{2}\cdot 2^{n-1} = q\cdot 2^{n-2}.$$

At the same time it holds

$$\mathrm{dist}_L(\widetilde{f},\widetilde{g}) = w_1\cdot 2^{n-1} + w_2\cdot 2^{n-1},$$

where

$$w_1 = \mathrm{wt}_L\left[\frac{3q}{4}(N_1 - N_2)\right], \qquad w_2 = \mathrm{wt}_L\left[\frac{q}{2} + \frac{3q}{4}(N_1 - N_2)\right].$$

The pair of Lee weights $(w_1, w_2)$ belongs to the following set

$$\left\{\left(0,\frac{q}{2}\right),\left(\frac{3q}{4},\frac{q}{4}\right),\left(\frac{q}{2},0\right),\left(\frac{q}{4},\frac{3q}{4}\right)\right\},$$

therefore we have $\mathrm{dist}_L(f,g) = \mathrm{dist}_L(\widetilde{f},\widetilde{g})$ since $\mathrm{wt}_L(\frac{3q}{4}) = \frac{q}{4}$. □

The problem if the duality mapping preserves the Lee distance for a pair of gbent functions of any possible degree, at least for the case $q = 2^k$ with $k \geqslant 2$, is still open.

## 4 Metrical regularity of the set of self-dual bent functions for quaternary case

Let $X \subseteq \mathbb{Z}_q^n$ be an arbitrary set and let $y \in \mathbb{Z}_q^n$ be an arbitrary vector. Define the *distance* (either Hamming or Lee) between $y$ and $X$ as $\mathrm{dist}(y,X) = \min\limits_{x\in X}\mathrm{dist}(y,x)$. The *maximal distance* from the set $X$ is

$$\mathrm{d}(X) = \max_{y\in\mathbb{Z}_q^n}\mathrm{dist}(y,X).$$

In coding theory this number is also known as the *covering radius* of the set $X$. A vector $z \in \mathbb{Z}_q^n$ is called *maximally distant* from the set $X$ if $\mathrm{dist}(z,X) = \mathrm{d}(X)$. The set of all maximally distant vectors from the set $X$ is called the *metrical complement* of the set $X$ and denoted by $\widehat{X}$. A set $X$ is said to be *metrically regular* if $\widehat{\widehat{X}} = X$. In order, a subset of Boolean functions is called *metrically regular* if the set of corresponding vectors of values is metrically regular [52].

In paper [21] it was proved that for $n \geqslant 4$ the metrical complement of the set $\mathcal{SB}_n^+$ is the set $\mathcal{SB}_n^-$ and vice versa. The case $n = 2$ was considered separately and it was shown that these sets are metrically regular for any positive even $n$.

Here we study the problem for the sets $\mathcal{SB}_n^{q,+}$ and $\mathcal{SB}_n^{q,-}$ with $q = 4$.

Further we will use the following notation. Let $I_n$ be the identity matrix of size $n$ and $H_n = H_1^{\otimes n}$ be the $n$-fold tensor product of the matrix $H_1$ with itself, where

$$H_1 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

It is known the Hadamard property of this matrix

$$H_n H_n^T = 2^n I_{2^n},$$

where $H_n^T$ is transpose of $H_n$ (it holds $H_n^T = H_n$ by symmetricity of $H_n$). Denote $\mathcal{H}_n = 2^{-n/2} H_n$.

**Theorem 5.** *Let $n \geqslant 4$, then the following statements hold:*

- *the metrical complement of the set of quaternary self-dual bent functions coincides with the set of quaternary anti-self-dual bent functions;*
- *the metrical complement of the set of quaternary anti-self-dual bent functions coincides with the set of quaternary self-dual bent functions.*

*In both assertions the Lee distance is considered.*

*Proof.* It is known the following obvious relation for Lee distance between quaternary functions $f, g$ and squared Euclidian distance between their polyphase vectors $F = i^f$ and $G = i^g$.

$$\left(\mathrm{dist}_E\left(F, G\right)\right)^2 = 2 \cdot \mathrm{dist}_L(f, g) = 2^{n+1} - 2\mathrm{Re}\left\langle F, G\right\rangle,$$

therefore, $\mathrm{dist}_L(f, g) = 2^n - \mathrm{Re}\left\langle F, G\right\rangle$.

The negation $f + 2$ of any (anti-)self-dual quaternary bent function $f$ in $n$ variables is again (anti-)self-dual hence the covering radii of $\mathcal{SB}_n^{4,+}$ and $\mathcal{SB}_n^{4,-}$ are at most $2^n$. Since for any pair $f \in \mathcal{GB}_n^{4,+}$ and $g \in \mathcal{SB}_n^{4,-}$ their polyphase vectors, $F$ and $G$ correspondingly, are orthogonal vectors from the space $\mathbb{C}^{2^n}$, the inner product $\langle F, G\rangle$ is zero and, consequently, $\mathrm{d}\big(\mathcal{SB}_n^{4,+}\big) = \mathrm{d}\big(\mathcal{SB}_n^{4,-}\big) = 2^n$. Moreover, we have inclusions

$$\mathcal{SB}_n^{4,-} \subseteq \widehat{\mathcal{SB}}_n^{4,+} \qquad\qquad \mathcal{SB}_n^{4,+} \subseteq \widehat{\mathcal{SB}}_n^{4,-}$$

Recall that the kernel $\mathrm{Ker}(A)$ of the linear operator $A$ (in a fixed basis we see it as matrix) is such $x \in \mathbb{C}^{2^n}$ that $Ax = \mathbf{0}$. As in the proof for the Boolean case, see Theorem 3 in [21], consider the orthogonal decomposition

$$\mathbb{C}^{2^n} = \mathrm{Ker}\left(H_n + 2^{n/2} I_{2^n}\right) \oplus \mathrm{Ker}\left(H_n - 2^{n/2} I_{2^n}\right),$$

from which it follows that if polyphase vector of some quaternary function is orthogonal to one of these eigenspaces, it immediately belongs to another one.

Consider the basis of the subspace $\mathrm{Ker}\,(\mathcal{H}_n - I_{2^{n-1}})$ that consists of real valued vectors, that correspond to the set of quaternary self-dual bent functions $\{2f_i\}_{i=1}^{2^{n-1}}$, where $\{f_i\}_{i=1}^{2^n} \subseteq \mathcal{SB}_n^+$ was constructed in [21]. Assume that $F = A + Bi$ with $A, B \in \{0, \pm 1\}^{2^n}$, is a polyphase vector of some quaternary function at the Lee distance $2^n$ from the set $\mathcal{SB}_n^{4,+}$. Then $A$ is a vector from the subspace $\mathrm{Ker}\,(\mathcal{H}_n - I_{2^n})$ since the real part of the inner product of $F$ with every element of the aforementioned basis of $\mathrm{Ker}\,(\mathcal{H}_n - I_{2^n})$ is zero. One can show that $B$ is also a vector from the subspace $\mathrm{Ker}\,(\mathcal{H}_n - I_{2^n})$. For that we consider the basis which consists of real valued vectors, that correspond to the set of quaternary self-dual bent functions $\{2f_i + 1\}_{i=1}^{2^n}$, where again $\{f_i\}_{i=1}^{2^n} \subseteq \mathcal{SB}_n^+$. Thus, both $A$ and $B$ are vectors from the subspace $\mathrm{Ker}\,(\mathcal{H}_n - I_{2^n})$, therefore vector $F = A + Bi$, which is in fact their linear combination, is an element of $\mathrm{Ker}\,(\mathcal{H}_n - I_{2^n})$ as well. As a result we have the inclusion $\widehat{\mathcal{SB}}_n^{4,+} \subseteq \mathcal{SB}_n^{4,-}$.

By the same arguments one can show that $\widehat{\mathcal{SB}}_n^{4,-} \subseteq \mathcal{SB}_n^{4,+}$. □

Combining Theorem 3 from [21] and Theorem 5 we obtain

**Theorem 6.** *Let $n \geqslant 4$ and $q \in \{2, 4\}$, then the following statements hold:*

- *the metrical complement of the set $\mathcal{SB}_n^{q,+}$ coincides with the set $\mathcal{SB}_n^{q,-}$;*
- *the metrical complement of the set $\mathcal{SB}_n^{q,-}$ coincides with the set $\mathcal{SB}_n^{q,+}$.*

*In both assertions the Lee distance is considered.*

The case $n = 2$ for quaternary case was checked computationally and by Theorem 4 from [21] we have the following

**Theorem 7.** *Let $n \geqslant 2$ be an even number and $q \in \{2, 4\}$, then the sets $\mathcal{SB}_n^{q,+}$ and $\mathcal{SB}_n^{q,-}$ are metrically regular in the Lee distance with covering radii $q \cdot 2^{n-2}$.*

It follows that the sets of quaternary self-dual and anti-self-dual bent functions in even number of variables provide a non-trivial pair of quaternary metrically regular sets that are metrical complements of each other.

# 5 Decomposition of the form $(f_0, f_1, f_2, f_3)$ for general case

In this section the form and properties of the Gram matrix of an *arbitrary* bent function are considered. We generalize main results concerning Gram matrices for the self-dual case that were obtained in [23].

It is interesting to study the conditions when the subfunctions $f_0, f_1, f_2, f_3$ of the function itself and its dual are bent simultaneously. In current section we give an example of the property of the initial bent function that provides such double bentness.

Subfunctions of a bent function, comprising the restriction of a bent function on all subspaces of codimension 2, were studied in works [2, 3]. The considered sets of subfunctions were referred to as 4-*decompositions* of a bent function. In particular, it was shown that such subfunctions of a bent function in $n$ variables have the same Walsh–Hadamard spectrum: either all of them are bent, all are the three valued almost optimal (these are precisely near-bent functions with the spectrum having three values $0, \pm 2^{n/2}$), or they have the same Walsh–Hadamard spectrum with five values $0, \pm 2^{(n-2)/2}, \pm 2^{n/2}$.

Throughout this section given a function $f$ in $n$ variables we will refer to four Boolean functions $f_i$, $i = 0, 1, 2, 3$, in $n-2$ variables as to its subfunctions obtained by fixing the first and the second coordinates of the argument with the values $\{(00), (01), (10), (11)\}$, correspondingly. In order, vector of values of $f$ will have the form $(f_0, f_1, f_2, f_3)$. The sign vector of $f_i$ will be denoted by $F_i$, $i = 0, 1, 2, 3$. Let the notation $\mathcal{H}$ states for $\mathcal{H}_{n-2}$.

Further we will use the following observation. Let $f$ be a bent function in $n$ variables, then

$$\frac{1}{2}\begin{pmatrix} \mathcal{H} & \mathcal{H} & \mathcal{H} & \mathcal{H} \\ \mathcal{H} & -\mathcal{H} & \mathcal{H} & -\mathcal{H} \\ \mathcal{H} & \mathcal{H} & -\mathcal{H} & -\mathcal{H} \\ \mathcal{H} & -\mathcal{H} & -\mathcal{H} & \mathcal{H} \end{pmatrix} \begin{pmatrix} F_0 \\ F_1 \\ F_2 \\ F_3 \end{pmatrix} = \begin{pmatrix} R_0 \\ R_1 \\ R_2 \\ R_3 \end{pmatrix}$$

or, equivalently,

$$\begin{cases} F_0 + F_1 + F_2 + F_3 = 2\mathcal{H}R_0, \\ F_0 - F_1 + F_2 - F_3 = 2\mathcal{H}R_1, \\ F_0 + F_1 - F_2 - F_3 = 2\mathcal{H}R_2, \\ F_0 - F_1 - F_2 + F_3 = 2\mathcal{H}R_3, \end{cases} \tag{2}$$

where $R_i$, $i = 0, 1, 2, 3$, are sign vectors of subfunctions of $\widetilde{f}$. Obviously, the function $f$ is self-dual if and only if $R_i = F_i$, $i = 0, 1, 2, 3$.

### 5.1 Concatenation of four bent functions

The case when all four subfunctions are bent essentially leads to the idea of an iterative construction of a bent function in $n+2$ variables through four bent functions in $n$ variables. In [37] Preneel et al. proved that given four bent functions $f_i$, $i = 0, 1, 2, 3$, in $n$ variables, the concatenation of vectors of values of $f_i$ yields a bent function in $n+2$ variables if and only if

$$W_{f_0}(y)W_{f_1}(y)W_{f_2}(y)W_{f_3}(y) = -2^{2n} \text{ for any } y \in \mathbb{F}_2^n.$$

In terms of duals this condition is equivalent to the following

$$\widetilde{f}_0(y) \oplus \widetilde{f}_1(y) \oplus \widetilde{f}_2(y) \oplus \widetilde{f}_3(y) = 1 \text{ for any } y \in \mathbb{F}_2^n.$$

Bent functions in $n+2$ variables obtained by the concatenation of four bent functions in $n$ variables were also studied in [51] from the point of view of obtaining lower bounds on the cardinality of the set of bent functions. Such functions were referred to as *bent iterative* functions. Concatenation constructions were also considered in recent papers [16, 1].

### 5.2 General form of the Gram matrix

In this subsection we will study the Gram matrix of vectors $F_i$, $i = 0, 1, 2, 3$ which are sign vectors of subfunctions of a Boolean function $f$. Recall that elements $g_{ij}$ of the Gram matrix of vectors $\{v_k\}_{k \in M} \subset \mathbb{R}^d$ are inner products between $v_i$ and $v_j$, $i, j \in M$. The determinant of the Gram matrix is called the Gramian of the corresponding system of vectors. The basic properties of real Gram matrices are:

- symmetricity;
- positive semi-definiteness;
- the Gramian is zero if and only if the vectors are linearly dependent.

Denote the inner products by $g_{ij} = \langle F_i, F_j \rangle$, $i, j = 0, 1, 2, 3$.

The form of the Gram matrix of a bent function and its dual one is characterized by the following

**Theorem 8.** *The Gram matrices of any bent function, say $f$, in $n$ variables*

*and its dual bent function* $\widetilde{f}$ *have form*

$$\mathrm{Gram}(f) = \begin{pmatrix} 2^{n-2} & b & c & -a \\ b & 2^{n-2} & a & -c \\ c & a & 2^{n-2} & -b \\ -a & -c & -b & 2^{n-2} \end{pmatrix},$$

$$\mathrm{Gram}(\widetilde{f}) = \begin{pmatrix} 2^{n-2} & c & b & -a \\ c & 2^{n-2} & a & -b \\ b & a & 2^{n-2} & -c \\ -a & -b & -c & 2^{n-2} \end{pmatrix}$$

*for some even integers* $a, b, c$ *such that*

$$-2^{n-2} + |b + c| \leqslant a \leqslant 2^{n-2} - |b - c|.$$

*Proof.* For self-dual case the system (2) has form

$$\begin{cases} F_0 + F_1 + F_2 + F_3 = 2\mathcal{H}R_0, \\ F_0 - F_1 + F_2 - F_3 = 2\mathcal{H}R_1, \\ F_0 + F_1 - F_2 - F_3 = 2\mathcal{H}R_2, \\ F_0 - F_1 - F_2 + F_3 = 2\mathcal{H}R_3. \end{cases} \tag{3}$$

Consider inner products of right parts of all equations in the system (3) with themselves. The symmetricity of $\mathrm{Gram}(f)$ implies that we have at most six different coefficients outside the main diagonal in fact. For example, the 1st equation's expression inner product with itself is

$$\begin{aligned} \langle 2\mathcal{H}R_0, 2\mathcal{H}R_0 \rangle &= \langle F_0 + F_1 + F_2 + F_3, F_0 + F_1 + F_2 + F_3 \rangle \\ &= \sum_{i,j=0}^{3} g_{ij} = 4 \cdot 2^{n-2} + \sum_{\substack{i,j=0, \\ i \neq j}}^{3} g_{ij} = 2^n. \end{aligned}$$

It yields the following equation on the coefficients:

$$g_{01} + g_{02} + g_{03} + g_{12} + g_{13} + g_{23} = 0.$$

Finally, after considering the rest ones, we have the following system of equations that describe necessary relations betweeen the entries of the Gram matrix:

$$\begin{cases} g_{01} + g_{02} + g_{03} + g_{12} + g_{13} + g_{23} = 0, \\ g_{01} - g_{02} + g_{03} + g_{12} - g_{13} + g_{23} = 0, \\ g_{01} - g_{02} - g_{03} - g_{12} - g_{13} + g_{23} = 0, \\ g_{01} + g_{02} - g_{03} - g_{12} + g_{13} + g_{23} = 0. \end{cases}$$

This system has rank 3, its general solution is

$$\begin{aligned} &g_{01} = -g_{23}, && g_{02} = -g_{13}, && g_{03} = -g_{12}, \\ & && g_{12} = g_{12}, && g_{13} = g_{13}, \\ & && && g_{23} = g_{23} \end{aligned}$$

for $g_{12}$, $g_{13}$ and $g_{23}$ being free variables. Denote $a = g_{12}$, $b = -g_{23}$ and $c = -g_{13}$, then we obtain the desired form of the Gram matrix:

$$\begin{pmatrix} 2^{n-2} & b & c & -a \\ b & 2^{n-2} & a & -c \\ c & a & 2^{n-2} & -b \\ -a & -c & -b & 2^{n-2} \end{pmatrix}.$$

We can also deduce that the Gram matrix of the dual function is strictly connected with the matrix of the function itself. Since the dual function $\widetilde{f}$ is bent as well, it is enough to investigate the first row of its Gram matrix. We have

$$\begin{aligned} \langle 2\mathcal{H}R_0, 2\mathcal{H}R_1\rangle &= \langle F_0 + F_1 + F_2 + F_3, F_0 - F_1 + F_2 - F_3\rangle \\ &= 2g_{02} - 2g_{13} = 2c - 2(-c) = 4c, \\ \langle 2\mathcal{H}R_0, 2\mathcal{H}R_2\rangle &= \langle F_0 + F_1 + F_2 + F_3, F_0 + F_1 - F_2 - F_3\rangle \\ &= 2g_{01} - 2g_{23} = 2b - 2(-b) = 4b, \\ \langle 2\mathcal{H}R_0, 2\mathcal{H}R_3\rangle &= \langle F_0 + F_1 + F_2 + F_3, F_0 - F_1 - F_2 + F_3\rangle \\ &= 2g_{03} - 2g_{12} = 2(-a) - 2a = -4a, \end{aligned}$$

hence $\langle R_0, R_1\rangle = c$, $\langle R_0, R_2\rangle = b$ and $\langle R_0, R_3\rangle = -a$.

Now we are to point essential bounds on values of $a$, $b$ and $c$ and deduce some relations between them. In order to do it recall that any Gram matrix is positive semi-definite, hence all its eigenvalues must be nonnegative. The matrix $\mathrm{Gram}(f)$ has four eigenvalues, they are

$$\begin{aligned} \lambda_1 &= 2^{n-2} - a + b - c, \\ \lambda_2 &= 2^{n-2} - a - b + c, \\ \lambda_3 &= 2^{n-2} + a - b - c, \\ \lambda_4 &= 2^{n-2} + a + b + c. \end{aligned}$$

One can note that these numbers are nonnegative if and only if

$$\begin{aligned} a &\leqslant 2^{n-2} \pm (b - c), \\ a &\geqslant -2^{n-2} \pm (b + c), \end{aligned}$$

that is

$$a \leqslant 2^{n-2} + \min\{b-c, c-b\},$$
$$a \geqslant -2^{n-2} + \max\{b+c, -b-c\},$$

and, consequently,

$$a \leqslant 2^{n-2} - |b-c|,$$
$$a \geqslant -2^{n-2} + |b+c|,$$

where $|b|$ and $|c|$ are essentially bounded by $2^{n-2}$ from above. Parity of the numbers $a, b, c$ comes from the fact that they are are inner products of integer vectors of an even dimension having odd coordinates. □

Thus, the duality mapping acts on the Gram matrix by switching values of the coefficients $b$ and $c$.

Theorem 8 can be reformulated in terms of Hamming distances between subfunctions:

**Corollary 3.** *Let $f$ be a bent function in $n$ variables. The distances between $\{f_i\}_{i=0}^3$ are characterized by the matrix*

$$\mathrm{Dist}(f) = \begin{pmatrix} 0 & 0 & 0 & 2^{n-2} \\ 0 & 0 & 0 & 2^{n-2} \\ 0 & 0 & 0 & 2^{n-2} \\ 2^{n-2} & 2^{n-2} & 2^{n-2} & 0 \end{pmatrix} + 2 \begin{pmatrix} 0 & d_2 & d_3 & -d_1 \\ d_2 & 0 & d_1 & -d_3 \\ d_3 & d_1 & 0 & -d_2 \\ -d_1 & -d_3 & -d_2 & 0 \end{pmatrix}$$

*for some positive integers $d_1, d_2, d_3$ such that*

$$|d_2 - d_3| \leqslant d_1 \leqslant 2^{n-2} - \left|2^{n-2} - d_2 - d_3\right|,$$

$$\left|2^{n-2} - 2 \cdot \min(d_2, d_3)\right| \leqslant d_1 + \left|2^{n-2} - d_2 - d_3\right|.$$

*Proof.* The relation between the inner product and the Hamming distance yields the matrix whereas the inequalities are obtained from

$$-2^{n-2} + |b+c| \leqslant a \leqslant 2^{n-2} - |b-c|$$

with

$$a = 2^{n-2} - 2d_1, \qquad b = 2^{n-2} - 2d_2, \qquad c = 2^{n-2} - 2d_3.$$

□

### 5.3 Linear dependence and bentness

In this subsection we are to consider the connection between singularity of the Gram matrix of an arbitrary bent function and bentness of its subfunctions.

We will need the following

**Lemma 2.** *Let $f$ and $g$ be Boolean functions in even number $k$ of variables, such that $W_f(y), W_g(y) \in \left\{0, \pm 2^{k/2}, \pm 2^{(k+2)/2}\right\}$ for any $y \in \mathbb{F}_2^k$ and*

$$\sum_{x,y \in \mathbb{F}_2^k} (-1)^{f(x) \oplus g(y) \oplus \langle x,y \rangle} = 2^{3k/2}. \tag{4}$$

*Then $f$ and $g$ are bent functions, moreover, it holds $g = \widetilde{f}$.*

*Proof.* Consider five nonnegative integers

$$\begin{aligned}
t_0 &= \left|\left\{y \in \mathbb{F}_2^k : W_f(y) = 0\right\}\right|, \\
t_1 &= \left|\left\{y \in \mathbb{F}_2^k : (-1)^{g(y)} W_f(y) = 2^{k/2}\right\}\right|, \\
t_2 &= \left|\left\{y \in \mathbb{F}_2^k : (-1)^{g(y)} W_f(y) = -2^{k/2}\right\}\right|, \\
t_3 &= \left|\left\{y \in \mathbb{F}_2^k : (-1)^{g(y)} W_f(y) = 2^{(k+2)/2}\right\}\right|, \\
t_4 &= \left|\left\{y \in \mathbb{F}_2^k : (-1)^{g(y)} W_f(y) = -2^{(k+2)/2}\right\}\right|.
\end{aligned}$$

Then we have following system

$$\begin{cases}
t_0 + t_1 + t_2 + t_3 + t_4 = 2^k, \\
(t_1 + t_2)\, 2^k + (t_3 + t_4)\, 2^{k+2} = 2^{2k}, \\
(t_1 - t_2)\, 2^{k/2} + (t_3 - t_4)\, 2^{(k+2)/2} = 2^{3k/2},
\end{cases}$$

where the second equation follows from the Parseval's identity applied for the function $g$, and the third one is the product (4). The only nonnegative solution is

$$t_0 = 0, \qquad t_1 = 2^k, \qquad t_2 = 0, \qquad t_3 = 0, \qquad t_4 = 0.$$

Hence, we have $|W_f(y)| = 2^{k/2}$ for any $y \in \mathbb{F}_2^k$.

By the same arguments one can show that $|W_g(y)| = 2^{k/2}$ for any $y \in \mathbb{F}_2^k$, therefore both of $f$ and $g$ are bent functions. Finally, it is enough to note that in this case the product (4) is exactly

$$\sum_{x,y \in \mathbb{F}_2^k} (-1)^{f(x) \oplus g(y) \oplus \langle x,y \rangle} = 2^{k/2} \sum_{y \in \mathbb{F}_2^k} (-1)^{\widetilde{f}(y) \oplus g(y)} = 2^{3k/2},$$

that is $\widetilde{f} = g$. □

For a bent function $f$ with the Gram matrix $\mathrm{Gram}(f)$ the Gramian has the following expression

$$\begin{aligned}\mathrm{Gramian}(f) = &\left(2^{n-2} - a + b - c\right)\left(2^{n-2} - a - b + c\right)\left(2^{n-2} + a - b - c\right)\\ &\times \left(2^{n-2} + a + b + c\right).\end{aligned} \tag{5}$$

In [23] it was proved that the singularity of the Gram matrix of a self-dual bent function implies bentness of its subfunctions. It appears that this fact holds for *any* bent function as well.

**Theorem 9.** *If the Gram matrix of a bent function $f$ is singular, then subfunctions $\{f_i\}_{i=0}^{3}$ are bent. Moreover, the subfunctions of $\widetilde{f}$ are also bent.*

*Proof.* It is enough to consider all such combinations of $a, b, c$ that the Gramian (5) is zero. We will consider all the cases separately.

$2^{n-2} - a + b - c = 0$: in terms of sign vectors it is equivalent to the following

$$2^{n-2} + \langle F_0, F_1 - F_2 + F_3\rangle = 0.$$

From system (2) it follows that

$$\begin{cases} F_0 + (-F_1 + F_2 - F_3) = 2\mathcal{H}R_1,\\ \langle F_0, -F_1 + F_2 - F_3\rangle = 2^{n-2}, \end{cases}$$

that after simple transformations becomes

$$\begin{cases} -F_1 + F_2 - F_3 = 2\mathcal{H}R_1 - F_0,\\ \langle F_0, \mathcal{H}R_1\rangle = 2^{n-2}. \end{cases}$$

By Lemma 2 from the second equation we obtain that $F_0$ and $R_1$ are sign vectors of bent functions.

For other cases it is sufficient to list the systems, the rest of considerations are the same.

$2^{n-2} - a - b + c = 0$:

$$\begin{cases} F_1 - F_2 - F_3 = 2\mathcal{H}R_2 - F_0,\\ \langle F_0, \mathcal{H}R_2\rangle = 2^{n-2}; \end{cases}$$

$2^{n-2} + a - b - c = 0$:

$$\begin{cases} F_1 + F_2 + F_3 = 2\mathcal{H}R_0 - F_0,\\ \langle F_0, \mathcal{H}R_0\rangle = 2^{n-2}; \end{cases}$$

$2^{n-2} + a + b + c = 0$:

$$\begin{cases} -F_1 - F_2 + F_3 = 2\mathcal{H}R_3 - F_0, \\ \langle F_0, \mathcal{H}R_3 \rangle = 2^{n-2}. \end{cases}$$

□

Again, from Theorem 9 and properties of the Gram matrix we conclude that

**Corollary 4.** *If sign vectors of subfunctions $\{f_i\}_{i=0}^3$ of a bent function $f$ are linearly dependent, then these subfunctions are bent. Moreover, the subfunctions of $\widetilde{f}$ are bent as well.*

We can also consider $4 \times 4$ integer matrix, say $M$, with elements

$$m_{ij} = \langle F_i, \mathcal{H}R_j \rangle = \langle \mathcal{H}F_i, R_j \rangle .$$

In order to calculate its elements one can refer to the system (2) and consider inner products of equations with sign vectors of the subfunctions of $f$. The matrix $\frac{1}{2}M$ is the following

$$\begin{pmatrix} 2^{n-2} - a + b + c & 2^{n-2} + a - b + c & 2^{n-2} + a + b - c & 2^{n-2} - a - b - c \\ 2^{n-2} + a + b - c & -2^{n-2} + a + b + c & 2^{n-2} - a + b + c & -2^{n-2} - a + b - c \\ 2^{n-2} + a - b + c & 2^{n-2} - a + b + c & -2^{n-2} + a + b + c & -2^{n-2} - a - b + c \\ 2^{n-2} - a - b - c & -2^{n-2} - a - b + c & -2^{n-2} - a + b - c & 2^{n-2} - a + b + c \end{pmatrix}$$

It is clear that it is symmetric if and only if $b = c$. This matrix provides one more condition for bentness of subfunctions.

**Proposition 3.** *If matrix $M$ of a bent function $f$ is singular, then the subfunctions $\{f_i\}_{i=0}^3$ are bent. Moreover, the subfunctions of $\widetilde{f}$ are bent as well.*

*Proof.* It is enough to straightly calculate the eigenvalues of the matrix $M$ and consider cases when at least one of them is zero. The eigenvalues are

$$\begin{aligned} \mu_1 &= \sqrt{(2^{n-2} - a)^2 - (b - c)^2}, \\ \mu_2 &= -\sqrt{(2^{n-2} - a)^2 - (b - c)^2}, \\ \mu_3 &= -2^{n-2} - a + b + c, \\ \mu_4 &= 2^{n-2} + a + b + c. \end{aligned}$$

Note that $\mu_3 = -\lambda_3$ and $\mu_4 = \lambda_4$. The eigenvalues $\mu_{1,2}$ are zero if and only if $2^{n-2} - a = |b - c|$, but it implies that either $\lambda_1 = 0$ or $\lambda_2 = 0$. So, in all cases we have that at least one of the eigenvalues of the matrix $\mathrm{Gram}(f)$ is zero, hence from Theorem 9 the result follows. □

Thus, we obtain the properties of a matrix that consists of the inner products between sign vectors of subfunctions of a bent function and its dual. Whereas one of vectors is given in its Walsh–Hadamard transform.

# 6 Conclusion

In current work we study some classes of generalized bent functions, in particular, affine gbent functions and quaternary self-dual gbent functions. The metrical regularity of the last one was shown, but for the case $q = 2^k$ with $k \geqslant 3$ the question is still open. It might comprise the consideration of different metrics on the set of generalized Boolean functions. Self-dual gbent functions connot be affine but it is worth to study the upper bound for their algebraic degree.

**Acknowledgments.**

The work was carried out within the framework of the state contract of the Sobolev Institute of Mathematics (project no. FWNF-2022-0018).

# References

[1] Bapić A., Pasalic E., Zhang F., Hodžić S., "Constructing new superclasses of bent functions from known ones", *Cryptogr. Commun.*, **14**:6 (2022), 1229–1256.

[2] Canteaut A., Carlet C., Charpin P., Fontaine C., "On cryptographic properties of the cosets of $R(1, m)$", *IEEE Trans. Inform. Theory*, **47**:4 (2001), 1494–1513.

[3] Canteaut A., Charpin P., "Decomposing bent functions", *IEEE Trans. Inform. Theory*, **49**:8 (2003), 2004–2019.

[4] Carlet C., "Two new classes of bent functions", *LNCS, volume 765*, Advances in Cryptology — EUROCRYPT '93, 1994, 77–101.

[5] Carlet C., Danielsen L.E., Parker M.G., Solé. P., "Self-dual bent functions", *Int. J. Inform. Coding Theory*, **1** (2010), 384–399.

[6] Carlet C., Mesnager S., "Four decades of research on bent functions", *Des. Codes Cryptogr.*, **78**:1 (2016), 5–50.

[7] Çeşmelioğlu A., Meidl W., Pott A., "On the dual of (non)-weakly regular bent functions and self-dual bent functions over $\mathbb{Z}_4$", *Adv. Math. Commun.*, **7**:4 (2013), 425–440.

[8] Davis, J.A. and Jedwab, J., "Peak-to-mean power control in OFDM, Golay complementary sequences, and Reed–Muller codes", *IEEE Trans. Inform. Theory*, **45**:7 (1999), 2397–2417.

[9] Çeşmelioğlu A., Meidl W., Pott A., "Vectorial bent functions and their duals", *Linear Algebra Appl.*, **548** (2018), 305–320.

[10] Dillon J., "Elementary Hadamard Difference Sets", 1974, PhD dissertation.

[11] Feulner T., Sok L., Solé P., Wassermann A., "Towards the Classification of Self-Dual Bent Functions in Eight Variables", *Des. Codes Cryptogr.*, **68**:1 (2013), 395–406.

[12] Heng Z., Yue Q., "Optimal codebooks achieving the Levenshtein bound from generalized bent functions over $\mathbb{Z}_4$", *Cryptogr. Commun.*, **9**:1 (2017), 41–53.

[13] Hodžić S., Pasalic E., "Generalized bent functions — some general construction methods and related necessary and sufficient conditions", *Cryptogr. Commun.*, **7**:4 (2015), 469–483.

[14] Hodžić S., Pasalic E., "Construction methods for generalized bent functions", *Discrete Appl. Math.*, **238** (2018), 14–23.

[15] Hodžić S., Meidl W., Pasalic E., "Full characterization of generalized bent functions as (semi)-bent spaces, their dual, and the Gray image", *IEEE Trans. Inform. Theory*, **64**:7 (2018), 5432–5440.

[16] Hodžić S., Pasalic E., Wei Y., "A general framework for secondary constructions of bent and plateaued functions", *Des. Codes Cryptogr.*, **88**:10 (2020), 2007–2035.

[17] Hou X.-D., "Classification of self dual quadratic bent functions", *Des. Codes Cryptogr.*, **63**:2 (2012), 183–198.

[18] Hou X.-D., "Classification of $p$-ary self dual quadratic bent functions, $p$ odd", *J. Algebra*, **391** (2013), 62–81.

[19] Hyun J.Y., Lee H., Lee Y., "MacWilliams duality and Gleason-type theorem on self-dual bent functions", *Des. Codes Cryptogr.*, **63**:3 (2012), 295–304.

[20] Kumar P.V., Scholtz R.A., Welch L.R., "Generalized bent functions and their properties", *J. Comb. Theory Series A*, **40**:1 (1985), 90–107.

[21] Kutsenko A., "Metrical properties of self-dual bent functions", *Des. Codes Cryptogr.*, **88**:1 (2020), 201–222.

[22] Kutsenko A., Tokareva N., "Metrical properties of the set of bent functions in view of duality", *Applied Discrete Math.*, **49** (2020), 18–34.

[23] Kutsenko A, "On subfunctions of self-dual bent functions", Proceedings of the 11th Workshop on Current Trends in Cryptology (CTCrypt 2021), 2022, 210–239.

[24] Kuz'min A.S., Markov V.T., Nechaev A.A., Shishkin V.A., Shishkov A.B., "Bent and hyperbent functions over a field of $2^l$ elements", *Probl. Inf. Transm.*, **44**:1 (2008), 12–33.

[25] Leung K.H., Wang Q., "New nonexistence results on $(m, n)$-generalized bent functions", *Des. Codes Cryptogr.*, **88**:4 (2020), 755–770.

[26] Li Y., Kan H., Mesnager S., Peng J., Tan C.H., Zheng L., "Generic constructions of (Boolean and vectorial) bent functions and their consequences", *IEEE Trans. Inform. Theory*, **68**:4 (2022), 2735–2751.

[27] Liu H., Feng K., Feng R., "Nonexistence of generalized bent functions from $\mathbb{Z}_2^n$ to $\mathbb{Z}_m$", *Des. Codes Cryptogr.*, **82**:3 (2017), 647–662.

[28] Logachev O.A., Sal'nikov A.A., Yashchenko V.V., "Bent functions on a finite Abelian group", *Discrete Math. Appl.*, **7**:6 (1997), 547–564.

[29] Luo G., Cao X., Mesnager S., "Several new classes of self-dual bent functions derived from involutions", *Cryptogr. Commun.*, **11**:6 (2019).

[30] Martinsen T., Meidl W., Stănică P., "Generalized bent functions and their Gray images", *LNSC, volume 10064*, WAIFI 2016: Arithmetic of Finite Fields, 2017, 160–173.

[31] Matsufuji S., Imamura K., "Balanced quadriphase sequences with optimal periodic correlation properties constructed by real-valued bent functions", *IEEE Trans. Inform. Theory*, **39**:1 (1993), 305–310.

[32] Mesnager S., "Several new infinite families of bent Functions and their duals", *IEEE Trans. Inf. Theory*, **60**:7 (2014), 4397–4407.

[33] Mesnager S., *Bent functions: fundamentals and results*, Springer, Berlin, 2016, 544 p.

[34] Mesnager S., Tang C., Qi Y., Wang L., Wu B., Feng K., "Further results on generalized bent functions and their complete characterization", *IEEE Trans. Inform. Theory*, **64**:7 (2018), 5441–5452.

[35] Mesnager S., Riera C., Stănică P., "Multiple characters transforms and generalized Boolean functions", *Cryptogr. Commun.*, **11**:6 (2019), 1247–1260.

[36] Paterson K.G., "Generalized Reed–Muller codes and power control in OFDM Modulation", *IEEE Trans. Inform. Theory*, **46**:1 (2000), 104–120.

[37] Preneel B., Van Leekwijck W., Van Linden L., Govaerts R., Vandewalle J., "Propagation characteristics of Boolean functions", *LNCS, volume 473*, Advances in Cryptology-EUROCRYPT, **473**, Berlin, Heidelberg, 1990, 161–173.

[38] Riera C., Stănică P., Gangopadhyay S., "Generalized bent Boolean functions and strongly regular Cayley graphs", *Discrete Appl. Math.*, **283** (2020), 367–374.

[39] Rothaus O.S., "On "bent" functions", *J. Combin. Theory. Ser. A*, **20**:3 (1976), 300–305.

[40] Schmidt K.-U., "Complementary sets, generalized Reed–Muller codes, and power control for OFDM", *IEEE Trans. Inform. Theory*, **53**:2 (2007), 808–814.

[41] Schmidt K.-U., "Quaternary constant-amplitude codes for multicode CDMA", *IEEE Trans. Inform. Theory*, **55**:4 (2009), 1824–1832.

[42] Shi M., Li Y., Cheng W., Crnković D., Krotov D., Solé P., "Self-dual bent sequences for complex Hadamard matrices", *Des. Codes Cryptogr.*, 2022.

[43] Singh B.K., "On cross-correlation spectrum of generalized bent functions in generalized Maiorana–McFarland class", *Inf. Sci. Lett.*, **2**:3 (2013), 139–145.

[44] Sok L., Shi M., Solé P., "Classification and construction of quaternary self-dual bent functions", *Cryptogr. Commun.*, **10**:2 (2018), 277–289.

[45] Solé P., Tokareva N., "Connections between quaternary and binary bent functions", 2009, https://eprint.iacr.org/2009/544.pdf.

[46] Solodovnikov V.I., "Bent functions from a finite Abelian group into a finite Abelian group", *Discret. Math. Appl.*, **12**:2 (2002), 111–126.

[47] Stănică P., Martinsen T., Gangopadhyay S., Singh B.K., "Bent and generalized bent Boolean functions", *Des. Codes Cryptogr.*, **69**:1 (2013), 77–94.

[48] Su S., Guo X., "A further study on the construction methods of bent functions and self-dual bent functions based on Rothaus's bent function", *Des. Codes Cryptogr.*, 2022.

[49] Tang C., Xiang C., Qi Y., Feng K., "Complete characterization of generalized bent and $2^k$-bent Boolean functions", *IEEE Trans. Inform. Theory*, **63**:7 (2017), 4668–4674.

[50] Tokareva N.N., "Generalizations of bent functions — a survey", *J. Appl. Ind. Math.*, **5**:1 (2011), 110–129.

[51] Tokareva N., "On the number of bent functions from iterative constructions: lower bounds", *Adv. Math. Commun.*, **5**:4 (2011), 609–621.

[52] Tokareva N., *Bent functions, results and applications to cryptography*, Acad. Press. Elsevier, 2015, 230 p.

[53] Zhou J., Xu Y., Wang L., Li N., "Nearly optimal codebooks from generalized Boolean bent functions over $\mathbb{Z}_4$", *Adv. Math. Commun.*, **16**:3 (2022), 485–501.

# Reliability of two-level testing approach of the NIST SP800-22 test suite and two-sided estimates for quantile of binomial distribution

Aleksandr Serov

Steklov Mathematical Institute of Russian Academy of Sciences, Russia
serov@mi-ras.ru

**Abstract**

The two-level approach for testing RNGs involving the well known NIST SP 800-22 test suite, i.e., counting the sequences passing a basic test and checking the $p$-values distribution with a chi-square test, was considered. Such approach may increase the reliability of the test. However it is sensitive to the approximation error introduced by the computing of $p$-values. In this paper it is shown that for AES-based sequences two-level testing approach is not reliable too. Systematic error in the computing of the $p$-values is dependent only on the accuracy of approximation of the exact distribution of statistic by its theoretical counterpart and the number of bits in the analyzed sequences $n$. For a reliable second-level test, this error should be smaller, or at least, approximately equal to $\sigma/N = \frac{1}{k}\sqrt{\frac{k-1}{N}}$, where $\sigma = \sqrt{\frac{1}{k}\left(1-\frac{1}{k}\right)N}$ is the standard deviation from the mean number of particles in a bin for equiprobable scheme of allocation $N$ particles in $k$ bins. Such heuristic assumptions and carried out experiments suggest that for example in the second-level test of the Frequency test of NIST SP 800-22 test suite with $n = 2^{20}$ the number of tested sequences $N$ should not exceed 26184.

To completely eliminate the systematic error appearing in the Frequency test when determining the number of bin from $k$ bins (disjoint sub-intervals of $[0, 1]$) to which the $p$-value belongs the two-sided estimates of the quantiles of the binomial law are proposed.

**Keywords:** random sequences, pseudorandom sequences, statistical testing, reliability of statistical test, binomial distribution, two-sided estimates

## 1 Introduction

Random sequences are used in a large variety of areas, such as quantum mechanics, game theory, statistics, cryptography and so on. These sequences may be generated either by physical sources or deterministic algorithms. Random Number Generators (RNGs) represent a fundamental component in many applications, they are essential for cryptographic systems (see, for example, [7]). The security of many cryptographic schemes and protocols is

based on the perfect randomness of RNG outputs. RNGs are classified into two types: pseudo/deterministic and true/non-deterministic random number generators (PRNGs and TRNGs, respectively). In general, TRNGs based on some random physical phenomenons, may be used directly as random bit sources or generate seeds for PRNGs, and PRNGs extend the seeds to produce deterministic long sequences.

For any type of RNG statistical hypothesis tests have been widely employed to assess the quality of the RNG, which evaluate whether the output sequences conform with the given null hypothesis (e. g., the elements of the sequence independent and uniformly distributed) or not. In addition, statistical randomness tests are also used to evaluate the outputs of other cryptographic primitives such as block ciphers and hash functions, to preliminarily validate the indistinguishability of their outputs from a uniform random permutation or a equiprobable random mapping. The quality checking of binary sequences usually is based on some well-known batteries of tests, each of which is composed of a serial of tests, include Diehard [6] proposed by Marsaglia, SP 800-22 [14] standardized by US National Institute of Standard and Technology (NIST) or a software library TestU01 [19].

Randomness testing of cryptographic algorithms are of crucial importance to both designer and the attacker. Note that a test for randomness may be interpreted only in a probabilistic way. Looking at a sequence of all 0s, we say that the sequence is not random at all, though for an ideal RNG this sequence has the same probability of any other sequence of the same length; on the contrary, a RNG which is not able to generate this sequence is not ideal.

The main problem of the statistical testing is the interpretation of the results. Roughly speaking, while a failed test is a serious indicator for the weakness of a RNG, a passed test does not provide a direct positive proof for the quality of a RNG. From a mathematical point of view a test may be considered as a function of a sequence of $n$ elements (e.g., a sequence of $n$ bits) with output value in $[0, 1]$, called a $p$-value. In null-hypothesis significance testing, the $p$-value is the probability of obtaining test results at least as extreme as the result actually observed, under the assumption that the null hypothesis is correct.

In general case, if the null hypothesis is true, the $p$-value based on a continuous test statistic has a uniform distribution over the interval $[0, 1]$, regardless of the sample size of the experiment. In contrast, the distribution of the $p$-value under the alternative hypothesis is a function of both sample size and the true value or range of true values of the tested parameter of the

true distribution. In the discrete case, then the test statistic distribution is not continuous and the null hypothesis is true, i.e. all elements of the input sequence are independent and drawn according to the uniform distribution, the $p$-value is approximately uniformly distributed in the interval $[0, 1]$, and its cumulative distribution function (CDF) $F_p(x)$ is approximately equal to $x$ ($F_p(x) \approx x$). If the elements of the generated sequence is non-independent or not distributed according to the uniform distribution, then the CDF of $p$-value is not known as a rule.

## 2 NIST SP 800-22 test suite

In this paper the most commonly used statistical test suite, the SP 800-22 test suite from US National Institute of Standard and Technology (NIST) [14], is considered. It is well known that the NIST statistical test suite was used for the evaluation of AES candidate algorithms. This statistical test suite is build for analyzing the randomness properties of sequences and generators, is composed of 15 tests, and provides comprehensive evaluation for different randomness aspects of assessed sequences. All NIST SP 800-22 tests are listed in Table 1.

The main reason for using the suite, from an engineering point of view, is that it has several appealing properties. First, it is composed of several different tests, each of them is applied to the same input sequence of $n$ bits (for many of the tests the assumption has been made that the sequence length $n$ is a value from $10^3$ to $10^7$), searching for a specific statistical feature and expressing it as numerical quantity of $p$-value[1]. Second, the suite is composed by a number of well known tests and, for all of them, an exhaustive mathematical treatment is available. The source code of all the tests in the suite is public available.

The purpose of this paper is to consider the testing strategy proposed in Section 4 of the NIST publication [14] and discuss under which assumptions this strategy increases the reliability and when, on the other hand, produces incorrect results, i. e. the empirical significance level $\alpha$ does not correspond to the theoretical one. In that context, a *reliable test* should be understood as a test such that the probability of a false positive (Type I error) is agreed with the expected one.

[1]Actually, the *Non-overlapping Template Matching*, *Random Excursions*, *Random Excursions Variant*, *Serial* and *Cumulative Sums* tests generate 148, 8, 18, 2 and 2 $p$-values correspondingly, other tests — one $p$-value each; however it is very common considering only one of $p$-value from the set of $p$-values related to the test.

| # | **Test Name** | **Used Statistics** |
|---|---|---|
| 1 | *Frequency* | normalized modulus of the difference between frequencies of 1 and 0 |
| 2 | *Block Frequency* | $\chi^2$ statistics of 1 frequencies in adjacent non-intersecting 128-bit blocks |
| 3 | *Cumulative Sums* | maximal deviation from 0 for partial sums of $\pm 1$ walk constructed by the binary sequence |
| 4 | *Runs* | the total number of 1-runs and 0-runs |
| 5 | *Longest Run* | $\chi^2$ statistics of maximal lengths of 1-runs frequencies in adjacent non-intersecting $10^4$-bit blocks |
| 6 | *Binary Matrix Rank* | $\chi^2$ statistics of binary $32 \times 32$-matrices ranks of frequencies formed from adjacent nonintersecting 1024-bit blocks of the sequence |
| 7 | *Discrete Fourier Transform* | normalized difference between the number of DFT coefficients of $n$-bit sequence, exceeding $\sqrt{n \ln 20}$ in absolute value, and $0.95n/2$ |
| 8 | *Overlapping Templ.Match.* | $\chi^2$ statistics of 9-bit 1-series frequencies in 1032-bit adjacent blocks with overlappings |
| 9 | *Universal statistical test* | sum of base 2 logarithms of distances between equal nonintersecting 7-bit blocks |
| 10 | *Approximate Entropy* | difference of logarithmic frequencies statistics of 10- and 11-bit segments with overlappings |
| 11 | *Serial* | differences $\chi^2$ statistics of frequencies of all 14-, 15- and 16-bit segments with overlappings |
| 12 | *Linear Complexity* | $\chi^2$ statistics of shortest LSR lengths frequencies generating 500-bit segments of the sequence |
| 13 | *Non-overlap. Templ.Match.* | $\chi^2$ statistics of aperiodic fixed segments frequencies in 8 adjacent non-overlapping blocks |
| 14 | *Random Excursion* | $\chi^2$ statistics of Random Walk cycles frequencies with fixed numbers visiting of $-4, -3, \ldots, 4$ states |
| 15 | *Random Excrsion Variant* | $\chi^2$ statistics of frequencies of visiting 18 states from $-9$ to 9 by Random Walk |

Table 1: All NIST SP 800-22 tests with brief descriptions

In the beginning of the testing process, the whole bit sequence is divided into $N$ blocks of the length $n$ bits, then a test statistic value is computed for each block. All 15 tests may be divided into two types (according to the test statistic distribution): binomial-based (i.e., the two-sided tests) and chi-square based tests (i.e., the one-sided tests). The Frequency (Monobit) Test,

the Runs Test, the Spectral Test, Maurer's "Universal Statistical" Test, the Random Excursions Variant Test and the Cumulative Sums (Cusum) Test belong to the binomial-based tests, and the others are chi-square based.

For each value of test statistic the $p$-value is computed according to the theoretic test statistic distribution[2]. A test is considered to be passed when the computed $p$-value is larger than the significance level $\alpha$. According to the computed $p$-values for $N$ blocks, each test performs two-level test: the first-level and the second-level tests. The two-level testing approach was found to increase the testing capability [12]. The first-level test focuses on the passing ratio of the $N$ $p$-values, and the second-level test further focuses on the uniformity of the $N$ $p$-values to assess whether the test statistic values follow the expected distribution, i.e., the chi-square distribution.

## 3 Related Work

There are many articles on the NIST SP 800-22 test suite. Among them, S.-J. Kim, K. Umeno and A. Hasegawa in [9] have found that the test setting of Discrete Fourier Transform test and Lempel-Ziv test from an earlier version of NIST SP 800-22 suite are wrong, they give four corrections of mistakes in the test settings, K. Hamano, T. Kaneko in [10, 11] derived the distribution function of the Discrete Fourier Transform spectrum, changed the threshold value from the default value of $\sqrt{3n}$ to the value of $\sqrt{\left(\ln \frac{1}{0.05}\right) n}$, where $n$ is the length of a random number sequence, and corrected the occurrence probabilities $\pi_i$ of the the overlapping template matching test included. F. Sulak, A. Doğanaksoy, B. Ege, O. Koçak in [15] found that the $p$-values for short sequences (less than 512 bits) follow a specific discrete distribution, rather than the assumed uniform distribution for long sequences, calculated the probabilities of subintervals for some NIST test and proposed an alternative approach to evaluate test results for short sequences. F. Pareschi, R. Rovatti, G. Setti in [12] investigated the reliability of the second-level tests, and analyzed the sensitivity to the approximation errors introduced by the computation of $p$-values. Since the sequence length is finite in practice and thus the set of possible statistic values is discrete, F. Pareschi et al. in [17] provided the more closer distributions to the actual ones of $p$-values for the Frequency Test, the Runs Test, the Spectral Test.

[2]Remind that the $p$-value is the probability to obtain the value of statistic at least as large as the actually observed one, under the assumption that the null hypothesis is true.

# 4 First-level testing approach of the NIST SP800-22

Denote by $\mathcal{H}_0$ the null hypothesis on the uniformity and independence of binary sequence elements, i.e. that elements of binary sequence are independent and take values 0 and 1 with probabilities $\frac{1}{2}$. In other words, hypothesis $\mathcal{H}_0$ is that the random number generator under test produces sequences with the above properties, such a generator will be called an *ideal random number generator* further. In the statistical hypothesis testing, a test statistic is chosen and used to determine whether $\mathcal{H}_0$ should be accepted or rejected. Under the null hypothesis $\mathcal{H}_0$, the theoretical reference distribution of the statistic is determined by the mathematical methods. From the reference distribution, a range of acceptable statistic values is determined based on a preset coefficient $\gamma = 1 - \alpha$ (for example, $\gamma = 0.99$), $\alpha$ is the *significance level*, i.e., $\gamma$ is the probability that the statistic value is inside of the acceptable statistic value range. If the test statistic value lies outside the acceptable statistic value range, the hypothesis $\mathcal{H}_0$ is rejected, otherwise, $\mathcal{H}_0$ is accepted.

A randomness test suite may contain a serial of tests, which evaluate different aspects of randomness. Since these tests produce different acceptable statistic value ranges based on the same $\gamma$, then the $p$-value $p$ is used as a unified measure for different tests, which is calculated using the test statistic. In [14] the $p$-value is defined as "the probability that a perfect random number generator would have produced a sequence less random than the sequence that was tested". More specifically, the $p$-value is the probability of obtaining a statistic value $s$ equal to or "more extreme" than the observed statistic value $s_{obs}$ for the tested sequence, under the assumption that the null hypothesis is correct. According to the definition of "more extreme" cases, the tests are generally divided into two categories: one-sided tests and two-sided tests.

When $\mathcal{H}_0$ is true and $p \leq \alpha$ we have a *false positive* in the test interpretation. This is called Type I error, we can compute its probability since we have a complete characterization of the sequences generated by a true RNG. It is known that if the distribution of statistic is continuous then $p$-values should be uniformly distributed under the null hypothesis. Since $p$ is uniformly distributed, the probability of a Type I error is

$$\mathbf{P}_{\mathcal{H}_0}\{p \leq \alpha\} = F_p^{(0)}(\alpha) = \alpha.$$

For this reason, $\alpha$ is also called *level of significance* or *statistical significance*. The value of $\alpha$ is usually small, the significance level recommended by NIST is $\alpha = 0.01$.

On the contrary, the fact that $p > \alpha$ when $\mathcal{H}_0$ is false we have a *false negative* in the test interpretation. This is called Type II Error (accepting the sequence as random when $\mathcal{H}_0$ is false), and we denote this probability by $\beta$.

So we can commit two errors:

I) reject $\mathcal{H}_0$, when the sequence is generated by an ideal random number generator.

II) accept $\mathcal{H}_0$, when the sequence is generated by a generator that is non-ideal.

It's important to find a balance between the probabilities of making Type I and Type II errors. Reducing $\alpha$ always comes at the cost of increasing $\beta$, and vice versa. This approach is known as *first-level testing* and is followed by [14]. The value $(1 - \beta)$ is also called *statistical power* of the test, its exact computation is not possible and also not sensible, since it depends on the specific (non-ideal) generator under test.

## 5 Second-level testing approach of the NIST SP800-22

The usual way to test a true random or pseudo random number generator is to generate a sequence of $n$ bits and analyze it with the test suite. Given a level of significance $\alpha$, the sequence may be considered random if all tests in the suite produce $p$-values greater than $\alpha$.

This approach presents a serious weakness. It is well known that some pseudo-random generators can easily pass all tests. For example a periodic (and thus, not ideal) generator always passes the Frequency test if the number of 1s and of 0s in the sequence is balanced.

To overcome this weakness, a more intensive test is necessary, involving a number $N$ of different sequences generated by the RNG under test. In [14] NIST recommends using the second-level testing approach; a long binary sequence is partitioned into $N$ subsequences, each with $n$ bits. A standard test is applied for each sequence, and the distribution of the $N$ obtained $p$-values is compared with a uniform distribution $F_p^{(0)}$. Note that from the definition of null hypothesis it follows that if $\mathcal{H}_0$ is true, then the $N$ sequences (and so the $N$ $p$-values) are independent of one another and approximately uniformly distributed on $[0, 1]$. To check this in [14] NIST proposes a chi-square goodness-of-fit test, this test is again a statistical test and gives another (a second-level) $p$-value $p_{II}$.

The NIST SP 800-22 suite includes 15 tests and for each of them the second-level test is performed. For each test and for each of $N$ blocks of the sequence $N$ $p$-values are computed, then the following second-level test is performed based on these $N$ $p$-values.

This approach has been known for a long time [19] and it may increase the testing power as compared to a standard approach [12, 17]:

(a) given the empirical results for a particular statistical test, compute the fraction $\zeta$ of the sequences passed the test. For example, if $N$ binary sequences were tested with the significance level $\alpha = 0.01$ and $m$ binary sequences had $p$-values equal to or greater than $\alpha$, then the fraction is $\zeta = m/N$. If $\mathcal{H}_0$ is true, then the distribution of $\zeta N$ is binomial with parameters $N$ and $(1-\alpha)$; if $N$ is large enough (e.g., $N \geq 1000$), $\zeta$ can be approximated with a normal random variable $\mathcal{N}(\mu, \sigma)$, where

$$\mu = 1 - \alpha \quad \text{and} \quad \sigma = \sqrt{\alpha(1-\alpha)/N}.$$

The range of acceptable fractions may be determined by the interval

$$\left[1 - \alpha - 3\sqrt{\frac{\alpha(1-\alpha)}{N}}, 1 - \alpha + 3\sqrt{\frac{\alpha(1-\alpha)}{N}}\right].$$

If the $\zeta$ falls outside of this interval, then there is evidence that the data does not conform to the null hypothesis and $\mathcal{H}_0$ is rejected. The normal distribution approximation for the binomial distribution is reasonably accurate for large sample sizes.

(b) given $N$ sequences, the distribution of $p$-values is examined to check uniformity. The interval $[0, 1]$ is divided into $k$ disjoint sub-intervals, and the number of $p$-values falling in each sub-interval are counted. Uniformity is determined via an application of a *chi-square* goodness-of-fit test in $k$ sub-interval with statistic $\chi^2$. This is accomplished by computing chi-square statistic value

$$\chi^2_{obs} = \sum_{i=1}^{k} \frac{(\pi_i - N/k)^2}{N/k},$$

where $\pi_i$ is the number of $p$-values in $i$-th sub-interval. This statistical test yields a level-two $p$-value $p_{II} = \mathbf{P}_{\mathcal{H}_0}\left\{\chi^2 > \chi^2_{obs}\right\}$, which is calculated as follows

$$p_{II} = \frac{1}{\Gamma((k-1)/2)} \int_{\chi^2/2}^{\infty} t^{(k-1)/2-1} e^{-t} dt, \ \Gamma((k-1)/2) = \int_0^{\infty} t^{(k-1)/2-1} e^{-t} dt.$$

Given a significance $\alpha_{II}$, $\mathcal{H}_0$ is rejected if $p_{II} \leq \alpha_{II}$, otherwise the sequences may be considered to be uniformly distributed. In [14] NIST recommends to use $\alpha_{II} = 0.0001$ and $k = 10$ sub-intervals.

The choice of $N$ in a two-level approach is usually a tradeoff. For example, the length of the sequence $n$ may be limited by the memory size or by the computational power available. On the other hand, it may be preferable to test short sequences, thereby limiting $n$.

In every statistical test some approximations are adopted, introducing errors in the $p$-value computation and so in the $p$-value distribution. It was observed [12, 17] that for extremely large values of $N$, the level-two approach always fail, it ends with $p_{II} \simeq 0$. In this case we can say that the test is *not reliable*, since its significance is sensible different from the expected one. From this point of views, $N$ is also a trade-off between statistical power $(1 - \beta)$ and reliability of a level-two test.

## 6 Advantages of two-level test

The undoubted advantage of using the two-level test, for example, is rejection the generator with a small period size by Frequency test: regardless of the sequence length, a basic first-level test is always passed, while the advantage of the second-level test is that it is able to recognize that a generator that always passes a basic test is not ideal.

Two pseudo-random generator were considered in [12]: the 32 bits version of the KISS [3], which is a very simple but effective generator, and the BBS generator [2]:

$$x_{n+1} = x_n^2 \bmod M,\ M = P \cdot Q,\ P, Q \text{ — large distinct primes,}$$

$x_0$ is a quadratic residue $\bmod M$; the output of the BBS generator is the bit parity of $x_{n+1}$, that is a computationally complex pseudo-random generator that, under a certain intractability assumption, has proven to be cryptographically secure (i.e. polynomial-time unpredictable, see [2]). For both generators in [12] were performed a first-level test on a single sequence and a second-level test, checking the uniformity of $N = 10000$ $p$-values, on $N$ different sequences with a chi-square test over 16 sub-intervals, and with a Kolmogorov-Smirnov test.

The empirical distributions of the tsets of $p$-values were computed for both generators, which were then compared with the uniform distribution in $[0, 1]$ by calculating the $p$-values $p_{II}$. In this case, $\mathcal{H}_0$ correspond to "the

two distributions match"; again, we reject $\mathcal{H}_0$ if $p_{II} \leq \alpha_{II}$ and accept $\mathcal{H}_0$ if $p_{II} > \alpha_{II}$. Although in [14] NIST recommends to use $\alpha_{II} = 0.0001$, the value of $\alpha_{II}$ in [12] was set to $\alpha = 0.01$ to make possible a direct comparison between a first- and a second-level tests. The authors noted that the comparison may seem unfair, since a first level-test considers $n$ bits, while a second-level $nN$.

Both generators had passed all first-level tests; however (except for the Spectral test that is well-known to have an error in the parameter of the reference distribution of the considered statistic) only the BBS generator passed the second-level tests. The proposed second-level test is able to recognize the non-randomness of the KISS generator, while a simple first-level test fails. In addition to the above test, the authors of [12] considered a second one involving a much larger number of sequences generated with the BBS algorithm ($N = 150,000$ sequences) in order to obtain more reliable results. The test was also applied to three high-end physical process based true-random generators. All three generators have been considered with an additional post processing stage, this in order to hide all possible imperfections and be sure that the properties of analyzing streams are close to the properties of sequences of independent and equiprobable bits. Results of experiments are far from the desired ones, since too many tests fail.

## 7 Experiments

All 15 tests from Table 1 were applied to pseudorandom sequences generated by AES block cipher. The detailed description of the design of AES-based RNG may be found in [22, 23]. For generation such sequences AES block cipher in the CFB (Cipher Feedback Block) mode (see Fig.1) with the zero initialization vector (IV) and plaintext block (Plaintext) is used; the key for each sequence was randomly selected from the set of 128-bit binary sequences, the length in bits of all sequences was chosen to be the same and equal to $n = 2^{20} = 1.048.576$ each (value corresponds to the NIST input size recommendation for the length of tested sequences).

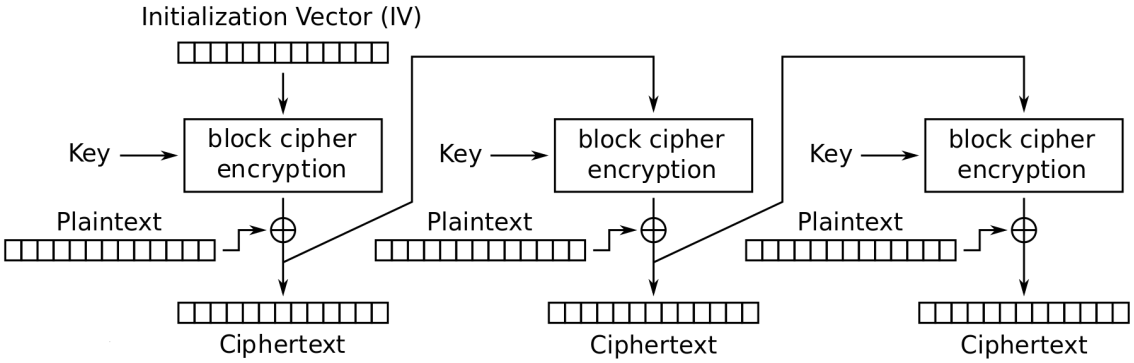



Figure 1:

Keys (on Fig.1) were obtained using the open source cryptographic library OpenSSL:

```
int RAND_bytes(unsigned char *buf, int num);
```

this function generates `num` bytes using the cryptographicaly secure generator of pseudorandom numbers (CSPRNG), and saves them in the `buf` array. By default, the OpenSSL CSPRNG supports a security level of 256 bits, provided it was able to seed itself from a trusted entropy source [25].

If we will use the AES-based pseudorandom generator, then this allow us to arbitrarily increase $N$. The effect of increasing the testing power can be seen in the example of Table 2, where the testing results for AES-based RNG are shown. The two-level test was performed for all 188 statistics values computed by NIST test suite, but only 15 of them are presented in the Table 2. In order to have the same significance in all tests, $\alpha_{II}$ in all experiments was set equal to $\alpha = 0.01$. These results confirm that for extreme values of $N$ (e.g. $N = 2^{20}$) the two-level testing approach was carried out on the sequences obtained by AES algorithm (it is known that such sequences are practically indistinguishable from random one, see [22, 23]) are failed too, i.e. it ends with $p_{II} \simeq 0$.

| # | Test Name | $\chi^2$ test, $N$ sequences $N = 10^3$ | $N = 10^4$ | $N = 10^5$ | $N = 2^{20}$ |
|---|---|---|---|---|---|
| 1 | *Frequency* | 0.616305 | 0.290806 | 0.588411 | **0.000803** |
| 2 | *Block Frequency* | 0.187581 | 0.773212 | 0.374097 | **0.000125** |
| 3 | *Cumulative Sums* | 0.401199 | 0.124765 | 0.959543 | **0.000009** |
| 4 | *Runs* | 0.150340 | 0.885418 | 0.910568 | 0.107966 |
| 5 | *Longest Run* | 0.610070 | 0.239883 | **0.000355** | **0.000000** |
| 6 | *Binary Matrix Rank* | 0.878618 | 0.341017 | **0.000000** | **0.000000** |
| 7 | *Discrete Fourier Transform* | 0.371941 | 0.014836 | **0.000000** | **0.000000** |
| 8 | *Overlapping Templ. Match.* | 0.071177 | 0.202268 | **0.000000** | **0.000000** |
| 9 | *Universal statistical test* | 0.574903 | 0.108534 | **0.000000** | **0.000000** |
| 10 | *Approximate Entropy* | 0.246750 | 0.078038 | 0.219501 | **0.000000** |
| 11 | *Serial* | 0.942198 | 0.174057 | 0.213964 | 0.572679 |
| 12 | *Linear Complexity* | 0.839507 | 0.279152 | 0.299852 | 0.117305 |
| 13 | *Non-overlap. Templ. Match.* | 0.092041 | 0.372782 | 0.121382 | 0.275416 |
| 14 | *Random Excursion* | 0.914727 | 0.663838 | 0.346173 | **0.000028** |
| 15 | *Random Excursion Variant* | 0.238264 | 0.133576 | **0.000080** | **0.000000** |

Table 2: Results of the chi-square based two-level randomness test for the AES-based RNG, with $N$ ranging from 1000 to 1048576. Tests where $p_{II} \leq 0.01$ are in bold

In order to identify the problem, let's take a look on the simple Frequency test.

# 8 Frequency (Monobits) Test

The purpose of the Frequency test is to determine whether the number of ones and zeros in a sequence are approximately the same as would be expected for a truly random sequence. The test assesses the closeness of the fraction of ones to 1/2, that is, the number of ones and zeroes in a sequence should be about the same. All subsequent tests depend on passing this first basic test.

Let $n$ be the length of the bit string, $\varepsilon = \varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n$ be the input sequence of bits and the null hypothesis $\mathcal{H}_0$ is that the sequence $\varepsilon$ consists of independent identically distributed Bernoulli random variables ($X_i$ or $\varepsilon_i$), where $X_i = 2\varepsilon_i - 1$, and so the probability of ones is 1/2. Denote by

$$S_n = X_1 + \ldots + X_n = 2(\varepsilon_1 + \ldots + \varepsilon_n) - n. \quad (1)$$

Under the null hypothesis, $S_n$ is assumed to follow a binomial distribution. By the classic De Moivre-Laplace theorem, for a sufficiently large number of trials, the distribution of the binomial sum, normalized by $\sqrt{n}$, is closely approximated by a standard normal distribution. This test makes use of that approximation to assess the closeness of the fraction of 1's to 1/2.

The test is derived from the well-known central limit theorem for the random walk, $S_n = X_1 + \ldots + X_n$. According to the central limit theorem,

$$\lim_{n\to\infty} F_{S_n}(z) = \Phi(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} e^{-u^2/2} du, \text{ where } F_{S_n}(z) = \mathbf{P}\left\{ \frac{S_n}{\sqrt{n}} < z \right\}.$$

This classical result serves as the basis of the simplest test for randomness. It implies that, if $S_n$ is normal, then $|S_n|$ is half normal (i.e. $F_{|S_n|}(z) = 2F_{S_n}(z), z \geq 0$) and

$$F_{|S_n|}(z) = \mathbf{P}\left\{ \frac{|S_n|}{\sqrt{n}} \leq z \right\}, \quad \lim_{n\to\infty} F_{|S_n|}(z) = 2\Phi(z) - 1.$$

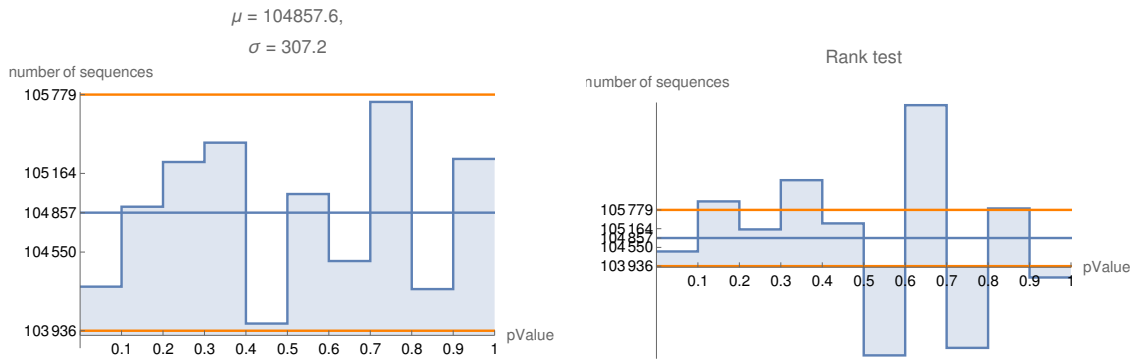
So, according to the test statistic $S = |S_n|/\sqrt{n}$, it is necessary to determine observed value $S(\varepsilon) = |X_1 + \ldots + X_n| / \sqrt{n}$, and then calculate the corresponding $p$-value, which is equal to

$$p_I = \lim_{n\to\infty} \left(1 - F_{|S_n|}(S(\varepsilon))\right) = 2\left(1 - \Phi(S(\varepsilon))\right| = erfc\left(\frac{S(\varepsilon)}{\sqrt{2}}\right), \qquad (2)$$

where $erfc(\cdot)$ is the complementary error function

$$erfc(z) = \frac{2}{\sqrt{\pi}} \int_{z}^{\infty} e^{-u^2} du\,.$$

Figure 2 (a) shows the histogram of the set of $p$-values for $k = 10$ sub-intervals corresponding to the Frequency test applied to the AES-based generator, and for the comparison Figure 2 (b) — the Binary Matrix Rank test from the NIST test suite applied to the same sequences from the AES-based generator. If we consider the theoretical standard deviation of the number of random $N = 2^{20}$ independent, uniformly distributed values over $k = 10$ sub-intervals, we easily get $\sigma = \sqrt{N(k-1)}/k$. In this case, $\sigma \simeq 307.2$, the upper and lower continuous lines in the figures correspond to the deviation of $3\sigma$ from the mean value; the figure shows that the observed deviation is far from this value.




(a) (b)

Figure 2: Comparison between expected deviation and measured deviation in the distribution of $p$-values in the interval $[0, 1]$ for (a) Frequency test, (b) Binary Matrix Rank test

Let $p_1, \ldots, p_N \in [0, 1]$ be the sample of $p$-values with assumed continuous cumulative distribution function (CDF) $F(x)$. The empirical CDF $F_N(x)$ for this sample is defined as follows

$$F_N(x) = \frac{1}{N} \sum_{j=1}^{N} H\left(x - p_j\right),$$

where $H(x)$ is the unity-step function, defined as $H(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}$. The Kolmogorov-Smirnov statistic for a given CDF $F(x)$ is

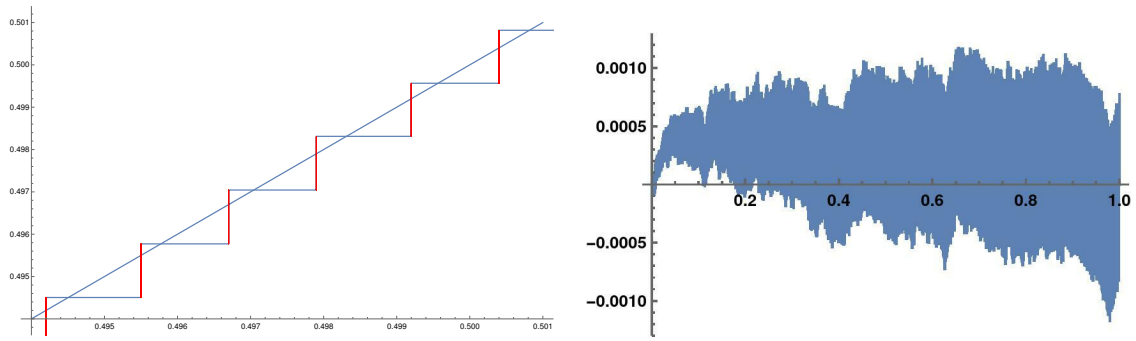
$$D_N = \sup_x |F_N(x) - F(x)|.$$

Let $\mathcal{K}$ be the CDF of $D_N$, then the $p$-values is given by $1 - \mathcal{K}(D_N)$. Under null hypothesis that the sample comes from the hypothesized distribution $F(x)$

$$\sqrt{N} D_N \xrightarrow[N \to \infty]{} \sup_t |B(F(t))|$$

in distribution, where $B(t)$ is the Brownian bridge.

Figure 3 (a) shows the discrete empirical CDF $F_N$ of $p$-values for Frequency test applied to the AES-based sequences, Figure 1 (b) — differences between empirical $F_N$ and uniform CDF, $n = N = 2^{20}$.




(a) (b)

Figure 3: (a) Comparison between empirical CDF $F_N$ of the $p$-values generated by Frequency test for $n = 2^{20}$ and uniform CDF, (b) Deviation of the empirical CDF $F_N$ of the $p$-value distribution of the Frequency test from continuous uniform CDF

The deviation may be identified with an error propagated from the computation of the $p$-value in the first-level test which is a consequence of the introduced approximations. The random variable $S_n$ in (1) has a binomial distribution, which is however assumed to be normal. From Berry and Esseen theorem, we know that the error of this approximation, under the assumptions of the central limit theorem, is bounded by (see [5])

$$\sup_x |F_{S_n}(x) - \Phi(x)| \leq \frac{C\mathbf{E}|X_i|^3}{\sigma^3\sqrt{n}},$$

where $F_{S_n}(z) = \mathbf{P}\left\{\frac{S_n}{\sqrt{n}} < z\right\}$, $\Phi(\cdot)$ is a CDF of the standard normal distribution; $n$ is the number of independent variables $X_i$, $1 \leq i \leq n$, summed in (1), i.e. the number of bits in the sequence; $\sigma = \mathbf{E}\left(X_i^2\right) = 1$; the third order moment is $\mathbf{E}|X_i|^3 = 1$; and $C < 0.4748$ (see [16]). The maximal error $\epsilon$ of $p$-value is the error of the approximation $F_{|S_n|}(x)$ by $\Phi(x)$ and from (2) is twice the above error, i.e. $\epsilon = 2C/\sqrt{n}$. If $n = 2^{20}$, then $\epsilon = 9,3 \cdot 10^{-4}$.

Assuming this bound of the error in the computation of a $p$-value, we can bound also the maximal error in the distribution of $N$ $p$-values in $k$ sub-intervals. A $p$-value that should belong to a sub-interval may be found into the nearby one only if its distance from the border of two sub-intervals is less than $\epsilon$. If we have $N$ $p$-values uniformly distributed in $[0, 1]$ the maximal number of $p$-values that may be found in the wrong adjacent sub-interval is $\epsilon N$. This is independent of the numbers of sub-intervals.

Since all sub-intervals (but the first and the last), have two neighbors, the maximum error $\Delta$ in the number of $p$-values in a bin is $\Delta = 2N\epsilon$. In our case, $N = 2^{20}$, so $\Delta \simeq 972.4$. This value is compatible with that may be observe in the Figure 2 (a). The increasing $n$ will result in decreasing the

propagated error as $1/\sqrt{n}$ (the results of the experiments are confirm that in [12]). More generally, increasing $n$ will reduce the error in the first-level $p$-value and so increasing the reliability of a second-level test, for all tests in the suite. A very simple reliability condition is requiring that $\Delta$ is smaller than the standard deviation of the number of $p$-values in a sub-interval from its mean value, i.e.

$$\Delta < \sqrt{N(k-1)}/k\,.$$

In the case $\Delta = 2N\epsilon$, $k = 10$ and $n = 2^{20}$, we get

$$N \leq \frac{1}{4\epsilon^2 k}\left(1 - \frac{1}{k}\right) \simeq 26163,7.$$

## 9 Two-sided estimates for the quantiles of the binomial distribution

To construct a statistical test with the specified error probabilities, we need the information on the distribution "tails". But relative errors of the Moivre–Laplace approximations for the tails of binomial distribution function are large. This was one of the first problems in the theory of large deviations. There are a lot of inequalities for binomial distribution tails, as well as for the distributions of sums of random variables.

In [18] explicit two-sided estimates for the distribution function $F_{n,p}(k)$ of the binomial law with parameters $n, p$ were obtained. The character of these results is analogous to the Bernstein and Feller theorems, but the formulas are explicit and (from a practical viewpoint) constitute an almost final solution of the large deviation problem for the binomial law.

Let $X_{n,p}$ be a random variable with the binomial distribution with parameters $(n, p)$:

$$\mathbf{P}\{X_{n,p} \leq k\} = \sum_{0 \leq m \leq k} C_n^m p^m (1-p)^{n-m}.$$

**Theorem 1** ([18])**.** *Let*

$$H\,(x,p) = x \ln\left(\frac{x}{p}\right) + (1-x)\ln\left(\frac{1-x}{1-p}\right), \quad \operatorname{sign}(x) = \frac{x}{|x|}$$

*for* $x \neq 0$ *and* $\operatorname{sign}(0) = 0$*, and let* $\{C_{n,p}(k)\}_{k=0}^{n}$ *be increasing sequences defined as follows:*

$$C_{n,p}(0) = (1-p)^n,\ C_{n,p}(n) = 1 - p^n,$$

$$C_{n,p}(k) = \Phi\left(\operatorname{sign}\left(\frac{k}{n} - p\right)\sqrt{2nH\left(\frac{k}{n}, p\right)}\right), \quad 1 \le k < n.$$

*Then for every* $k = 0, 1, \ldots, n-1$ *and for every* $p \in (0,1)$

$$C_{n,p}(k) \le \mathbf{P}\{X_{n,p} \le k\} \le C_{n,p}(k+1) \tag{3}$$

*and equalities hold only for* $k = 0$ *and* $k = n-1$.

We will use the following standard notations

$$\lfloor x \rfloor = \max\{m \in \mathbb{Z} \mid m \le x\}, \quad \lceil x \rceil = \min\{n \in \mathbb{Z} \mid n \ge x\}.$$

By definition of the $\alpha$-level quantile $x_\alpha(n,p)$ of the binomial distribution with parameters $(n,p)$

$$x_\alpha(n,p) = \min\{k\colon \mathbf{P}\{X_{n,p} \le k\} \ge \alpha\} \in \{0,1,\ldots,n\},$$
$$x_0(n,p) = 0 \text{ and } x_1(n,p) = n,$$
$$x_{1/2}(n,p) = \lfloor n/2 \rfloor.$$

From Theorem 1 for any $\alpha \in [0,1]$ it follows that

$$\min\{k\colon C_{n,p}(k+1) \ge \alpha\} \le x_\alpha(n,p) = \min\{k\colon \mathbf{P}\{X_{n,p} \le k\} \ge \alpha\} \le \min\{k\colon C_{n,p}(k) \ge \alpha\}. \tag{4}$$

Recently the author had obtained the two-sided estimates for the $\alpha$-level quantile $x_\alpha(n,p)$ of the binomial law. Such estimates allow for any $\alpha$-level to find two subsequent integers one of which is the quantile $x_\alpha(n,p)$. These estimates allow to completely eliminate the systematic error in determining one of the $k$ disjoint sub-intervals to which the $p$-value belong on $[0,1]$ for the Frequency test.

**Theorem 2.** *The following estimates are true for* $0 < \alpha < 1/2$

$$r_1 - 1 \le x_\alpha(n,p) \le r_1,$$

*where* $r_1 = \left\lceil np + \frac{q-p}{6}\Phi^{-1}(1-\alpha)^2 - \Phi^{-1}(1-\alpha)\sqrt{npq + \frac{(q-p)^2}{6^2}\Phi^{-1}(1-\alpha)^2}\right\rceil$,

*for* $1/2 < \alpha \le 1$

$$r_2 - 1 \le x_\alpha(n,p) \le r_2,$$

*where* $r_2 = \left\lceil np + \frac{q-p}{6}\Phi^{-1}(\alpha)^2 + \Phi^{-1}(\alpha)\sqrt{npq + \frac{(q-p)^2\Phi^{-1}(\alpha)^2}{6^2}}\right\rceil$.

**Corollary 1.** *The following estimates are true*

$$r_1 - 1 \le x_\alpha(n, 1/2) \le r_1, \quad 0 \le \alpha < 1/2,$$

*where* $r_1 = \left\lceil \frac{n}{2} - \frac{\sqrt{n}}{2}\Phi^{-1}(\alpha) \right\rceil$ ,

$$r_2 - 1 \le x_\alpha(n, 1/2) \le r_2, \quad 1/2 < \alpha \le 1,$$

*where* $r_2 = \left\lceil \frac{n}{2} + \frac{\sqrt{n}}{2}\Phi^{-1}(\alpha) \right\rceil$ .

The 11 quantiles of binomial distribution with parameters $(2^{20}, 1/2)$ are shown in the table below

| $\alpha$ | 0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $x_\alpha(2^{20}, \frac{1}{2})$ | 0 | 523632 | 523857 | 524020 | 524158 | 524288 | 524418 | 524556 | 524719 | 524944 | $2^{20}$ |

Table 3: Quantiles of binomial distribution

# 10 Conclusion

The two-level approach for testing RNGs involving the well known NIST SP 800-22 test suite was considered. Such approach may increase the reliability of the test. However it is sensitive to the approximation error introduced by the computing of $p$-values. In this paper it has been shown that for AES-based sequences two-level testing approach is not reliable too. Systematic error in the computing of the $p$-values is dependent only on $n$, that is the number of bits in the analyzed sequences. For a reliable second-level test, this error should be approximately no greater than $\sigma/N = \frac{1}{k}\sqrt{\frac{k-1}{N}}$, where $\sigma$ is the standard deviation from the mean number of particles in a bin for equiprobable scheme of allocation $N$ particles in $k$ bins (disjoint sub-intervals of $[0, 1]$). For example in the second-level test of the Frequency test with $n = 2^{20}$ the number of tested sequences $N$ should not exceed 26184.

To completely eliminate the systematic error appearing in the Frequency test when determining the number of bin from $k$ bins to which the $p$-value belongs the two-sided estimates of the quantiles of the binomial law are proposed.

# References

[1] Chernoff H., "A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations", *Ann. Math. Stat.*, **23** (1952), 493–507.

[2] Blum L., Blum M., Shub M., "A simple unpredictable pseudo-random number generator", *SIAM J. Computing*, **15** (1986), 364–383.

[3] Marsaglia G., Zaman A., "The KISS generator", Technical Report, Department of Statistics, University of Florida, 1993.

[4] Alfers D., Dinges H., "A normal approximation for beta and gamma tail probabilities", *Z. Wahrscheinlichkeitstheor. verw. Geb.*, **65**:3 (1984), 399–420.

[5] Shiryaev A.N., Boas R.P., *Probability (Graduate Texts in Mathematics)*, Springer-Verlag, 1995.

[6] Marsaglia G., "DIEHARD: a battery of tests of randomness", 1996, http://stat.fsu.edu/ geo/diehard.html.

[7] Menezes A.J., van Oorschot P.C., Vanstone S.A., *Handbook of Applied Cryptography*, CRC Press, 1996.

[8] Rukhin A., Soto J., Nechvatal J., Smid M., Barker E., Leigh S., Levenson M., Vangel M., Banks D., Heckert A., Dray J., Vo S., "A statistical test suite for random and pseudorandom number generators for cryptographic applications", NIST Special Publication 800-22, May 15, 2001, http://csrc.nist.gov/rng/SP800-22b.pdf.

[9] Kim S-J., Umeno K., Hasegawa A., *https://eprint.iacr.org/2004/018.pdf.*

[10] Hamano K., "The distribution of the spectrum for the discrete fourier transform test included in SP800-22", *IEICE Tran. Fund.*, **E88-A**:1 (2005).

[11] Hamano K., Kaneko T., "Correction of overlapping template matching test included in NIST randomness test suite", *IEICE Trans. Fund.*, **90–A**:9 (2007), 1788–1792.

[12] Pareschi F., Rovatti R., Setti G., "Second-level NIST randomness test for improving test reliability", ISCAS 2007 (New Orleans (USA), May 27–30, 2007), 1437–1440.

[13] Pareschi F., Rovatti R., Setti G., "On the approximation errors in the frequency test included in the NIST SP800-22 statistical test suite", APCCAS 2008 (Macao, China, 2008), 1216–1219.

[14] Rukhin A., Soto J., Nechvatal J., Smid M., Barker E., Leigh S., Levenson M., Vangel M., Banks D., Heckert A., Dray J., Vo S., "A statistical test suite for the validation of random number generators and pseudo random number generators for cryptographic applications", NIST Special Publication 800-22 Revision 1a, 27 April 2010.

[15] Sulak F., Doğanaksoy A., Ege B., Koçak O., "Evaluation of randomness test results for short sequences", SETA 2010, Lect. Notes Comput. Sci., **6338**, 2010, 309–319.

[16] Korolev V., Shevtsova I., "An improvement of the Berry–Esseen inequality with applications to Poisson and mixed Poisson random sums", *Scandinavian Actuarial J.*, **2** (2012), 1–25, arXiv: 0912.2795.

[17] Pareschi F., Rovatti R., Setti G., "On statistical tests for randomness included in the NIST SP800-22 test suite and based on the binomial distribution", *IEEE Trans. Inf. For. Sec.*, **7**:2 (2012), 491–505.

[18] Zubkov A. M., Serov A. A., "A complete proof of universal inequalities for distribution function of binomial law", *Theory Probab. Appl.*, **57**:3 (2013), 539–544.

[19] L'Ecuyer P., Simard R., *TestU01*, Dept. d'Inform. Rech. Oper. Univ. Montreal, 2013, 214 pp., http://simul.iro.umontreal.ca/testu01/guideshorttestu01.pdf.

[20] Short M., "Improved inequalities for the Poisson and Binomial distribution and upper tail quantile functions", Article ID 412958, *Hindawi*, 2013, 6 pp.

[21] Janson S. Large deviation inequalities for sums of indicator variables, 2016, arXiv: https://arxiv.org/abs/1609.00533.

[22] Zubkov A. M., Serov A. A., "Testing the NIST Statistical Test Suite on artificial pseudorandom sequences", *Matematicheskie Voprosy Kriptografii*, **10**:2 (2019), 89–96.

[23] Zubkov A. M., Serov A. A., "A natural approach to the experimental study of dependence between statistical tests", *Matematicheskie Voprosy Kriptografii*, **12**:1 (2021), 131–142.

[24] Short M., "On binomial quantile and proportion bounds with applications in engineering and informatics", *Communication in Statistics — Theory and Methods*, 17 pp.

[25] https://www.openssl.org/docs/man1.1.1/man3/RAND_bytes.html.