9th Workshop on Current Trends in Cryptology (CTCrypt 2020)

September 15-17, 2020, Dorokhovo, Ruza District, Moscow Region, Russia

Pre-proceedings

In cooperation

— General partner —

infotecs

— Official partner —

КРИПТОПРО

— Support —

AVANGARD media group

ФСРБИТ

CTCrypt 2020 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

Sergey Agievich	–	Research Institute for Applied Problems of Mathematics and Informatics, Belarus
Sergey Aleshnikov	–	Immanuel Kant Baltic Federal University, Russia
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
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
Thomas Peyrin	–	Nanyang Technological University, Singapore
Eduard Primenko	–	Lomonosov Moscow State University, Russia
Boris Ryabko	–	Institute of Computational Technologies SB RAS; and Novosibirsk State University, Russia
Markku-Juhani Olavi Saarinen	–	Independent expert on information security, Finland
Igor Semaev	–	The University of Bergen, Norway
Vasily Shishkin	–	TC 026, Russia
Stanislav Smyshlyaev	–	TC 026, Russia
Alexey Tarasov	–	Federal Educational and Methodical Association in System of Higher Education on Information Security, Russia
Andrey Trishin	–	«Certification Research Center» Ltd., Russia
Alexey Urivskiy	–	«InfoTeCS», JSC, Russia
Amr Youssef	–	Concordia University, Canada
Andrey Zyazin	–	Russian Technological University (MIREA), Russia

Invited Talks

Quantum Security of Feistel Ciphers

Tetsu Iwata

Nagoya University, Nagoya, Japan
tetsu.iwata@nagoya-u.jp

Abstract

It is well known that quantum computers can break a number of public key cryptosystems. On the other hand, the impact of quantum computers on symmetric key cryptosystems is largely unexplored. In 2010, Kuwakado and Morii demonstrated that, based on Simon's quantum period finding algorithm [Simon, SIAM J. Comput., 1997], the 3-round Feistel cipher can be broken in polynomial time if an adversary can make superposition queries [Kuwakado, Morii, ISIT 2010]. Since then, the quantum security of various symmetric key cryptosystems has been analyzed.

In this talk, we review the developments on the quantum security analysis of Feistel ciphers and their variants, covering both quantum attacks and provable security results.

Keywords: Quantum Security, symmetric key cryptosystems, Feistel ciphers.

Some Security Aspects of Contact Tracing Protocols for COVD-19

Mridul Nandi

Indian Statistical Institute, India
mridul.nandi@gmail.com

Abstract

In this talk I will describe what we mean by contact tracing and what are the security concerns. In the last few months several such designs were proposed from several countries. Some of them are centralized and some are decentralized. We discuss some security concerns and how cryptographic tools such as Private Set Intersection can help to resolve those issues.

Keywords: cryptography, contact tracing, implementation.

Elementary quantum cryptography

Igor Arbekov

JSC «InfoTeCS», LLC «SFB Lab», Russia
igor.arbekov@sfblaboratory.ru

Abstract

In the report in an accessible form the *mathematical problems* of proving the *secrecy* of a key in quantum cryptography are formulated. The main stages of the proof of the *secrecy* of a quantum key in terms of *classical* probability theory are presented.

This proof is given on the example of the well-known BB84 protocol (1984), when the so-called attack of individual measurements of quantum states, carried out by the eavesdropping Eve, is considered. This attack *is not optimal*, but it leads to a simple and well-known binary symmetric communication channel with distortions both in the *legitimate* Alice-Bob channel and in the Alice-Eve *interception* channel. Bit errors in these channels are functionally connected. This is all that is needed as a manifestation of the *quantum nature* of the transmission of binary information over an optical channel.

The historical development of the *secrecy criterion of a* key is considered – from the *rough physical* to the entropic one and further to the criterion of the variational distance.

For the privacy amplification key procedure we consider in detail the Leftover-Hash Lemma associated with hashing the bit sequences of Alice and Bob into the final key and giving an estimate of the Eve's average variational distance between the a posteriori distribution and ideal key distribution. An example of the used 2nd order of universal hash functions class is given. We show how the estimates of the variational distance changed in accordance with the use of various conditional entropies of the hashed sequence – from the 2nd order Renyi entropy to the minimum entropy, the minimum smooth entropy, and, finally, to the classical binary Shannon entropy corrected for asymptotics.

A short derivation of the classic leak value associated with the implementation of the error correction procedure in the bit sequence on Bob's side is given.

The historical development of the results related to the relationship between the posterior key distribution and the average complexity of algorithms for finding it, such as testing the most probable key, total testing, truncated testing with a given probability of success is presented.

In conclusion, using the uncertainty relation for minimum smooth entropy, it will be shown how suboptimal the attack of individual measurements is in terms of estimating the magnitude of the quantum leakage.

Keywords: quantum cryptography, BB84, individual attack, secrecy criterions.

Contents

Probabilistic Aspects

Symmetric cryptography

Probabilistic Aspects

A Natural Approach to the Experimental Study of Dependence Between Statistical Tests

Andrei Zubkov and Aleksandr Serov

Steklov Mathematical Institute of Russian Academy of Sciences, Russia
{zubkov, serov}@mi-ras.ru

Abstract

We discuss the results of empirical testing the hypothesis on the independence of tests included in the NIST Statistical Test Suite. To test this hypothesis we consider sets of binary segments and for each segment compute the number of tests which reject this segment as not corresponding to the equiprobable Bernoulli sequence. If the tests were independent, then for segment of equiprobable Bernoulli sequence the number of rejecting tests should have the binomial distribution. It appears that to satisfy this condition we have to exclude some tests from the NIST Statistical Test Suite.

Keywords: NIST Statistical Test Suite, Pseudo Random Number Generators, binary segment, Bernoulli sequence.

1 Introduction

The problem of testing output binary sequences of Pseudo Random Number Generators (PRNG) is very important for cryptography (as well as for other applications of PRNG). The aim of such testing is to decide whether PRNG generates sequences which may be considered as realizations of equiprobable Bernoulli sequence and therefore may be used in cryptosystems.

Denote by H_0 the statistical hypothesis that elements of binary sequence are independent and takes values 0 and 1 with probabilities $\frac{1}{2}$. Under this hypothesis all 2^n binary segments of length n have the same probability 2^{-n}. So, from probabilistic viewpoint there is no reason to consider some n-bit segments as better corresponding to the hypothesis H_0 than other segments.

In fact each test for the hypothesis H_0 realizes checking whether there exists some another hypothesis H^* such that the probability of appearance of tested segments under hypothesis H^* is essentially larger than under H_0. Usually the type of H^* does not stated explicitly: application of test consists

in computation of some statistics (function of tested segments) and hypothesis H_0 is rejected if the value of this statistics falls into predefined "critical" set having too small probability under H_0.

Different tests use different statistics and different "critical" sets. Usually testing of segments of binary sequences is based on some well-known batteries of tests, for example, NIST [1], TestU01 [2] etc. In such case all tests of the battery are applied to each tested segment of sequence.

Let V_n be the set of all 2^n binary segments of length n. If the number of tests used to check H_0 is too large, then almost each segment from V_n may be rejected by some tests. If the number of tests used is bounded and n is not small, then large part of segments from V_n will not be rejected and this part may contain segments having not very complex structure. (We do not discuss here the approach based on the complexity theory.) For example, in [6] we have constructed simple combinations of pairs of binary linear recurrent sequences which were accepted by all tests of the NIST statistical package with high probability (see below sequences of type MixL).

Nevertheless, despite the shortcomings of statistical testing procedures such testing of output sequences generated by PRNG's is necessary: without testing bad PRNG's may be included into cryptosystems and destroy their safety properties.

When a battery of tests is used, some segments may be rejected by several tests. Does it mean that such segments are *very* bad and that the PRNG producing such segments should be considered as non-admissible? Or this effect may be a consequence of dependencies between tests?

In recent years we have investigated these questions by means of different methods. Some recent results on dependencies between tests included into NIST Statistical Test Suite [1] are presented in this note.

2 Short review of NIST Statistical Test Suite

In the complete version of NIST Statistical Test Suite the decision on the acceptance or rejection of hypothesis on the randomness and uniformity of tested sequence is based on the results of 15 tests. The values of statistics of each test are transformed into p-values; the whole number of p-values equals 188 (148 p-values are generated by Non-overlapping Template Matching Test, 8 p-values by Random Excursions Test, 18 p-values by Random Excursions Variant Test, each of Serial and Cumulative Sums Tests generate 2 p-values). For the study of dependencies between tests we have not considered the Non-overlapping Template Matching Test and two tests connected with Random

Excursions (the reasons will be explained at the end of next section). So, we have studied 12 tests generating 14 p-values, see Table 1.

Test No.	Name of Test and character of statistics used
1	**Frequency** (normed difference between frequencies of ones and zeros in the whole segment of the sequence)
2	**Block Frequency** (relative frequences of ones in adjacent nonintersecting 128-bit blocks of the segment of the sequence)
3	**Runs** (the total number of 1-runs and 0-runs in the whole segment of the sequence)
4	**Longest Run** (maximal lengths of 1-runs in adjacent nonintersecting 10^4-bit blocks of the segment of the sequence)
5	**Binary Matrix Rank** (ranks of binary $32x32$-matrices formed from adjacent nonintersecting 1024-bit blocks of segments of the sequence)
6	**Discrete Fourier Transform** (the number of coefficients of discrete Fourier transform of the n-bit segment exceeding $h = \sqrt{n \log \frac{1}{0.05}}$ in absolute value)
7	**Overlapping Template Matching** (numbers of 9-bit 1-series in 1032-bit blocks adjacent intersecting blocks of segment of sequence)
8	**Universal (Maurer's "Universal statistical test")** (sum of base 2 logarithms of distances between equal nonintersecting 7-bit blocks of 2^{20}-bit segment)
9	**Linear Complexity** (lengths of shortest linear shift registers generating adjacent nonintersecting 500-bit blocks of the segment of the sequence)
10	**Serial** (frequencies of intersecting 16-, 15- and 14-bit blocks in the whole segment of the sequence, results in 2 statistics)
11	**Approximate Entropy** (frequencies of intersecting 10- and 11-bit blocks in the whole segment of the sequence)
12	**Cumulative Sums** (maximal deviation from 0 for ± 1 walk constructed by a segment of binary sequence, results in 2 statistics

Table 1: List of NIST Statistical Tests considered

In [1, Section 4.4] some notes were made on the results of the study on the dependencies between tests included in NIST Statistical Test Suite:

a) by the Kolmogorov–Smirnov test the empirical distributions of p-values were compared with the uniform distribution on [0, 1],

b) factor analysis of p-values was conducted for 161 types of statistics generated by different tests ,

c) for pseudorandom sequences of length 10^6 generated by the Blum–Blum–Shub generator the principal component analysis was conducted for the set of vectors $z_j = (z_{1j}, \ldots, z_{161j})$, $1 \le j \le m$, where $z_{ij} = \Phi^{-1}(p_{ij})$, p_{ij} is a p-value for statistics of j-th: computed for for j-th m-sequence, and Φ is the distribution function of the standard normal distribution,

d) correlation matrix for vectors z_j, $1 \le j \le m$, was computed.

The results of these investigations was formulated in [1, section 4.4] as

follows:

1) there is no large redundancy among the tests,

2) the degree of duplication among the tests seems to be very small.

Along with this summary the authors of [1] note that there exist dependencies between some tests.

The existence of dependencies between tests of the NIST package were studied in [3]. The author had applied all tests of the NIST package (except Random Excursions and Random Excursions Variant tests) to each of m segments consisting of n bits each, obtaining 162 p-values for each segment. Further for each segment he had computed arithmetic mean of these p-values and had compared the histogram constructed by n arithmetic means with the normal density approximating the sum of 162 independent random variables having uniform distribution on the interval (0,1). It appears that the histogram is wider than normal density; this is interpreted as the existence of positive correlations between p-values.

Remark. *In [1] and [3] the study of dependencies was concentrated on the global distributions of p-values. But decision made by each test depends on the closeness of p-value to 0, so it seems more natural to measure the dependence between tests in terms of left tail dependence between their statistics or p-values, or simply between test decisions. In our paper we use the last approach, namely, for each of m tested segments we compute the number of tests rejecting it and compare frequencies of these numbers in the sample of n segments with probabilities of binomial disribution Bin(m, q), where q is the rejection probability for the segment of equiprobable Bernoulli sequence. This binomial distribution corresponds the case when decisions of tests are independent.*

For some simple tests the characteristics of dependence between them may be computed exactly and studied theoretically. For example, in [5, theorems 10.3.3, 10.3.4] it was shown that if $s_0 = 0, s_1, \ldots, s_n$ is the walk on $\mathbb{Z}$ with steps ± 1, then the number of walks with $s_n = 2w - n$ and $z_n = \max_{0 \leqslant k \leqslant n} |s_k| \leqslant u$ is equal to

$$L(w, n-w, u) = \sum_{k \in \mathbb{Z}} \left(C_n^{w-2k(u+1)} - C_n^{w-(2k-1)(u+1)} \right). \qquad (1)$$

Analogous combinatorial formula may be found in [4] as a problem to Chapter 3. Since s_n and z_n correspond the statistics of the Frequency and Cumulative Sums tests, by means of (1) it is possible to compute the correlations

and joint distributions of these statistics. It was found that for symmetrical random walk the correlation of $|s_n|$ and z_n is approximately 0.85 (depending on n), and

$$\mathbf{P}\{|s_n| > a, z_n > b\} \approx \min\{\mathbf{P}\{|s_n| > a\}, \mathbf{P}\{z_n > b\}\}$$

if $\mathbf{P}\{|s_n| > a\}$ and $\mathbf{P}\{z_n > b\}$ are smaller than 0.01.

Our experiments had shown that correlations between statistics of Frequency and Cumulative Sums tests are larger than 0.8.

In [1] it is recommended to discontinue the Random Excursions and Random Excursions Variant tests if the trajectory of walk with steps ± 1 constructed by the tested binary segment have smaller than 500 returns to 0. In [4][Ch.III, § 7, Theorem 4] it is proved that for simple symmetrical random walk r-th return to 0 becomes at the moment n with the probability

$$\varphi_{r,n-r} = \frac{r}{(n-r)2^{n-r}} C_{n-r}^{n/2} = r\sqrt{\frac{2}{\pi n^3}} e^{-r^2/n} \left(1 + O\left(\frac{r}{n} + \frac{r^3}{n^2}\right)\right),$$

if $n \to \infty$, $r = \text{const}$. By means of this formula it is possible to check that for $n = 10^6$ the probability that the number of returns to 0 will be smaller than 500 is larger than 0.3, and for $n = 10^7$ it is larger than 0.1. So, tests based on Random Excursions do not generate any p-value for large part of tested segments, and it seems unreasonable to use such tests if the probability of error is chosen to be, for example, 0.01.

Finally, the Non-overlapping Template Matching Test generates 148 p-values and corresponding decisions. "It is highly likely" that among such large family of tests there exist dependent ones. For statistics of frequencies of numbers of tests rejecting the same segment some dependencies between 148 tests may hide dependencies between other tests, so we exclude this test from experiments described below. Dependencies between components of Non-overlapping Template Matching Test will be examined separately.

3 Description of our experiments

We have applied 12 tests from Table 1 to pseudorandom sequences of two types: a) sequences generated by a block cipher, b) sequences obtained from pairs of binary linear recurrent sequences.

The first type sequences were obtained by means of AES block cipher in the CFB (Cipher Feedback Block) mode (see Figure 1) with the zero starting plaintext block, IV=0xffffffffffffffffffffffffffff80, zero key for the sequence

denoted by AES_0, and key 0x2b7e151628aed2a6abf7158809cf4f3c for the sequence denoted by AES_1.

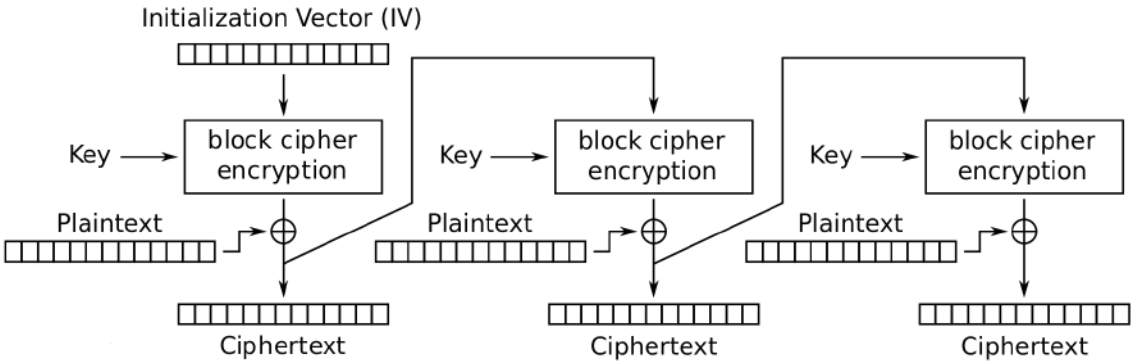


Figure 1:

The second type sequences were obtained by quasirandom partitioning two binary linear recurrent sequences into segments and concatenating these segments in alternating manner. We use linear recurrent sequences over GF(2) having maximal periods and characteristic polynomials

$$\begin{aligned} f(x) &= x^{43} + x^{27} + x^{22} + x^5 + 1, \\ g(x) &= x^{63} + x + 1, \\ h(x) &= x^{33} + x^{26} + x^{22} + x^{21} + x^3 + x + 1, \\ m(x) &= x^{33} + x^{13} + 1, \\ q(x) &= x^{63} + x^{60} + x^{22} + x^{18} + x^8 + x^5 + 1, \\ u(x) &= x^{64} + x^{61} + x^{56} + x^{31} + x^{28} + x^{23} + 1. \end{aligned}$$

Initial states of recurrent sequences have only one nonzero element which was the most significant bit.

The recurrent sequences were divided into adjacent segments of pseudorandom lengths according to the following rule:

– the first segment $(x_1, \ldots, x_{L_1^*})$ of the first recurrent sequence $\{x_i\}_{i=0}^{2^{n_1}-1}$ has length $L_1^* = 64$,

– the first segment $(y_1, \ldots, y_{L_2^*})$ of the second recurrent sequence $\{y_i\}_{i=0}^{2^{n_2}-1}$ has length

$$L_2^* = L_1^* + 2^6 x_{L_1^*-6} + 2^5 x_{L_1^*-5} + 2^4 x_{L_1^*-4} + 2^3 x_{L_1^*-3} + 2^2 x_{L_1^*-2} + 2 x_{L_1^*-1} + x_{L_1^*},$$

– the second segment $(x_{L_1^*+1}, x_{L_1^*+2}, \ldots, x_{L_1^*+L_3^*})$ of the first sequence has length

$$L_3^* = L_1^* + 2^6 y_{L_2^*-6} + 2^5 y_{L_2^*-5} + 2^4 y_{L_2^*-4} + 2^3 y_{L_2^*-3} + 2^2 y_{L_2^*-2} + 2 y_{L_2^*-1} + y_{L_2^*},$$

– and so on.

Tested sequences $\{w_1, w_2, \ldots\}$ of the second type have the form

$$\{x_1, \ldots, x_{L_1^*}, y_1, \ldots, y_{L_2^*}, x_{L_1^*+1}, \ldots, x_{L_1^*+L_1^*}, y_{L_2^*+1}, \ldots\}.$$

Denote tested sequences by corresponding pairs of characteristic polynomials:

MixL$_1$: $h(x)$, $m(y)$,

MixL$_2$: $f(x)$, $g(y)$,

MixL$_3$: $f(x)$, $q(y)$,

MixL$_4$: $u(x)$, $q(y)$.

Each tested sequence was divided into $s = 10000$ segments of length $l = 2^{20} = 1.048.576$ each (this length corresponds the NIST package recommendations). Each segment was tested by all 12 tests from Table 1.

The application of 12 tests to each segment results in 14 different p-values (each of tests 10 and 12 results in two p-values). The critical level of p-value was chosen as 0.01: if the p-value does not exceed 0.01, then test rejects the hypothesis H_0 on the randomness and equiprobability of elements of tested segment. For each segment we compute the number of p-values not exceeding 0.01, and for each tested sequence compute frequencies $\eta_0, \eta_1, \ldots, \eta_{14}$ of these numbers over sample of 10000 volume.

Under the hypothesis on the independence of tests for each segment of "ideal" Bernoulli sequence the number of tests giving p-value not exceeding 0.01 should have the binomial distribution with the parameters (14, 0.01). Since for each sequence we have considered $s = 10000$ segments, then for independent tests

$$\mathbf{E}\eta_k = 10000 C_{14}^k (0.01)^k (0.99)^{14-k}, \quad k = 0, 1, \ldots, 14. \tag{2}$$

To test the hypothesis
H_{14}: $(\eta_0, \eta_1, \ldots, \eta_{14})$ corresponds the sample of 10000 random variables with distribution Bin(14,0.01)
we use the Pearson statistics for $(\eta_0, \eta_1, \eta_2 + \ldots + \eta_{14})$, joining all frequencies exceeding 1 in one class (to make the mean value of outcomes in each class at least 5):

$$Pear = \sum_{k=0}^{1} \frac{(\eta_k - \mathbf{E}\eta_k)^2}{\mathbf{E}\eta_k} + \frac{\left(\sum_{k=2}^{14} \eta_k - \sum_{k=2}^{14} \mathbf{E}\eta_k\right)^2}{\sum_{k=2}^{14} \mathbf{E}\eta_k}. \tag{3}$$

Under hypothesis on the independence of tests the distribution of this statistics should be close to the chi-square distribution with 2 degrees of freedom having mean 2 and variance 4.

Table 2 contains frequencies $\eta_0, \eta_1, \ldots, \eta_9$ for samples of 10000 segments of sequences AES_0, AES_1, MixL_1, MixL_2, MixL_3, MixL_4. Last column contains values $\mathbf{E}\eta_k$ for the binomial distribution with the parameters (14, 0.01) (for the case of independent tests). Last row contains values of Pearson statistics (3).

k	AES_0	AES_1	MixL_1	MixL_2	MixL_3	MixL_4	$\mathbf{E}\eta_k$
0	8768	8826	8729	5602	8816	8800	8687.46
1	1012	992	1103	1906	983	1023	1228.53
2	125	100	100	978	128	108	80.66
3	83	71	58	734	63	59	3.26
4	12	9	9	496	9	8	0.09
5	0	1	1	158	1	2	0.018
6	0	1	0	59	0	0	$2.7 \cdot 10^{-5}$
7	0	0	0	29	0	0	$3.2 \cdot 10^{-7}$
8	0	0	0	21	0	0	$2.8 \cdot 10^{-9}$
9	0	0	0	17	0	0	$1.9 \cdot 10^{-11}$
Total	10000	10000	10000	10000	10000	10000	10000
Pear	259	162	97	70487	214	139	

Table 2: Numbers of segments resulting in k (out of 14) p-values smaller than 0.01 and values of Pearson statistics for truncated binomial distribution

Table 2 shows that the hypothesis on the independence of tests should be rejected.

Note that frequencies corresponding the sequence MixL_2 show that it is rejected by many tests. It may be a consequence of bad mixing properties of linear recurrent sequence with the characteristic polynomial $g(x)$.

More detailed analysis of results of experiments have shown that p-values corresponding the Frequency, the Cumulative Sums and the Serial tests are simultaneously smaller than 0.01 with probabilities larger than it should be for independent tests.

If we exclude from the experimental results one of the statistics of the Serial test (number 10 in Table 1) and both statistics of the Cumulative Sums test (number 12 in Table 1), then the remaining tests became more similar to independent ones from the viewpoint of the distribution of the number of tests simultaneously rejecting the same segments. In this case we use equations (2) and (3) with 11 instead of 14. Corresponding results are contained in Table 3.

Note that after exclusion of three p-values the correpondence with hy-

k	AES_0	AES_1	MixL_1	MixL_2	MixL_3	MixL_4	$\mathbf{E}\eta_k$
0	8878	8920	8835	5650	8919	8905	8953.38
1	1043	1016	1097	2189	1012	1025	994.82
2	78	61	64	1212	66	66	50.24
3	1	2	4	638	3	4	1.52
4	0	1	0	202	0	0	0.03
5	0	0	0	67	0	0	$4 \cdot 10^{-4}$
6	0	0	0	25	0	0	$4.4 \cdot 10^{-6}$
7	0	0	0	13	0	0	$3.2 \cdot 10^{-8}$
8	0	0	0	4	0	0	$1.6 \cdot 10^{-10}$
9	0	0	0	0	0	0	$5.4 \cdot 10^{-13}$
Total	10000	10000	10000	10000	10000	10000	10000
Pear	17.25	3.45	17.12	88539	6.14	7.57	

Table 3: Numbers of segments resulting in k (out of 11) p-values smaller than 0.01 and values of Pearson statistics for truncated binomial distribution

pothesis on independence became better, but nevertheless is far from normal. If χ^2_2 denotes random variable having the chi-square distribution with 2 degrees of freedom (i. e. exponential distribution with parameter $\frac{1}{2}$ and mean 2), then $\mathbf{P}\{\chi^2_2 \geqslant 17.12\} \approx 0.0002$, $\mathbf{P}\{\chi^2_2 \geqslant 3.45\} \approx 0.178$, $\mathbf{P}\{\chi^2_2 \geqslant 6.14\} \approx 0.046$, $\mathbf{P}\{\chi^2_2 \geqslant 7.57\} \approx 0.023$. Note also that the sequence AES_0 was obtained with zero key, sequence MixL_1 was constructed from sequences having characteristic polynomials of the same degree 33, sequence MixL_2 contains segments of recurrent sequence with three-term characteristic polynomial. So, sequences of these types may have properties which are reflected by some tests.

4 Conclusion

The hypothesis on the independence of tests included in the NIST Statistical Test Suite was rejected on the base of computer experiments by means of statistics of frequencies of segments simultaneously rejected by given number of tests. It is shown that from the viewpoint of this statistics tests from some subset of the set of NIST tests look more independent than tests of the whole set.

It is shown also that pseudorandom sequences constructed by alternating concatenation of segments of two linear recurrent sequences over GF(2) with non-regularly varying lengths do not rejected by the NIST tests if these recurrent sequences are not too simple.

References

[1] 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.

[2] L'Ecuyer P., Simard R., "TestU01", 2013, 214 `http://simul.iro.umontreal.ca/testu01/guideshorttestu01.pdf` pp.

[3] Iwasaki A., "Analysis of NIST SP800-22 focusing on randomness of each sequence", 2017, 8 pp., arXiv: 1710.01441

[4] Feller W., *An Introduction to Probability Theory and its Applications. Vol. 1. 3rd edition*, N.-Y. e.a.: J.Wiley & Sons, 1970, 528 pp.

[5] Krattenthaler C., "Lattice path enumeration", *Handbook of Enumerative Combinatorics*, Boca Raton–London–New York: CRC Press, 2015, 589-678

[6] Zubkov A. M., Serov A. A., "Testing the NIST Statistical Test Suite on artificial pseudo-random sequences", *Matematicheskie Voprosy Kriptografii*, **10**:2 (2019), 89–96.

On Time-Adaptive Statistical Testing for Random Number Generators

Boris Ryabko[1,2]

[1]Institute of Computational Technologies of SB RAS, Novosibirsk, Russia
[2]Novosibirsk State University, Russia
boris@ryabko.net

Abstract

Random number generators and statistical tests for checking them are widely used in data protection systems and are intensively researched in modern cryptography. Nowadays there are hundreds of RNG statistical tests that are often combined into so-called batteries, each containing from a dozen to more than one hundred tests. We propose an adaptive way to use batteries (and other sets) of tests, which requires less time but, in a certain sense, preserves the power of the original battery. We call this method time-adaptive battery of tests.

Keywords: cryptographic randomness, statistical test, randomness testing, random number generators, adaptive statistical test, battery of tests.

1 Introduction

Random number generators (RNG) and pseudo-random number generators (PRNG) as well as statistical tests for checking them are widely used in data protection systems. RNGs are based on physical sources, while pseudo-random numbers are generated by computers. The goal of RNG and PRNG is to generate sequences of binary digits, which are distributed as a result of throwing an "honest" coin, or, more precisely, obey the Bernoulli distribution with parameters $(1/2, 1/2)$. As a rule, for practically used RNG and PRNG this property is verified experimentally with the help of statistical tests developed for this purpose.

Currently, there are more than one hundred applicable statistical tests, as well as dozens RNGs based on different physical processes, and an even greater number of PRNGs based on different mathematical algorithms; see for review [1, 2, 3]. Informally, an ideal RNG should generate sequences that pass all tests. In practice, especially in cryptographic applications, this requirement is formulated as follows: an RNG must pass a so-called battery of statistical tests, that is, some fixed set of tests. When a battery is applied,

each test in the test battery is applied separately to the RNG. Among these batteries, we mention the National Institute of Standards and Technology (NIST) battery of 15 tests which designed for data protection systems [4], several batteries proposed by L'Ecuyer and Simard [2], which contain from 10 to 106 tests and many others (see for review [1, 2, 5]). In addition, these batteries contain many tests that can be used with different values of the parameters, potentially increasing the total number of tests in the battery. Note that practically used RNG should be tested from time to time like any physical equipment, and therefore these test batteries should be used continuously. It is worth noting that most of the research is related to cryptography applications, see, for example, [4, 5, 6].

How to evaluate large batteries of tests? On the one hand, the larger the test battery, the more likely it is to find flaws in the tested RNG. On the other hand, the larger the battery, the more time is required for testing. (Thus, L'Ecuyer and Simard [2] remark the need for small batteries to increase computational efficiency.) Another view is as follows: in reality, the time available to study any RNG is limited. Given a certain time budget, one can either use more tests and relatively short sequences generated by the RNG, or use fewer tests, but longer sequences and, in turn, this gives more chances to find deviations of randomness of the considered RNG.

In order to reduce this trade-off, we propose a time-adaptive testing of RNG, in which, informally speaking, first all the tests are executed on relatively short sequences generated by the RNG, and then a few "promising" tests are applied for the final testing. Of course, the key question here is which tests are promising. For example, if a battery of two tests is applied to (relatively short) sequences of the same length, it can be assumed that the smaller the p-value, the more promising the test. But a more complicated situation may arise when we have to compare two tests that were applied to sequences of different lengths (for example, the first test was applied to a sequence of length l_1, and the second to a sequence of length of l_2, $l_1 \neq l_2$). We show that if our goal is to choose the most powerful test, then a good strategy is to choose the test i for which the ratio $-\log \pi_i / l_i$ is maximum, where π_i is p-value of the i-th test. This recommendation is based on the following theorem: if an RNG can be modelled by a stationary ergodic source, the value $-log\, \pi(x_1 x_2 ... x_n)/n$ goes to $1 - h$, if n grows, where $x_1 x_2 ...$ is a generated sequence, $\pi(\,)$ is the p-value of the most powerful test, h is the limit Shannon entropy of the stationary ergodic source. This theorem plays an important rule in the suggested time-adaptive scheme and will be described in the first part of the paper, whereas the time-adaptive testing will

be described afterwords. The description will be illustrated by experiments with the battery Rabbit from [2].

2 Hypothesis testing and properties of pi-values

2.1 Notation

We consider RNG which generates a sequence of letters $x = x_1x_2\ ...x_n$, $n \geq 1$, from a finite alphabet $\{0,1\}^n$. Two statistical hypotheses are considered: $H_0 = \{x \quad obeys\ the\ uniform\ distribution\ (\mu_U)\ on\ \{0,1\}^n\ \}$, and the alternative hypothesis $H_1 = \bar{H}_0$, that is, H_1 is the negation of H_0. It is a particular case of the so-called goodness-of-fit problem, and any test for it is called a test of fit, see [11]. Let t be a test. Then, by definition, the significance level α equals the probability of the Type I error, $\alpha \in (0,1)$. Denote a critical region of the test t for the significance level α by $C_t(\alpha)$ and let $\bar{C}_t(\alpha) = \{0,1\}^n \setminus C_t(\alpha)$. (Recall that Type I error occurs if H_0 is true and is rejected. Type II error occurs if H_1 is true, but H_0 is accepted. Besides, for a certain $x = x_1x_2...x_n$ H_0 is rejected if and only if $x \in C_t(\alpha)$.)

Suppose that H_1 is true, and the investigated sequence $x = x_1x_2...x_n$ is generated by an (unknown) source ν. By definition, a test t is consistent, if for any significance level $\alpha \in (0,1)$ the probability of Type II error goes to 0, that is

$$\lim_{n\to\infty} \nu(\bar{C}_t(\alpha)) = 0\,. \tag{1}$$

Suppose, that H_1 is true and the sequences $x \in \{0,1\}^n$ obey a certain distribution ν. It is well-known in mathematical statistics that the optimal test (Neyman-Pearson or NP test) is described by the Neyman-Pearson lemma and the critical region of this test is defined as follows:

$$C_{NP}(\alpha) = \{x :\ \mu_U(x)/\nu(x) \leq \lambda_\alpha\}\,,$$

where $\alpha \in (0,1)$ is the significance level and the constant λ_α is chosen in such a way that $\mu_U(C_{NP}(\alpha)) = \alpha$, see [11]. (We did not take into account that the set $\{0,1\}^n$ is finite. Strictly speaking, in such a case a randomized test should be used, but in what follows we will consider asymptotic behaviour of tests for large n, and this effect will be negligible). Note that, by definition, $\mu_U(x) = 2^{-n}$ for any $x \in \{0,1\}^n$.

2.2 The p-value and its properties

The notion of the critical region is connected with the so-called p-value, which we define for the NP-test by the following equation:

$$\pi_{NP}(x) = \mu_U\{y : \nu(y) > \nu(x)\} = |\{y : \nu(y) > \nu(x)\}|/2^n\,. \tag{2}$$

Informally, $\pi_{NP}(x)$ is the probability to meet a random point y which is worse than the observed when considering the null hypothesis.

The NP-test is optimal in the sense that its probability of a Type II error is minimal, but when testing an RNG the alternative distribution is unknown, and, hence, different tests are necessary. Let us consider a certain statistic τ (that is, a function on $\{0,1\}^n$), and define the p-value for this τ and x as follows:

$$\pi_\tau(x) = \mu_U\{y : \tau(y) > \tau(x)\} = |\{y : \tau(y) > \tau(x)\}|/2^n\,. \tag{3}$$

(Note, that the definition π_{NP} in (2) corresponds to this equation if the value $\nu(x)$ is considered as a statistic, i.e. $\tau(x) = \nu(x)$).

2.3 The p-value and Shannon entropy

It turns out that there exist such tests whose asymptotic behaviour is close to that of the NP-test for any (unknown) stationary ergodic source ν, see [7]. Those tests are based on so-called universal codes (or data-compressors) and are described in [8, 9], where it is shown that they are consistent. We describe those tests in Appendix 1 and show that they are asymptotically optimal. The following theorem describes the asymptotic behaviour of p-values for stationary ergodic sources for NP test and the mentioned above tests which are based on universal codes (see Appendix 1). We use this theorem as the theoretical basis for adaptive statistical testing developed in this paper.

Theorem 1. *i) If ν is a stationary ergodic measure, then, with probability 1,*

$$\lim_{n\to\infty} -\frac{1}{n}\log \pi_{NP}(x) = 1 - h(\nu)\,, \tag{4}$$

where $h(\nu)$ is the Shannon entropy of ν, see for definition [10].

ii) There exists such a statistic τ that for any stationary ergodic measure ν, with probability 1,

$$\lim_{n\to\infty} -\frac{1}{n}\log \pi_\tau(x) = 1 - h(\nu)\,, \tag{5}$$

where p-values π_{NP} and π_τ are defined in (2) and (3), correspondingly.

The test of fit τ is described in Appendix 1, the proof of the theorem is given in Appendix 2, but here we note that this theorem gives some idea of the relation between the Shannon entropy of the (unknown) process ν and the required sample size. Indeed, suppose that the NP test is used and the desired significance level is α. Then, we can see that (asymptotically) α should be larger than $\pi_{NP}(x)$ and from (4) we obtain $n > -\log\alpha/(1-h(\nu))$ (for the most powerful test). It is known that the Shannon entropy is 1 if and only if ν is a uniform measure μ_u. Therefore, in a certain sense, the difference $1-h(\nu)$ estimates the distance between the distributions, and the last inequality shows that the sample size becomes infinite if ν approaches a uniform distribution.

3 Time-adaptive statistical tests and their experimental investigation

3.1 Batteries of tests

Let us consider a situation where the randomness testing is performed by conducting a battery of statistical tests for randomness. Suppose that the battery contains s tests and α_i is the significance level of $i-$th test, $i = 1, ..., s$. If the battery is applied in such a way that the hypothesis H_0 is rejected when at least one test in the battery rejects it, then the significance level α of this battery satisfies the following inequality:

$$\alpha \leq \sum_{i=1}^{s} \alpha_i \,. \tag{6}$$

If all the tests in the battery are independent, then the following equation is valid: $\alpha = 1 - \prod_{i=1}^{s}(1-\alpha_i)$. Clearly, the upper bound (6) is true for this case and $1-\prod_{i=1}^{s}(1-\alpha_i)$ is close to $\sum_{i=1}^{s}\alpha_i$, if each α_i is much smaller than $1/s$. That is why we will use the estimate (6) below.

We have considered a scenario in which a test is applied to a single sequence generated by an RNG, and then the researcher makes a decision on the RNG based on the test results. Another possibility that has been considered by several authors, e.g. [2, 4], is to use the following two-step procedure for testing RNGs. The idea is to generate r sequences $x^1, x^2, ..., x^r$ and apply one test (say, τ) to each of them independently. Then apply another test to the received data $\tau(x^1), \tau(x^2), ..., \tau(x^r)$ (as a rule, those values are converted into a sequence of corresponding p-values, and then the hypothesis of the uniform distribution of those p-values is tested). Then this procedure is

repeated for the second test in the battery, and so on. The final decision is made on the basis of the results obtained. We do not consider this two-step procedure in detail, but note that time-adaptive testing can be applied in this situation, too.

3.2 The scheme of the time-adaptive testing

Let there be an RNG which generates binary sequences, and a battery of s tests with statistics $\tau_1, \tau_2, ..., \tau_s$. In addition, suppose that the total available testing time is limited to a certain amount T and the level of significance is $\alpha \in (0, 1)$.

When the time-adaptive testing is applied, all the calculations are separated into a preliminary stage and a final one. The result of the preliminary stage is the list of values

$$\gamma_1 = \frac{-\log \pi_{\tau_1}(x_1^1 x_2^1 ... x_{n_1}^1)}{n_1}, \gamma_2 = \frac{-\log \pi_{\tau_2}(x_1^2 x_2^2 ... x_{n_2}^2)}{n_2}$$

$$, ..., \ \gamma_s = \frac{-\log \pi_{\tau_s}(x_1^s x_2^s ... x_{n_s}^s)}{n_s}\,, \tag{7}$$

where the sequences $x_1^1 x_2^1 ... x_{n_1}^1$, ..., $x_1^s x_2^s ... x_{n_s}^s$ may have common parts (for example, the first sequence may be the prefix of the second, etc.). Then, taking into account the values (7), it is possible to choose some tests from the battery and apply them to the longer sequence, calculate new values γ, and so on. When the preliminary stage is carried out, several tests from the battery should be chosen for the next stage.

The final stage is as follows. First, we divide the significance level α into $\alpha_1, \alpha_2, ..., \alpha_k$ in such a way that $\sum_{i=1}^{k} \alpha_i = \alpha$. Then, we obtain new sequence(s) $y_1^1 y_2^1 ... y_{m_1}^1, ..., y_1^k y_2^k ... y_{m_k}^k$, which may have common parts, but are independent of $x_1^1 x_2^1 ... x_{n_1}^1$, ..., $x_1^s x_2^s ... x_{n_s}^s$ and calculate

$$\pi_{\tau_{i_1}}(y_1^1 y_2^1 ... y_{m_1}^1), ..., \pi_{\tau_{i_k}}(y_1^k y_2^k ... y_{m_k}^k)\,. \tag{8}$$

The hypothesis H_0 will be accepted, if $\pi_{\tau_{i_j}}(y_1^j y_2^j ... y_{m_j}^j) > \alpha_j$ for all $j = 1, ..., k$. Otherwise, H_0 is rejected. The parameters of the test should be chosen in such a way that the total time of calculation is not grater than the given limit T.

Claim. The significance level of the described time-adaptive test is not larger than α.

Indeed, the sequences $y_1^1 y_2^1 ... y_{m_1}^1, ..., y_1^k y_2^k ... y_{m_k}^k$ and $x_1^1 x_2^1 ... x_{n_1}^1$, ..., $x_1^s x_2^s ... x_{n_s}^s$ are independent and, hence, the results of the final stage

does not depend on the preliminary one. When the test battery $\tau_{i_1}, \tau_{i_2}, ..., \tau_{i_k}$ is applied, the significance level of τ_{i_j} equals α_j and the significance level of the battery equals $\sum_{i=1}^{k} \alpha_i$. From (6) we can see that the significance level of the battery (and, hence, of the described testing) is not grater than α.

Comment. The length of the sequences may depend on the speed of tests. For example, it can be done as follows: let v_i be the speed per bit of the test τ_i, $i = 1, ..., s$. One possible way to take into account the speed difference is to calculate

$$\hat{\gamma}_i = \frac{-\log \pi_{\tau_i}(x_1^i x_2^i ... x_{n_i}^i)}{n_i / v_i}, \quad i = 1, ..., s,$$

instead of (7) and similar expressions.

3.3 The experiments

We carried out some experiments with the time-adaptive test basing on the battery Rabbit from [2], which contains 26 tests. Let us first describe the choice of the RNG for our experiments. Nowadays there are many "bad" PRNGs and "good" ones. In other words, the output sequences of some known PRNGs have some deviations from randomness, which are quite easy to detect with many known tests, while other PRNGs do not have deviations that can be detected by known tests [2]. So, we need to have some families of RNGs with such deviations from randomness that they can be detected only for quite large output sequences. To do this, we take a good generator MRG32k3a and a bad one LCG from [2], generate sequences $g_1 g_2 ...$ and $b_1 b_2 ...$ by these two generators and then prepared a "mixed" sequence $m_1 m_2 ...$ in such a way that

$$m_i = \begin{cases} g_i & \text{if } i \mod D \neq 0 \\ b_i & \text{if } i \mod D = 0 \end{cases} \tag{9}$$

where D is a parameter.

The time-adaptive testing was organised as follows: during the preliminary stage we first generated a file $m_1 m_2 ... m_{l_1}$ with $l_1 = 2\,000\,000$ bytes, tested it by 25 tests from the Rabbit battery and calculated the values (7) with $\log \equiv \log_2$, see the left part of Table 1. (This battery contains 26 tests, but one of them cannot be applied to such a short sequence.) Then we chose 5 tests with the biggest value $-log\, \pi_{t_i}(m_1 ... m_{l_i})/l_1$ (let they be $t_{i_1}, ..., t_{i_5}$), generated a sequence $m_1 ... m_{l_2}$ with $l_2 = 6\,000\,000$ bytes and applied the tests $t_{i_1}, ..., t_{i_5}$ for testing this sequence (see the example in the right part of Table

1). After that we found a test t_f for which

$$-log\pi_{t_f}/l_f = \max_{r=1,...,25;\, j=i_1...i_5} \{-log\pi_r(m_1...m_{l_1})/l_1, -log\pi_j(m_1...m_{l_2})/l_2\}\,.$$

(In other words, for t_f the value $-log\pi_r(m_1...m_{l_k})/l_k$ is maximal for $k = 1, 2$ and all r (see the Table 1). The preliminary stage was finished. Then, during the second stage, we generated a 40 000 000 byte sequence, and applied the test t_f to it. If the obtained p-value was less than 0.001, the hypothesis H_0 was rejected. (Note that the sequence length $l_1 = 2\,000\,000$ and $l_2 = 6\,000\,000$ are 5% and 15% from the final length of 40 000 000 bytes. So, the total length of the sequences tested by all the tests during the preliminary stage is $25 \times 0.05 + 5 \times 0.15 = 2$ the final length, i.e. $2 \times 40\,000\,000$. On the other hand, if one applies the battery Rabbit to the sequence of the same length, the total length of investigated sequences is $25 \times 40\,000\,000$, i.e. 8,33 times more.

Let us consider one example in detail, taking $D = 2$ in (9).

Table 1 contains the results of all the calculations carried out during the preliminary stage. So, we can see that the value $-\log_2 \pi)/l$ is maximal for the test $t13$. Hence, at the final stage we applied the test $t13$ to the new 40 000 000-byte sequence. It turned out that $\pi_{t13} = 2.9\ 10^{-26}$ and, hence, H_0 is rejected. Besides, we estimated time of all calculation (during both stages).

After that, we conducted an additional experiment to get the full picture. Namely, we calculated p-values for all tests and for the same 40 000 000-byte sequence and the estimated the total time of calculations. It turned out that the p-values of the two tests were less than 0.001. Namely, $\pi_{t13} = 2.9\ 10^{-26}$, $\pi_{t22} = 1.1\ 10^{-6}$. Besides, we estimated time of calculations for all experiments. So, the described time-adaptive testing revealed one of the two most powerful tests, and the time used is 8 times less. We carried out similar experiments 20 times for $d = 2, 3, 4$ (in (9)) with different good and bad generators from [2]. Besides, we investigated several modifications of the considered scheme. In particular, we considered a case where during the preliminary stage we, as before, first chose 5 the best tests and them two of the best tests for the finale stage (instead of one, as in the experiment above). It turned out, that in all cases considered the battery Rabbit rejects H_0 and the time-adaptive testing rejected H_0, too.

4 Conclusion

First of all, we note that the proposed time-adaptive testing does not suggest exact values of numerous parameters. Among these parameters, we

test	length (l) (bytes)	p-value (π)	$-\log_2 \pi/l$	length (l) (bytes)	p-value	$-\log_2 \pi/l$
t1	$2\,10^6$	0.42	$6.3\ 10^{-7}$			
t2	$2\,10^6$	0.37	$7.3\ 10^{-7}$			
t3	$2\,10^6$	0.028	$26\ 10^{-7}$	$6\,10^6$	0,23	$3.6\ 10^{-7}$
t4	$2\,10^6$	0.78	$1.8\,10^{-7}$			
t5	$2\,10^6$	0.4	$6.5\,10^{-7}$			
t6	$2\,10^6$	0.37	$7.2\,10^{-7}$			
t7	$2\,10^6$	0.059	$20\,10^{-7}$			
t8	$2\,10^6$	0.026	$26\,10^{-7}$	$6\,10^6$	0.0037	$26\ 10^{-7}$
t9	$2\,10^6$	0.72	$2.4\,10^{-7}$			
t10	$2\,10^6$	0.72	$2.4\,10^{-7}$			
t11	$2\,10^6$	0.63	$3.3\,10^{-7}$			
t12	$2\,10^6$	0.74	$2.2\,10^{-7}$			
t13	$2\,10^6$	0.021	$28\,10^{-7}$	$6\,10^6$	0.0028	$14\,10^{-7}$
t14	$2\,10^6$	0.42	$6.2\,10^{-7}$			
t15	$2\,10^6$	0.9	$0.74\,10^{-7}$			
t16	$2\,10^6$	0.087	$18\,10^{-7}$			
t17	$2\,10^6$	0.72	$2.3\,10^{-7}$			
t18	$2\,10^6$	0.58	$3.9\,10^{-7}$			
t19	$2\,10^6$	0.89	$0.81\,10^{-7}$			
t20	$2\,10^6$	0.51	$4.9\,10^{-7}$			
t21	$2\,10^6$	0.047	$22\,10^{-7}$	$6\,10^6$	0.73	$0.76\,10^{-7}$
t22	$2\,10^6$	0.47	$0.47\,10^{-7}$			
t23	$2\,10^6$	0.18	$12\,10^{-7}$			
t24	$2\,10^6$	0.14	$14\,10^{-7}$			
t25	$2\,10^6$	0.024	$27\,10^{-7}$	$6\,10^6$	0.05	$7.2\,10^{-7}$

Table 1: Time-adaptive testing. Preliminary stage.

note the number of steps at the preliminary stage (in the considered example there were two such steps: selecting five tests and then one), the number of tests compared in one step, the length of the tested sequences, the rule for choosing tests at different stages, etc. The problem of choosing the parameters may be considered a problem of multidimensional optimization. There are many methods available for solving such problems (for example, neural networks and other AI algorithms), and some of them can be used along with the time-adaptive testing.

As far as we know, no one has applied adaptive methods for testing randomness, but there are several well-known approaches that can be considered as steps in this direction. For example, L'Ecuyer and Simard recommend several batteries of different sizes that require different times (and the investigator may use them depending on how much time he has) [2]. Another

popular battery recommended for cryptographic applications also has some parameters that allow one to adjust the testing time [4].

We believe that the proposed approach makes it possible to investigate and optimize time-adaptive testing.

5 Appendix 1. Consistent tests based on universal codes

The considered tests are based on so-called universal codes, that is why we first briefly describe them. For any integer m a code ϕ is defined as such a map from the set of m-letter words to the set of all binary words that for any m-letter u and v $\phi(u) \neq \phi(u)$. This property gives a possibility to uniquely decode. (More formally, ϕ is injective mapping from $\{0,1\}^m$ to $\{0,1\}^*$, where $\{0,1\}^* = \bigcup_{i=1}^{\infty}\{0,1\}^i$.) We will consider so-called universal codes which have the two following properties:

$$\forall\, m > 0 \qquad \sum_{u\in\{0,1\}^m} 2^{-|\phi(u)|} = 1 \tag{10}$$

and for any stationary ergodic ν defined on the set of all infinite binary words $x = x_1x_2...$, with probability one

$$\lim_{n\to\infty} \frac{1}{n}|\phi(x_1x_2...x_n)|/n = h(\nu) \tag{11}$$

where $h(\nu)$ is the Shannon entropy of ν. Such code exist, see [10]. Note, that a goal of codes is to " compress " sequences, i.e. make an average length of the codeword $\phi(x_1x_2...x_n)$ as small as possible. The second property (11) shows that the universal codes are asymptotically optimal, because the Shannon entropy is a low bound of the length of the compressed sequence (per letter), see [10].

Let us back to considered problem of hypothesis testing. Suppose, it is known that a sample sequence $x = x_1x_2...$ was generated by stationary ergodic source and, as before, we consider the same H_0 against the same H_1. Let ϕ be a universal code. The following test is suggested in [8]:

If the length $|\phi(x_1...x_n) \leq n - \log_2 \alpha$ then H_0 is rejected, otherwise accepted. Here, as before, α is the significance level, $|\phi(x_1...x_n)|$ is the length of encoded ("compressed") sequence. We denote this test by T_ϕ and its statistic by τ_ϕ, i.e.

$$\tau_\phi(x_1...x_n) = n - |\phi(x_1...x_n)|\,. \tag{12}$$

The following theorem is proven in [8, 9]:

Theorem 2. *For each stationary ergodic ν, alpha $\in (0,1)$ and a universal code ϕ, with probability 1 the Type I error of the described test is not larger than α and the Type II error goes to 0, when $n \to \infty$.*

6 Appendix 2. Proofs

Proof of Theorem 1. The known Shannon-McMillan-Breiman (SMB) theorem claims that for the stationary ergodic source ν and any $\epsilon > 0, \delta > 0$ there exists such n' that

$$\nu\{x : x \in \{0,1\}^n \ \& \ h(\nu) - \epsilon < -\frac{1}{n}\log\nu(x) < h(\nu) + \epsilon \ \} > 1 - \delta \quad (13)$$

for $n > n'$, see [10]. From this we obtain

$$\nu\{x : x \in \{0,1\}^n \ \& \ 2^{-n(h(\nu)-\epsilon)} > \nu(x) > 2^{-n(h(\nu)+\epsilon)}\} > 1 - \delta \quad (14)$$

for $n > n'$. It will be convenient to define

$$\Phi_{\epsilon,\delta,n} = \{x : x \in \{0,1\}^n \ \& \ h(\nu) - \epsilon < -\frac{1}{n}\log\nu(x) < h(\nu) + \epsilon \ \} \quad (15)$$

From this definition and (14) we obtain

$$(1-\delta)\, 2^{n(h(\nu)-\epsilon)} \le |\Phi_{\epsilon,\delta,n}| \le 2^{n(h(\nu)+\epsilon)}\,. \quad (16)$$

For any $x \in \Phi_{\epsilon,\delta,n}$ define

$$\Lambda_x = \{y : \nu(y) > \nu(x) \ \} \bigcap \Phi_{\epsilon,\delta,n}\,. \quad (17)$$

Note that, by definition, $|\Lambda_x| \le |\Phi_{\epsilon,\delta,n}|$ and from (16) we obtain

$$|\Lambda_x| \le 2^{n(h(\nu)+\epsilon)}\,. \quad (18)$$

For any $\rho \in (0,1)$ we define $\Psi_\rho \subset \Phi_{\epsilon,\delta,n}$ such that

$$\nu(\Psi_\rho) = \rho \ \& \ \forall u \in \Psi_\rho \forall v \in (\Phi_{\epsilon,\delta,n} \setminus \Psi_\rho) \implies \nu(u)) \ge \nu(v)\,. \quad (19)$$

(That is, Ψ_ρ contains the most probable words whose total probability equals ρ.) Let us consider any $x \in (\Phi_{\epsilon,\delta,n} \setminus \Psi_\rho)$. Taking into account the definition (19) and (16) we can see that for this x

$$|\Lambda_x| \ge \rho|\Phi_{\epsilon,\delta,n}| \ge \rho(1-\delta)2^{n(h(\nu)-\epsilon)}\,. \quad (20)$$

So, from this inequality and (18) we obtain

$$\rho(1-\delta)2^{n(h(\nu)-\epsilon)} \le |\Lambda_x| \le 2^{n(h(\nu)+\epsilon)}\,. \quad (21)$$

From equation (14), (15) and (19) we can see that $\nu(\Phi_{\epsilon,\delta,n} \setminus \Psi_\rho) \geq (1-\delta)(1-\rho)$. Taking into account (21) and this inequality, we can see that

$$\nu\{x : x \in \{0,1\}^n \& h(\nu) - \epsilon - \log(\rho(1-\delta))/n \leq \log|\Lambda_x|/n \leq h(\nu) + \epsilon\}$$
$$\geq (1-\delta)(1-\rho). \tag{22}$$

From the definition (2) of $\pi_{NP}(x)$ and the definition (17) of Λ_x, we can see that $\pi_{NP}(x) = |\Lambda_x|/2^n$. Taking into account this equation and (22) we obtain the following:

$$\nu\{x : x \in \{0,1\}^n \,\&\, 1 - (h(\nu) - \epsilon - \log(\rho(1-\delta))/n) \geq$$
$$-\log \pi_{NP}(x)/n \geq 1 - (h(\nu) + \epsilon)\} \geq (1-\delta)(1-\rho). \tag{23}$$

Having taken into account that this inequality is valid for all positive ϵ, δ and ρ we obtain the first statement of the theorem.

The proof of the second statement of the theorem is closed to the previous one. First, from the theorem 2 we see that for any $\epsilon > 0, \delta > 0$ we define

$$\hat{\Phi}_{\epsilon,\delta,n} = \{x : h(\nu) - \epsilon < |\phi(x_1 ... x_n)|/n < h(\nu) + \epsilon\}. \tag{24}$$

Note that from (11) we can see that there exists such n'' that, for $n > n''$,

$$\nu(\hat{\Phi}_{\epsilon,\delta,n}) > 1 - \delta. \tag{25}$$

We will use the set $\Phi_{\epsilon,\delta,n}$ (see (15)). Having taken into account the SMB theorem (13) and (25), we can see that

$$\nu(\hat{\Phi}_{\epsilon,\delta,n} \cap \Phi_{\epsilon,\delta,n}) > 1 - 2\delta, \tag{26}$$

if $n > \max(n', n'')$.

From this moment, the proof begins to repeat the proof of the first statement if we use the set $(\hat{\Phi}_{\epsilon,\delta,n} \cap \Phi_{\epsilon,\delta,n})$ instead of $\Phi_{\epsilon,\delta,n}$. The only difference is in the definitions (17) and (19) which should be changed as follows.

$$\Lambda_x = \{y : |\phi|(y)| < |\phi(x)|\ \} \cap (\hat{\Phi}_{\epsilon,\delta,n} \cap \Phi_{\epsilon,\delta,n})$$

and Ψ_ρ is such a subset of $(\hat{\Phi}_{\epsilon,\delta,n} \cap \Phi_{\epsilon,\delta,n})$ that

$$\nu(\Psi_\rho) = \rho \;\&\; \forall u \in \Psi_\rho \, \forall v \in (\Phi_{\epsilon,\delta,n} \setminus \Psi_\rho) \implies |\phi(u)| \leq |\phi(v)|.$$

If we replace π_{NP} with π_{τ_ϕ} and δ with 2δ, we obtain the proof of the second statement. Theorem is proven.

Acknowledgment

This work was supported by Russian Foundation for Basic Research (grant 18-29-03005).

References

[1] L'Ecuyer, P. History of uniform random number generation. In Proceedings of the WSC 2017-Winter Simulation Conference, Las Vegas, NV, USA, 3–6 December 2017.

[2] P. L'Ecuyer and R. Simard, "TestU01: AC library for empirical testing of random number generators," *ACM Transactions on Mathematical Software*, vol. 33, no. 4, p.22, 2007.

[3] Herrero-Collantes, M., Garcia-Escartin, J.C. Quantum random number generators, *Rev. Mod. Phys.*, vol. 89, 015004, 2017.

[4] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, and S. Vo, *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications.* National Institute of Standards and Technology, 2010.

[5] Demirhan, H., Bitirim, N. Statistical Testing of Cryptographic Randomness. *Journal of Statisticians: Statistics and Actuarial Sciences, IDIA* 2016, 9, 1, 1-11.

[6] Zubkov A. M., Serov A. A., "Testing the NIST Statistical Test Suite on artificial pseudorandom sequences", Matematicheskie Voprosy Kriptografii ("Mathematical Aspects of Cryptography"), 2019, 10, 2, 89–96.

[7] B. Ryabko, " On asymptotically optimal tests for random number generators," arXiv:1912.06542 [cs.IT], 2019.

[8] B. Ryabko and J. Astola, " Universal Codes as a Basis for Time Series Testing," *Statistical Methodology*, vol.3, pp. 375-397, 2006.

[9] B. Ryabko, J. Astola, M. Malyutov, *Compression-Based Methods of Statistical Analysis and Prediction of Time Series,* Springer International Publishing Switzerland, 2016.

[10] T. M. Cover and J. A. Thomas, *Elements of information theory.* New York, NY, USA: Wiley-Interscience, 2006.

[11] M. Kendall, A. Stuart, *The advanced theory of statistics; Vol.2: Inference and Relationship*; Hafner Publishing Company: New York, NY, USA, 1961.

Pseudo-random Number Generators with Proven Statistical Properties

Boris Ryabko[1,2] and Viacheslav Zhuravlev[2]

[1]Institute of Computational Technologies of SB RAS, Novosibirsk, Russia
[2]Novosibirsk State University, Russia
boris@ryabko.net

Abstract

Pseudo-random number generators (PRNGs) are widely used in data protection systems and are intensively researched in modern cryptography. In this report, we describe a PRNG class, which, firstly, has been successfully tested using the most powerful modern test batteries, and secondly, it is proved the generators from these class generate normal sequences, that is, for any generated sequence $x_1x_2...$ and any binary word w

$$\lim_{t\to\infty} \nu_t(w)/(t-|w|) = 2^{-|w|}$$

where $\nu_t(w)$ is the number of occurrences of w in the sequence $x_1...x_{|w|}$, $x_2...x_{|w|+1}$, ..., $x_{t-|w|+1}...x_t$.

Keywords: cryptographic randomness, pseudo-random number generators, normal sequences, battery of tests.

1 Introduction

Pseudo-random number generators (PRNG) are designed to generate sequences of binary digits, which are distributed as a result of throwing an "honest" coin, or, precisely, obey the Bernoulli distribution with parameters $(1/2, 1/2)$. Note that quite often this i.i.d. process and the sequences generated from it are called "truly random" [1].

True random sequences are very desirable in cryptography, simulation, and modeling applications. Of course, it is practically impossible to generate them tossing a coin and nowadays there are many so-called generators of pseudo-random numbers, whose aim is, informally speaking, to calculate sequences that mimic the truly random (see [1]).

A modern PRNG is a computer program whose input is a short word (a so-called seed), whereas its output is a long (compared to the input) word. Having taken into account that the seed is a true random word, the PRNG

can be considered as an expander of randomness which stretches a short seed into a long word [1].

The output of a perfect PRNG should look like a truly random sequence. However, it is impossible. More precisely, a mathematically correct definition of a random sequence was obtained in a framework of algorithmic information theory established by Kolmogorov (see [2, 3, 4, 5, 6]). In particular, it is shown that any algorithm (i.e., a Turing machine) can neither generate (infinite) random sequences nor stretch a short random sequence into a longer one. It means that PRNGs do not exist. The same is true in the framework of Shannon's information theory. Indeed, it is known that the Shannon entropy of the true random process is one bit per letter, whereas for all other processes the entropy (per letter) is less than one (see [7]). On the other hand, any PRNG stretches a short true random sequence into a long one. So, the entropy of the output is not greater then the entropy of the input and, hence, the per letter entropy of the output is strictly less than 1 bit. Therefore, the demands of true randomness and low entropy are contradictory. Thus, we see that, in a framework of algorithmic information theory, as well as in a framework of Shannon information theory, "perfect" PRNGs do not exist.

In such a situation, researchers suggest and investigate PRNGs which meet some "probabilistic" properties of truly random sequences [1, 8]. One of such properties is that a PRNG generates so-called normal (or ∞-distributed) sequences. The following definition of normal sequences belongs to Borel [9]: A sequence of digits in base 2 is k-distributed if for any k-letter word w over the alphabet $\{0, 1\}$

$$\lim_{t\to\infty} \nu_t(w)/(t - |w|) = 2^{-|w|} \tag{1}$$

where $\nu_t(w)$ is a number of occurrences of w in the sequence $x_1...x_{|w|}$, $x_2...x_{|w|+1}$, ..., $x_{t-|w|+1}...x_t$. The sequence is normal (or ∞-distributed) if it is k-distributed for any $k \geq 1$. (Borel called normal a real number from the interval $(0, 1)$ whose expansion in base 2 is a normal sequence, and showed that almost all real numbers are normal (with respect to the uniform measure) [9]. That is why, the property that PRNG generates normal sequences is very desirable (see [1, 8]).

In [10], the so-called two-faced processes were described which, on the one hand, generate normal sequences, and on the other hand, their entropy (per letter) may be close to zero. As shown in [10], they can be a promising tool for constructing a PRNG. The only problem is that the property of being a normal sequence is asymptotic, but, informally, real PRNGs must

generate sequences that are recognized by random modern battery of tests for any length of the sequence. (In short, we call such PRNGs statistically acceptable).

This paper aims to suggest such PRNGs that

– generate normal sequences.

– generate statistically acceptable sequences of any length.

The first property was proved mathematically in [10], and the second one was tested experimentally using batteries of statistical tests [11, 12], which are currently considered the most powerful.

The rest of the paper is as follows: the next part describes two-faced random processes and descriptions of PRNGs that generate normal sequences. The third part contains results of experiments which show that proposed PRNGs generate sequences which recognized as truly random by the batteries of tests from [11, 12].

2 Two-faced processes and PRNG generated normal sequences

2.1 Two-faced Markov chains

The purpose of this part is an informal explanation of the basic ideas underlying the proposed PRNG, which are connected with the so-called two-faced Markov chains.

First we consider several examples of two-faced Markov chains. Let a matrix of transition probabilities T_1 be as follows:

$$T_1 = \begin{array}{c|cc} & 0 & 1 \\ \hline 0 & \nu & 1-\nu \\ 1 & 1-\nu & \nu \end{array} \quad , \tag{2}$$

where $\nu \in (0,1)$ (i.e. $P\{x_{i+1}=0|x_i=0\}=\nu$, $P\{x_{i+1}=0|x_i=1\}=1-\nu, ...$).

For example, let $\nu = 0.9$. Then, a "typical" output sequence can be as follows:

$$0000000000\ 111111111\ 0000000000\ 1111111\ 0\,...$$

(Here the gaps correspond to state transitions.) If $\nu = 0.1$, then a "typical" output sequence is

$$01010101\ 1010101010\ 0101010101010101\ 1010\,...\ .$$

On the one hand, thse sequences are not truly random. On the other hand, the frequencies of 1's and 0's go to 1/2 due to the symmetry of the matrix (2). Hence, the output is 1-distributed.

Define

$$\hat{T}_1 = \begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 1-\nu & \nu \\ 1 & \nu & 1-\nu \end{array} \tag{3}$$

$$T_2 = (T_1\hat{T}_1) = \begin{array}{c|cccc} & 00 & 01 & 10 & 11 \\ \hline 0 & \nu & 1-\nu & 1-\nu & \nu \\ 1 & 1-\nu & \nu & \nu & 1-\nu \end{array}$$

(Here $P\{x_{i+1}=0|x_i=0,x_{i-1}=0\}=\nu$, $P\{x_{i+1}=0|x_i=0,x_{i-1}=1\}=1-\nu,\dots$.). For $\nu=0.9$ the "typical" output sequence can be as follows:

$$000000000000\ \ 11011011011011011011011011011011\ \ 000\dots$$

It can be easily seen that the frequency of any two-letter word goes to 1/4.

The general definition of a transition matrix two-faced Markov chain is as follows. The $k+1$-order transition matrix $T_{k+1}=T_k\hat{T}_k$, $\hat{T}_{k+1}=\hat{T}_kT_k$, $k=2,3,\dots$, whereas T_1 and $\hat{T}_1$ are defined in (2) and (3). To define the process $x_1x_2\dots$ the initial probability distribution needs to be specified. We define the initial distribution of the processes T_k and $\bar{T}_k$, $k=1,2,\dots,$ to be uniform on $\{0,1\}^k$, i.e. $P\{x_1\dots x_k=u\}=2^{-k}$ for any $u\in\{0,1\}^k$.

The following theorem from [10] describes the main properties of the processes defined above.

Theorem 1. *Let a sequence $x_1x_2\dots$ be generated by the process T_k (or $\bar{T}_k$), $k\geq 1$ and u be a binary word of length k. Then,*

i) if the initial state obeys the uniform distribution over $\{0,1\}^k$, then for any $j\geq 0$

$$P(x_{j+1}\dots x_{j+k}=u)=2^{-|u|}. \tag{4}$$

ii) for any initial state of the Markov chain

$$\lim_{j\to\infty} P(x_{j+1}\dots x_{j+k}=u)=2^{-|u|}. \tag{5}$$

iii) With probability one the Markov chains T_k and $\hat{T}_k$ generates k-distributed sequences.

iiii) For each $\nu \in (0,1)$ the k-order Shannon entropy (h_k) of the processes T_k and $\bar{T}_k$, equals 1 bit per letter whereas the limit Shannon entropy (h_∞) equals $-(\nu \log_2 \nu + (1-\nu)\log_2(1-\nu))$.

(Here the Shannon entropy of a stationary process μ of order m, $m = 1, 2, ...$, is defined by

$$h_m = -\sum_{u \in \{0,1\}^{m-1}} \mu(u) \sum_{v \in \{0,1\}} \mu(v/u) \log \mu(v/u)$$

and the limit Shannon entropy is defined by $h_\infty = \lim_{m\to\infty} h_m$, see [7].)

This theorem explains the name "two-faced process." The probability distribution of k -bit words is uniform, while the distribution of K -bit words for $K > k$ can be far from uniform.

So, we can see that for any integer k there exist a low-entropy random processes whose outputs simulate the true random sequences up to the length of the word k. In the following part we show how to convert them into a process with an ∞-distributed output.

2.2 ∞-distributed processes and PRNGs generated normal sequences

First, we give a formal definition of the two-faced processes.

Definition 1. *If the equation (5) is valid for any $w \in \{0,1\}^k$, the process is asymptotically two-faced of order k. The process is two-faced of order k if the equation (4) is true.*

The following simple statement from [10] shows that, in a certain sense, there are many two-faced processes. More precisely, the following theorem is true.

Theorem 2. *Let $X = x_1x_2...$, $Y = y_1y_2...$ be a random processes. We define the process $Z = z_1z_2...$ by the following equations $z_1 = x_1 \oplus y_1$, $z_2 = x_2 \oplus y_2, ...$ where $a \oplus b = (a+b) \bmod 2$. If X is a k-order (asymptotically) two-faced process, then Z is also the k-order (asymptotically) two-faced process ($k \geq 1$).*

Now, we describe a family of processes that generate ∞- distributed sequences. Suppose that $m^* = m_1, m_2, ...$ is a sequence of integers, $m_1 < m_2 < m_3....$ and $X^1 = x_1^1x_2^1..., X^2 = x_1^2x_2^2..., X^3 = x_1^3x_2^3..., ...$ are (asymptotically)

two-faced processes of order $m_1, m_2, ...$, correspondingly. Define a process $W = w_1 w_2 \ ...$ by the following equation:

$$w_i = \begin{cases} x_i^1 & i \le m_1, \\ x_i^1 \oplus x_i^2 & m_1 < i \le m_2, \\ x_i^1 \oplus x_i^2 \oplus x_i^3 & m_2 < i \le m_3, \\ \ldots\ldots\ldots\ldots\ldots\ldots & \end{cases} \tag{6}$$

and denote it as $\sum_{i=1}^{\infty} X^i$.

Theorem 3. *[10] Let all X^i, $i = 1, 2, ...$, be asymptotically two-faced, then $\sum_{i=1}^{\infty} X^i$ is normal.*

Note that the proof immediately follows from Theorem 2.

The proposed PRNG construction repeats the process W in (6), but before describing it, we need to describe how to transform any random process into k -distributed for any integer k. It can be done as follows:

Theorem 4. *[10] Let there be an integer k, a finite word $x_{-k+1}x_{-k+2}...x_0$, and infinite sequence $u_1u_2...$, which is generated by a stationary ergodic source. Then, $u_1u_2...$ can be transformed into k-distributed $x_1x_2...$ as follows:*

$$x_r = u_{r+1} \oplus \bigoplus_{i=r-k+1}^{r} x_i, \ \ r = 1, 2, \ldots \ . \tag{7}$$

Now we can describe a PRNG which is an implementation of (6).

Input: 1. The desired length of the generating pseudo-random sequence (N).

2. A sequence of integers $m_1 < m_2 < ...m_s \le N$, which are parameters of the method (to simplify the notation, it will be convenient to assume that all N/m_i are integers).

3. a seed of the PRNG, i.e. a sequence of random bits $r_1r_2...r_R$, where $R = m_1 + \sum_{i=1}^{s} M/m_i$.

Algorithm: We generate sequences $x_1^0x_2^0...x_N^0$, $x_{m_1+1}^1x_{m_1+2}^1...x_N^1$, ..., $x_{m_s+1}^sx_{m_s+2}^s...x_N^s$ according to (7), where $k^0 = m_1$, $x_{-m_1+1}^0 = r_1, x_{-m_1+2}^0 = r_2, ...$, $x_0^0 = r_{m_1}$,

$$u_i^0 = \begin{cases} 0, & if \ \ i/m_1 \ \ is\ not\ integer \\ r_{m_1+i/m_1}, & if \ \ i/m_1 \ \ is\ integer \end{cases},$$

$i = 1, 2, ..., N$,
$k^1 = m_2$, $x_{-m_1+1}^1 = x_1^0, x_{-m_1+2}^1 = x_2^0, ..., x_0^1 = x_{m_1}^0$,

$$u_i^1 = \begin{cases} 0, & if \ \ i/m \ \ is\ not\ integer \\ r_{m_1+N/m_1+i/m_2}, & if \ \ i/m_2 \ \ is\ integer \end{cases},$$

$i = 1, 2, ..., N, ...,$
$k^{s+1} = m_s,\ x^s_{-m_s+1} = x^{s-1}_1, x^s_{-m_s+2} = x^{s-1}_2, ..., x^s_0 = x^{s-1}_{m_{s-1}},$

$$u^s_i = \begin{cases} 0, & if \quad i/m \quad is\ not\ integer \\ r_{m_1+N/m_1+N/m_2+N/m_2+...+i/m_s}, & if \quad i/m_s \quad is\ integer \end{cases},$$

Finally, we calculate the output sequence $x^{output}_1 = \bigoplus^s_{i=0} x^i_1, ..., x^{output}_N = \bigoplus^s_{i=0} x^i_N$.

Note that all calculations can be performed in such a way that the sequences $x^0_i,\ x^1_i,\ ..., x^s_i$, $i = 1, ..., N$, are not stored in memory. The required amount of memory (in bits) and time (per output letter) are $O(sm_s)$.

3 Experimental investigation of the proposed PRNG

In this part we describe the results of experiments that were carried out with the PRNG introduced above. We tested the output of the PRNG by tests from the well-known test batteries suggested in [11, 12].

In our experiments, we used the PRNG under consideration with three parameters m_1, m_2, m_3, which were investigated for different values and different lengths of the output sequence. In the experiments described below, seeds were obtained using MRG32k3a from [11], but some experiments were performed with other generators from [11]. In all cases, the test results were independent of seed generators.

When using batteries "Alphabit" and "Rabbit" the length of the output sequence was 1 GB, while other batteries define their strategy for using output sequences themselves. When using NIST battery [12] the length of the output sequence was 100 MB (1 GB is impossible for this test). The results of applying the tests from [11, 12] are presented in Table 1.

The results of these experiments show that there are values of the parameters m_1, m_2 for which the output sequences cannot be distinguished from truly random ones using the best modern battery tests. Ratio $|seed|/|ouput|$ may be around 0.001, which is acceptable for many cryptographic applications. In particular, the PRNG with parameters $m_1 = 1021$, $m_2 = 1021001$, $m_1 = 1021$, $m_2 = 102101$ and some others can be recommended for practical use.

We see that statistically acceptable PRNGs can be obtained with m_1 and m_2. However, we also performed similar experiments with m_3 approximately equal to m^2_2. It turned out that for large m_2 (that is, $m_2 \geq 907$) this does not affect the test results.

m_1	m_2	NIST	Alphabit	Rabbit	SmallCrush	Crush	BigCrush
31	257	Passed	Passed	Rejected	Passed	Rejected	Rejected
31	907	Passed	Passed	Passed	Passed	Passed	Rejected
31	3571	Passed	Passed	Passed	Passed	Passed	Passed
61	257	Passed	Passed	Rejected	Passed	Rejected	Rejected
61	907	Passed	Passed	Passed	Passed	Passed	Passed
61	3571	Passed	Passed	Passed	Passed	Passed	Passed
127	257	Passed	Passed	Passed	Passed	Rejected	Rejected
127	907	Passed	Passed	Passed	Passed	Passed	Passed
127	3571	Passed	Passed	Passed	Passed	Passed	Passed
127	1277	Passed	Passed	Passed	Passed	Passed	Passed
127	12703	Passed	Passed	Passed	Passed	Passed	Passed
127	126997	Passed	Passed	Passed	Passed	Passed	Passed
257	2579	Passed	Passed	Passed	Passed	Passed	Passed
257	25703	Passed	Passed	Passed	Passed	Passed	Passed
257	257003	Passed	Passed	Passed	Passed	Passed	Passed
509	5087	Passed	Passed	Passed	Passed	Passed	Passed
509	50893	Passed	Passed	Passed	Passed	Passed	Passed
509	508987	Passed	Passed	Passed	Passed	Passed	Passed
1021	10211	Passed	Passed	Passed	Passed	Passed	Passed
1021	102101	Passed	Passed	Passed	Passed	Passed	Passed
1021	1021001	Passed	Passed	Passed	Passed	Passed	Passed

Table 1: Results of experiments.

4 Conclusion

In this article, we propose PRNGs that generate normal sequences (as far as the asymptotic properties can be valid for a real computer program). A series of experiments made it possible to find PRNG parameters for which the output sequence does not differ from truly random ones if we use the best batteries of statistical tests. This PRNG may be interesting for practical use. Thus, the proposed PRNGs combine an acceptable practical behavior with mathematically proven asymptotic properties.

Acknowledgment

This work was supported by the Russian Foundation for Basic Research (grant 18-29-03005).

References

[1] L'Ecuyer, P. History of uniform random number generation. In Proceedings of the WSC 2017-Winter Simulation Conference, Las Vegas, NV, USA, 3–6 December 2017.

[2] Li, M.; Vitányi, P. *An Introduction to Kolmogorov Complexity and Its Applications*; Springer-Verlag: Berlin/Heidelberg, Germany, 2008.

[3] Calude, C.S. *Information and Randomness—An Algorithmic Perspective*; Springer-Verlag: Berlin/Heidelberg, Germany, 2002.

[4] Downey, R.G.; Hirschfeldt, D.R. *Algorithmic Randomness and Complexity*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2010.

[5] Downey, R.; Hirschfeldt, D.R.; Nies, A.; Terwijn, S.A. Calibrating randomness. *Bull. Symb. Log.* **2006**, *12*, 411–491.

[6] Merkle, W.; Miller, J.S.; Nies, A.; Reimann, J.; Stephan, F. Kolmogorov–loveland randomness and stochasticity. *Ann. Pure Appl. Log.* **2006**, *138*, 183–210.

[7] Cover, T.M.; Thomas, J.A. *Elements of Information Theory*; Wiley-Interscience: New York, NY, USA, 2006.

[8] L'Ecuyer, P. *Random Number Generation and Quasi-Monte Carlo*; Wiley: Hoboken, NJ, USA, 2014.

[9] Borel, E. Le continu mathématique et le continu physique. 1909.

[10] B. Ryabko, Low-Entropy Stochastic Processes for Generating k-Distributed and Normal Sequences, and the Relationship of These Processes with Random Number Generators. *Mathematics* , 7(9), 838, 2019.

[11] L'Ecuyer, P.; Simard, R. TestU01: A C library for empirical testing of random number generators. *ACM Trans. Math. Softw. (TOMS)* **2007**, *33*, 22.

[12] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, and S. Vo, *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications.* National Institute of Standards and Technology, 2010.

Symmetric cryptography

A Compact Bit-Sliced Representation of Kuznyechik S-box

Olga Avraamova[1], Denis Fomin[2], Vladimir Serov[1], Alexander Smirnov[1], and Vasily Shokov[1]

[1] Lomonosov Moscow State University, Russia
[2] Higher School of Economics, Russia
olga.avraamova@gmail.com, dfomin@hse.ru, v_serov_@mail.ru, asmirnov80@gmail.com, shokov@srcc.msu.ru

©Academy of Cryptography of the Russian Federation, 2020

Abstract

In this paper we consider a bit-sliced implementation of the GOST R 34-12.2015 «Kuznyechik» non-linear transformation. We combine analytical and computer methods to get a 235 Boolean operations representation.

Keywords: block cipher, S-box, Kuznyechik, bit-slice, Boolean functions minimization.

1 Introduction

A permutation is a bijective mapping of a non-empty finite set onto itself. In applications, permutations of V_n, where V_n is an n-dimensional vector space over Galois field $GF(2)$, are often used. Usually the dimension of the vector space is not large, most popular versions being $n = 4$ or $n = 8$. Permutation can be implemented as a lookup table and this representation is often called an S-box.

S-boxes are widely used in the synthesis of block ciphers and hash-functions. They allow one to combine necessary non-linearity with reasonable implementation complexity of the overall scheme. At the same time, cipher strength against known methods of cryptanalysis depends strongly on certain properties of S-boxes used, and its potential speed and lower resource-consumption – on the effectiveness of software and hardware implementation of these elements.

We use the number of Boolean operations as the measure of complexity of an S-box implementation. In this paper we consider implementations of the non-linear transformation of «Kuznyechik» GOST R 34-12.2015 [1] in the basis of AND, OR, NOT and XOR Boolean functions and try to get

a way to make it as compact as possible. The choice of these four functions as the basis is justified by the existence of its effective hardware and software implementation on different platforms. Inside formulas, we'll use short notation: $\cdot$ for AND, $+$ for OR, $\neg$ for NOT and $\oplus$ for XOR.

In the classical problem of Boolean functions minimization the basis is formed by conjunction (AND), disjunction (OR) and negation (NOT). In this form the problem is intimately connected with electrical schemas simplification. Karnaugh charts used to complete the task for electrical schemas are quite usable for manual optimization of Boolean functions with small number of variables. Computer analog of this method was developed by W. Quine and extended by E. McCluskey and is known as Quine-McCluskey algorithm. The distinctive feature of our situation is the existence of additional XOR operation, so finding transformation rules for formulae minimization for the new set of operations is quite a reasonable task.

Composition of the paper is the following. In section 1 we consider the decomposition of «Kuznyechik» permutation found in Alex Biryukov, Leo Perrin, and Aleksei Udovenko paper [2], which we call BPU-decomposition. In section 2 we implement linear and bilinear elements of BPU-decomposition and eliminate branching. In part 3 we consider ways of computer optimization of non-linear elements of the decomposition. In concluding section we compare the overall complexity of our method with other existing solutions.

2 BPU-decomposition

In «Kuznyechik» algorithm, by GOST R 34-12.2015, a non-linear transformation π of vector space V_8 is used. If we try to minimize it directly, using, for example, Espresso algorithm (in Logic Friday [3] realization), we get 3492 Boolean operations – quite a lot. So some other approach is needed.

The authors of [2] suggested to represent π as a composition of non-linear transformations ν_0, ν_1, I, σ, φ of V_4, linear transformations α and ω of V_8 and multiplication $\circledcirc$ in Galois field $GF(2^4, \circledcirc, \oplus) = GF(2)[x]/(f(x))$ with irreducible polynomial $f(x) = x^4 \oplus x^3 \oplus 1$, where $\oplus$ is addition and $\circledcirc$ - multiplication in the finite field.

Consider the action of π on the set of pairs $(l, r), l, r \in GF(2^4)$. In [2] the following algorithm to compute $\pi(l \parallel r)$ is used:

1) $(l \parallel r) := \alpha(l \parallel r)$,
2) if $r = 0$, then $l := \nu_0(l)$, else $l := \nu_1(l \circledcirc I(r))$;
3) $r := \sigma(r \circledcirc \varphi(l))$,
4) $(l \parallel r) := \omega(l \parallel r)$.

The value of $(l \parallel r)$ after step 4 equals to $\pi(l \parallel r)$. The non-linear transformations (not all of them are bijective) ν_0, ν_1, I, σ, φ are given in the following table (in hexadecimal representation):

I	0, 1, c, 8, 6, f, 4, e, 3, d, b, a, 2, 9, 7, 5
v_0	2, 5, 3, b, 6, 9, e, a, 0, 4, f, l, 8, d, c, 7
v_1	7, 6, c, 9, 0, f, 8, 1, 4, 5, b, e, d, 2, 3, a
φ	b, 2, b, 8, c, 4, 1, c, 6, 3, 5, 8, e, 3, 6, b
σ	c, d, 0, 4, 8, b, a, e, 3, 9, 5, 2, f, 1, 6, 7

3 Implementation of linear and bilinear elements of decomposition and branching elimination

3.1 Bit-sliced implementation of multiplication in the finite field

One of important elements of the decomposition is the multiplication in the finite field $GF(2^4)$. It is used twice during the algorithm. Let's find the Boolean functions representation of this operation. To do it, we regard $GF(2^4)$ as a four-dimensional algebra over $GF(2)$. In this case the components of binary representation of elements of $GF(2^4)$ will be simply the coordinates of these vectors in the standard basis $\{\mathbf{e}_1 = (1000) = \mathbf{8}, \mathbf{e}_2 = (0100) = \mathbf{4}, \mathbf{e}_3 = (0010) = \mathbf{2}, \mathbf{e}_4 = (0001) = \mathbf{1}\}$. (In this section elements of $GF(2^4)$ will be typeset in bold and coordinate indexes will be written above).

By the associative, commutative and distributive laws of $GF(2^4)$, operations $\circledcirc$ and $\oplus$ are commuting, and the distributive law holds. So it is sufficient to know pair-wise products of basic vectors, and the product of any two arbitrary vectors can be computed by linearity:

$$\mathbf{z} = (\mathbf{x} \circledcirc \mathbf{y}) = \left(\sum_{i=1}^{4} x^i \mathbf{e}_i\right) \circledcirc \left(\sum_{j=1}^{4} y^j \mathbf{e}_j\right) = \sum_{i,j} x^i \cdot y^j (\mathbf{e}_i \circledcirc \mathbf{e}_j)$$

in vector form and

$$\mathbf{z}^k = (\mathbf{x} \circledcirc \mathbf{y})^k = \left(\left(\sum_{i=1}^{4} x^i \mathbf{e}_i\right) \circledcirc \left(\sum_{j=1}^{4} y^j \mathbf{e}_j\right)\right)^k = \left(\sum_{i,j} x^i y^j (\mathbf{e}_i \circledcirc \mathbf{e}_j)\right)^k = \\ = \sum_{i,j} x^i \cdot y^j (\mathbf{e}_i \circledcirc \mathbf{e}_j)^k, \quad k = 1, \ldots, 4$$

— in coordinate form.

Let's make a table of pair-wise products of basic vectors:

	1000	0100	0010	0001
1000	1111	1011	1001	1000
0100	1011	1001	1000	0100
0010	1001	1000	0100	0010
0001	1000	0100	0010	0001

Separating coordinates, we have

C^1				C^2				C^3				C^4			
1	1	1	1	1	0	0	0	1	1	0	0	1	1	1	0
1	1	1	0	0	0	0	1	1	0	0	0	1	1	0	0
1	1	0	0	0	0	1	0	0	0	0	1	1	0	0	0
1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1

and for every coordinate z^k $(k = 1, \ldots, 4)$ we have

$$z^k = (x^1, x^2, x^3, x^4) \begin{pmatrix} c^k_{11} & \ldots & c^k_{14} \\ \vdots & \ddots & \vdots \\ c^k_{41} & \ldots & c^k_{44} \end{pmatrix} \begin{pmatrix} y^1 \\ y^2 \\ y^3 \\ y^4 \end{pmatrix}.$$

Now (returning to lower coordinate indexes) we are able to write down formulae for every coordinate:

$$z_1 = (x_1 \oplus x_2 \oplus x_3 \oplus x_4) \cdot y_1 \oplus (x_1 \oplus x_2 \oplus x_3) \cdot y_2 \oplus (x_1 \oplus x_2) \cdot y_3 \oplus x_1 \cdot y_4,$$
$$z_2 = x_1 \cdot y_1 \oplus x_4 \cdot y_2 \oplus x_3 \cdot y_3 \oplus x_2 \cdot y_4,$$
$$z_3 = (x_1 \oplus x_2) \cdot y_1 \oplus x_1 \cdot y_2 \oplus x_4 \cdot y_3 \oplus x_3 \cdot y_4,$$
$$z_4 = (x_1 \oplus x_2 \oplus x_3) \cdot y_1 \oplus (x_1 \oplus x_2) \cdot y_2 \oplus x_1 \cdot y_3 \oplus x_4 \cdot y_4.$$

So to compute z_1 we need 13 operations, for z_2 — 7, for z_3 — 8, and for z_4 — 10. Multiplication in our realization of $GF(2^4)$ requires 38 Boolean operations if we do not use auxiliary variables to store intermediate results. If we use intermediate variables (denote them P with indices), the number of operations can be made even lower. For example, the expression $(x_1 \oplus x_2)$ appears more than once:

$$z_1 = (P_2 \oplus x_4) \cdot y_1 \oplus P_2 \cdot y_2 \oplus P_1 \cdot y_3 \oplus x_1 \cdot y_4,$$
$$z_2 = x_1 \cdot y_1 \oplus x_4 \cdot y_2 \oplus x_3 \cdot y_3 \oplus x_2 \cdot y_4,$$
$$z_3 = P_1 \cdot y_1 \oplus x_1 \cdot y_2 \oplus x_4 \cdot y_3 \oplus x_3 \cdot y_4,$$
$$z_4 = P_2 \cdot y_1 \oplus P_1 \cdot y_2 \oplus x_1 \cdot y_3 \oplus x_4 \cdot y_4,$$
$$P_1 = x_1 \oplus x_2,$$
$$P_2 = x_1 \oplus x_2 \oplus x_3 = P_1 \oplus x_3.$$

The latter realization has complexity 31.

3.2 Implementation of linear mappings

Let's estimate the complexity of linear transformations α and ω in terms of Boolean functions. Matrix representations of alpha and omega are the following:

$$\alpha = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \quad \omega = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Let $l = (l_1, l_2, l_3, l_4)$, $r = (r_1, r_2, r_3, r_4)$ be representations of field elements as vectors. Then α has the following Boolean representation:

$$\begin{aligned}
&\alpha_1(l_1, l_2, l_3, l_4, r_1, r_2, r_3, r_4) = r_1, \\
&\alpha_2(l_1, l_2, l_3, l_4, r_1, r_2, r_3, r_4) = l_2 \oplus r_4, \\
&\alpha_3(l_1, l_2, l_3, l_4, r_1, r_2, r_3, r_4) = l_2 \oplus r_3 \oplus r_4, \\
&\alpha_4(l_1, l_2, l_3, l_4, r_1, r_2, r_3, r_4) = l_1 \oplus l_2 \oplus l_3 \oplus r_1 \oplus r_2 \oplus r_3 \oplus r_4, \\
&\alpha_5(l_1, l_2, l_3, l_4, r_1, r_2, r_3, r_4) = l_1 \oplus r_1 \oplus r_3, \\
&\alpha_6(l_1, l_2, l_3, l_4, r_1, r_2, r_3, r_4) = l_2 \oplus r_2, \\
&\alpha_7(l_1, l_2, l_3, l_4, r_1, r_2, r_3, r_4) = l_4 \oplus r_1 \oplus r_3, \\
&\alpha_8(l_1, l_2, l_3, l_4, r_1, r_2, r_3, r_4) = l_3.
\end{aligned}$$

Overall complexity of α is 14 XOR operations. In the same way, the complexity of ω is 6 XOR operations. So we get 20 Boolean operations to represent two linear mappings.

3.3 Elimination of branching

Let's return to p.2 of BPU algorithm: «2.If $r = 0$ then $l := \nu_0(l)$ else $l := \nu_1(l \circledcirc I(r))$». Write down a Boolean function which takes the value 1 in a single point $r = (0, 0, 0, 0)$ and has zero value in all other points [3]:

$$I_{0,0,0,0}(r) = \bar{r}_1 \cdot \bar{r}_2 \cdot \bar{r}_3 \cdot \bar{r}_4 = \overline{r_1 + r_2 + r_3 + r_4}.$$

The complexity of this function is 4 Boolean operations, the complexity of its negation — 3 operations. Now we can express l by a non-branching formula:

$$l^i = I_{0,0,0,0}(r) \cdot \nu_0^i(l) + \overline{I_{0,0,0,0}(r)} \cdot \nu_1^i(l \circledcirc I(r)), \quad i = 1, \ldots, 4.$$

It should be stressed that we compute $I_{0,0,0,0}(r)$ only once. To be more precise, we first calculate its negation (3 operations) and then, by double-negation law, compute $I_{0,0,0,0}(r)$ itself, which gives one additional operation.

After branching elimination the complexity of our representation has increased by 16 operations. The number of operations of each kind is given in a table below:

	AND	OR	NOT	XOR	Total
$\overline{I_{0,0,0,0}(r)}$		3			3
$I_{0,0,0,0}(r)$			1		1
Final glue by every coordinate	2	1			
Final glue for all coordinaetes (previous line multiplied by 4)	8	4			12
				Total	16

4 Computer implementation of non-linear elements

4.1 Formulation of the problem

Consider a transformation table $2^4 \to 2^4$ or $2^8 \to 2^8$. The goal is to represent it as a set of Boolean functions in the basis of logical functions AND, OR, NOT, XOR with the minimal number of operations. The number of 1- and 2-input operations (AND, OR, NOT, XOR) required to implement a function will be used as the evaluation criteria.

This estimate coincides with the actual complexity of the circuit when it is implemented on integrated circuits containing corresponding logic elements. It also coincides if the FPGA function is used to implement it, since logical operations in this situation are formed using a matrix of macrocells, and the operation does not depend on the FPGA choice.

The algorithm is constructed as follows: at the first optimization step, the Quine-McCluskey algorithm is used, then the steps described below in sections 3.3.1-3.3.6 are iteratively repeated. After that, rules 3.4-3.7 are applied, with repetitive calls to procedures 3.3.1-3.3.6 if necessary.

For an example illustrating the steps of the algorithm we use the table of values of the ν_1 function from BPU-decomposition:

X	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
Y	7	6	c	9	0	f	8	1	4	5	b	e	d	2	3	a

Table 1

Each of the Y bits can be represented in the form of PDNF, consisting of precisely 8 members since it is a coordinate function of a permutation. For example, for Y_0 in this example, this function will have the following form:

$$\begin{aligned} Y_0 = (\overline{X}_0 \cdot \overline{X}_1 \cdot \overline{X}_2 \cdot \overline{X}_3) + (X_0 \cdot X_1 \cdot \overline{X}_2 \cdot \overline{X}_3) + (X_0 \cdot \overline{X}_1 \cdot X_2 \cdot \overline{X}_3) + \\ + (X_0 \cdot X_1 \cdot X_2 \cdot \overline{X}_3) + (X_0 \cdot \overline{X}_1 \cdot \overline{X}_2 \cdot X_3) + (\overline{X}_0 \cdot X_1 \cdot \overline{X}_2 \cdot X_3) + \\ + (\overline{X}_0 \cdot \overline{X}_1 \cdot X_2 \cdot X_3) + (\overline{X}_0 \cdot X_1 \cdot X_2 \cdot X_3). \end{aligned}$$

The complexity of this representation is 47 operations.

Using the optimization techniques shown below, this function can be simplified to the following form:

$$Y_0 = \overline{(X_1 + X_2) \oplus X_0 \oplus X_3}.$$

The complexity of this representation is 4. Thus, for this particular example, the complexity decreases by more than 11 times as a result of our optimization.

As a second example, consider Y_3:

$$\begin{aligned} Y_3 = (\overline{X}_0 \cdot X_1 \cdot \overline{X}_2 \cdot \overline{X}_3) + (X_0 \cdot X_1 \cdot \overline{X}_2 \cdot \overline{X}_3) + (X_0 \cdot \overline{X}_1 \cdot X_2 \cdot \overline{X}_3) + \\ + (\overline{X}_0 \cdot X_1 \cdot X_2 \cdot \overline{X}_3) + (\overline{X}_0 \cdot X_1 \cdot \overline{X}_2 \cdot X_3) + (X_0 \cdot X_1 \cdot \overline{X}_2 \cdot X_3) + \\ + (\overline{X}_0 \cdot \overline{X}_1 \cdot X_2 \cdot X_3) + (X_0 \cdot X_1 \cdot X_2 \cdot X_3). \end{aligned}$$

This function representation requires 45 operations.

With the help of optimizations, this function can be simplified to the following form:

$$Y_3 = ((X_0 \oplus X_3) \cdot X_2) \oplus X_1.$$

The complexity of this representation of the function is 3, which is 15 times better than the original one.

4.2 Quine–McCluskey algorithm

One of the algorithms for minimizing Boolean functions was proposed by Willard Quine and improved by Edward McCluskey. Since the algorithm

is described in sufficient detail in many sources (see, for example, [4]), its implementation will not be described in this paper.

Consider the simplification of the above examples of functions with this algorithm. For a formula representing the function Y_0 (with initial complexity of 47 operations)

$$Y_0 = (\overline{X}_0 \cdot \overline{X}_1 \cdot \overline{X}_2 \cdot \overline{X}_3) + (X_0 \cdot X_1 \cdot \overline{X}_2 \cdot \overline{X}_3) + (X_0 \cdot \overline{X}_1 \cdot X_2 \cdot \overline{X}_3) + \\ + (X_0 \cdot X_1 \cdot X_2 \cdot \overline{X}_3) + (X_0 \cdot \overline{X}_1 \cdot \overline{X}_2 \cdot X_3) + (\overline{X}_0 \cdot X_1 \cdot \overline{X}_2 \cdot X_3) + \\ + (\overline{X}_0 \cdot \overline{X}_1 \cdot X_2 \cdot X_3) + (\overline{X}_0 \cdot X_1 \cdot X_2 \cdot X_3)$$

after optimization, we get a formula with complexity 29 (1.5 times less complexity for this example):

$$Y_0 = (\overline{X}_0 \cdot \overline{X}_1 \cdot \overline{X}_2 \cdot \overline{X}_3) + (X_0 \cdot \overline{X}_1 \cdot \overline{X}_2 \cdot X_3) + \\ + (X_0 \cdot X_1 \cdot \overline{X}_3) + \\ + (X_0 \cdot X_2 \cdot \overline{X}_3) + (\overline{X}_0 \cdot X_1 \cdot X_3) + (\overline{X}_0 \cdot X_2 \cdot X_3).$$

For the Y_3 function (the initial complexity of the representation is 45 operations), the Quine–McClusky method gives the result of 22 operations (double optimization):

$$Y_0 = (X_0 \cdot \overline{X}_1 \cdot X_2 \cdot \overline{X}_3) + (\overline{X}_0 \cdot \overline{X}_1 \cdot X_2 \cdot X_3) + (\overline{X}_0 \cdot X_1 \cdot \overline{X}_3) + \\ + (X_0 \cdot X_2 \cdot X_3) + (X_1 \cdot \overline{X}_2).$$

4.3 Further optimization of the results of the Quine–McClaskey algorithm

As further steps for optimizing Boolean functions, an algorithm consisting of several steps is proposed. Moreover, each step is performed for all members of the polynomial inside the brackets. If optimization has been performed at any of the steps of the algorithm, then the algorithm starts again from the first step.

4.3.1 Duplicate Elimination

This step removes duplicates that may have appeared in previous iterations:

$$X_1 \cdot X_1 \cdot X_2 = X_1 \cdot X_2,$$
$$X_1 + X_1 + X_2 = X_1 + X_2,$$
$$X_1 \oplus X_1 \oplus X_2 = X_2.$$

4.3.2 Removing the common factor from under the logical OR, which is performed on logical AND

$$(X_1 \cdot X_2) + (X_1 \cdot X_3) = X_1 \cdot (X_2 + X_3).$$

For the above example (Y_0 function with complexity 47), the representation after this transformation takes the form:

$$Y_0 = \big(\overline{X}_0 \cdot ((\overline{X}_1 \cdot \overline{X}_2 \cdot \overline{X}_3) + (X_3 \cdot (X_1 + X_2)))\big) + \\ + \big(X_0 \cdot \big((\overline{X}_1 \cdot \overline{X}_2 \cdot X_3) + ((X_1 + X_2) \cdot \overline{X}_3\big)\big).$$

with the complexity value of 20.

4.3.3 Optimization of occurrences of the same term with and without negation for logical AND and OR, as well as optimization of constant members

We use formulas $X_1 \cdot \overline{X}_1 = 0, X_1 + \overline{X}_1 = 1, X_1 \cdot 0 = 0, X_1 + 1 = 1$.

4.3.4 Optimization of «NOT»

At this step, an attempt is made to reduce the number of NOT operations, using a single formula

$$\overline{X}_1 + \overline{X}_2 = \overline{X_1 \cdot X_2}.$$

For the above example (formula with complexity 47), the representation of the function after this operation will take the following form:

$$Y_0 = \big(\big(\overline{X_1 + X_2 + X_3} + (X_1 + X_2) \cdot X_3\big) \cdot \overline{X}_0\big) + \\ + \big(\big((\overline{X}_1 \cdot \overline{X}_2 \cdot X_3) + ((X_1 + X_2) \cdot \overline{X}_3)\big) \cdot X_0\big)$$

with complexity 18.

4.3.5 Search on the table of pattern optimal functions

At this step, a table of optimal functions is used. The construction of the table is described below. A table of values of the function being optimized (or its parts) is generated. After that the values from this table are compared with the reference ones from the table of optimal functions. If they coincide, the pattern optimal function is used instead of the original one. At the moment, the table of optimal functions is constructed for functions of up to 4 variables.

To construct the table of optimal functions, the following approach was used (let us consider the case of compiling a table for three variables). First,

functions are created within one operation (for each of the AND, OR, XOR operations) by sequentially arranging the brackets and NOT operations, for example:

$$F = X_0 + X_1 + X_2,\ F = \overline{X_0 + X_1 + X_2},\ F = \overline{X}_0 + X_1 + X_2,$$
$$F = \overline{\overline{X}_0 + X_1 + X_2},\ F = \overline{X}_0 + \overline{X}_1 + X_2,\ F = \overline{\overline{X}_0 + \overline{X}_1 + X_2},$$
$$F = \overline{X_0 + X_1} + X_2,\ F = \overline{\overline{X_0 + X_1} + X_2}.$$

and so on.

After that, formulas using combinations of binary operations are added (OR and XOR, for example), for which all combinations of parentheses and negation are also sorted out, for example:
$F = (X_0 + X_1) \oplus X_2, F = X_0 + (X_1 \oplus X_2), F = \overline{(X_0 + X_1)} \oplus X_2$, and so on.

At the next step, functions with the same value tables are removed from the list of functions, while leaving the function with the minimum number of operations. The final list of pattern optimal functions is entered into the program in symbolic form from a file, which allows, on one hand, not to spend time on calculating the table each time the algorithm is run, and, on the other hand, it allows to supply the table with functions obtained by other algorithms or empirically. We consider two examples of optimization of the components of the function Y_0

$$F = (\overline{X}_1 \cdot \overline{X}_2 \cdot X_3) + ((X_1 + X_2) \cdot \overline{X}_3).$$

The table of values for this function (the number of operations is 8) is the following:

X_1	0	1	0	1	0	1	0	1
X_2	0	0	1	1	0	0	1	1
X_3	0	0	0	0	1	1	1	1
F	0	1	1	1	1	0	0	0

In the table of optimal functions there is a function with complexity 2, the truth table of which is similar to the above:

$$F = X_3 \oplus (X_1 + X_2).$$

Accordingly, part of the polynomial can be replaced with this function.

Consider another example (the initial complexity of the representation is 8):

$$F = (\overline{X_1 + X_2 + X_3} + ((X_1 + X_2) \cdot X_3)) \cdot \overline{X}_0.$$

Note that in this polynomial there is a common part $X_1 + X_2$. Let us denote it by P_1. The function will take form:

$$F = (\overline{P_1 + X_3} + (P_1 \cdot X_3)) \cdot \overline{X}_0.$$

According to the truth table, the algorithm finds the following optimal function:

$$F = \overline{X_0 + (X_3 \oplus P_1)}.$$

Let us return from P back to $X_1 + X_2$

$$F = \overline{X_0 + (X_3 \oplus (X_1 + X_2))}.$$

The complexity of the resulting function is 4. As a result of this optimization step, the initial function will take the following form (complexity of 8 operations):

$$Y_0 = \big(\overline{((X_1 + X_2) \oplus X_3) + X_0}\big) + \big(((X_1 + X_2) \oplus X_3) \cdot X_0\big).$$

4.3.6 Search for XOR possible usage

This step analyzes the possibility to use XOR, in accordance with the formulas

$$X_0 \cdot \overline{X}_1 + \overline{X}_0 \cdot X_1 = X_0 \oplus X_1$$
$$\overline{(X_0 + X_1)} + (X_0 \cdot X_1) = \overline{X_0 \oplus X_1}.$$

For the above example, the formula for the function Y_0 will take the form

$$Y_0 = \overline{X_0 \oplus (X_1 + X_2) \oplus X_3}.$$

As a result, the complexity of the formula has become 4, which is about 10 times less than the original.

4.4 Additional optimization by combining brackets

In some cases, the above optimizations do not provide the minimal representation of the function.Consider an example. After a number of optimizations Y_2 function has become like that:

$$Y_2 = (\overline{X}_1 \cdot \overline{X}_2 \cdot X_3) + ((X_1 + X_2) \cdot \overline{X}_3).$$

An additional step is provided for handling such cases. If there are more than two terms under the brackets, then several functions are formed with different placement of brackets:

$$Y_2 = (\overline{X}_1 \cdot (\overline{X}_2 \cdot X_3)) + ((X_1 + X_2) \cdot \overline{X}_3), Y_2 = (\overline{X}_2 \cdot (\overline{X}_1 \cdot X_3)) + ((X_1 + X_2) \cdot \overline{X}_3),$$

$$Y_2 = (X_3 \cdot (\overline{X}_1 \cdot \overline{X}_2)) + ((X_1 + X_2) \cdot \overline{X}_3).$$

After that every formula from the set is optimized by the algorithm described above and the formula with smallest resulting complexity is chosen. In our case it is the third function. It is transformed like this:

$$Y_2 = (X_3 \cdot (\overline{X}_1 \cdot \overline{X}_2)) + ((X_1 + X_2) \cdot \overline{X}_3) = (X_3 \cdot \overline{(X_1 + X_2)}) + ((X_1 + X_2) \cdot \overline{X}_3 = \\ = (X_1 + X_2) \oplus X_3.$$

4.5 Primary XOR-optimization

In some cases, the optimization techniques discussed above do not lead to the best representation. Consider bit Y_3 of table 2:

$$Y_3 = (\overline{X}_0 \cdot X_1 \cdot \overline{X}_2 \cdot \overline{X}_3) + (X_0 \cdot X_1 \cdot \overline{X}_2 \cdot \overline{X}_3) + (X_0 \cdot \overline{X}_1 \cdot X_2 \cdot \overline{X}_3) + \\ + (\overline{X}_0 \cdot X_1 \cdot X_2 \cdot \overline{X}_3) + (\overline{X}_0 \cdot X_1 \cdot \overline{X}_2 \cdot X_3) + (X_0 \cdot X_1 \cdot \overline{X}_2 \cdot X_3) + \\ + (\overline{X}_0 \cdot \overline{X}_1 \cdot X_2 \cdot X_3) + (X_0 \cdot X_1 \cdot X_2 \cdot X_3).$$

The initial number of operations in the representation of this function is 45.

As a result of the above optimizations, one can get a function of the following form:

$$Y_3 = \big(\overline{(X_0 \oplus X_3) \cdot X_2} \cdot X_1\big) + \big((X_0 \oplus X_3) \cdot \overline{X}_1 \cdot X_2\big).$$

The number of operations in this representation is 9, which is 5 times less than the original.

Nevertheless, for representations of the function with the number of operations greater than 2, the algorithm makes an additional optimization attempt.

4.6 Using the Algebraic Normal Form (ANF)

Further, the program uses the representation of the function in the form of Zhegalkin polynomial (ANF). First, if one or more variables are found in the Zhegalkin polynomial in the form of terms of the first degree and are not found in any monomial anymore, they can be separated as terms modulo two, immediately reducing the dimension of the problem. If there are no such variables, in some cases it is still possible to reduce the complexity of the representation by separating certain variables as modulo two summands. Let us return to our example from section 3.1 (Table 1). Since the XOR

function takes the value 1 when arguments are different and 0 when their values are the same, we can represent the value Y in the following form:

$$Y_i = X_j \oplus F(X),$$

where X_j is one of X bits, and $F(X)$ is equal to 0 for all cases where $Y_i = X_j$ and is equal to 1 if $Y_i <> X_j$.

Obviously, one needs to choose the bit X so that as many as possible $Y_i = X_j$, because in this case $F(X)$ will have less terms. Consider the values of Y_3 in the following table:

$\mathbf{X}_3$	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
$\mathbf{X}_2$	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
$\mathbf{X}_1$	0	0	1	1	0	[0]	1	[1]	0	0	1	1	[0]	0	[1]	1
$\mathbf{X}_0$	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
$\mathbf{Y}_3$	0	0	1	1	0	[1]	1	[0]	0	0	1	1	[1]	0	[0]	1

For each of $X_i, i = 0, \ldots, 3$, the number of positions in which the value of X_i coincides with the value of Y, is calculated. For this example, we get the following result

$\mathbf{X}_0$	$\mathbf{X}_1$	$\mathbf{X}_2$	$\mathbf{X}_3$
8	12	8	8

Accordingly, the function will have the form

$$Y_3 = X_1 \oplus F(X).$$

We represent $F(X)$ in the form of PDNF with a value of 1 for cases of mismatch between X and Y (mismatching values are indicated by squares around them). So we get the function

$$Y_3 = X_0 \oplus \big((X_0 \cdot \overline{X}_1 \cdot X_2 \cdot \overline{X}_3) + (X_0 \cdot X_1 \cdot X_2 \cdot \overline{X}_3) + \\ + (\overline{X}_0 \cdot \overline{X}_1 \cdot X_2 \cdot X_3) + (\overline{X}_0 \cdot X_1 \cdot X_2 \cdot X_3)\big).$$

The complexity of this representation of the function is 21. However, $F(X)$ can be optimized by the methods indicated above. After the specified processing, the function takes the form

$$Y_3 = X_0 \oplus ((X_0 \oplus X_3) \cdot X_2).$$

The complexity of this representation is 3.

4.7 Merging common parts

As the final step, the algorithm attempts to find common parts both within one function and among all functions of the set, in order to optimize their calculation.

For example, for the above table, the algorithm worked out the following set of functions:

$$
\begin{aligned}
Y_0 &= \overline{X_0 \oplus (X_1 + X_2) \oplus X_3},\\
Y_1 &= \overline{((X_2 + X_3) \oplus (X_0 \cdot X_2)) + X_1} \oplus (X_1 \cdot X_3),\\
Y_2 &= ((X_1 \oplus X_2) \cdot (X_0 \oplus X_3)) \oplus \overline{X}_2,\\
Y_3 &= X_0 \oplus ((X_0 \oplus X_3) \cdot X_2).
\end{aligned}
$$

Total complexity - 19 operations. At the last step, the algorithm explores what is included in Y_0, Y_2, and Y_3. The algorithm denotes such common parts by the symbol P. For the given example, the following result is obtained:

$$
\begin{aligned}
Y_0 &= \overline{P_0 \oplus (X_1 + X_2)},\\
Y_1 &= \overline{((X_2 + X_3) \oplus (X_0 \cdot X_2)) + X_1} \oplus (X_1 \cdot X_3),\\
Y_2 &= ((X_1 \oplus X_2) \cdot P_0) \oplus \overline{X}_2,\\
Y_3 &= X_0 \oplus ((P_0 \cdot X_2)),\\
P_0 &= (X_0 \oplus X_3).
\end{aligned}
$$

As a result of merging the common parts into separate functions, the complexity of our example is reduced from 19 to 17.

4.8 Optimization results

As a result of the above algorithm, the following set of Y functions was produced for the test data table:

$$
\begin{aligned}
Y_0 &= \overline{P_0 \oplus (X_1 + X_2)},\\
Y_1 &= \overline{((X_2 + X_3) \oplus (X_0 \cdot X_2)) + X_1} \oplus (X_1 \cdot X_3),\\
Y_2 &= ((X_1 \oplus X_2) \cdot P_0) \oplus \overline{X}_2,\\
Y_3 &= X_0 \oplus ((P_0 \cdot X_2)),\\
P_0 &= (X_0 \oplus X_3).
\end{aligned}
$$

The total complexity of this set of functions is 17, which is approximately 10 times less than the complexity of the set of non-optimized functions, equal to 188. The following are explicit formulas for all nonlinear functions involved in BPU-decomposition:

F	Presentation	Number of operations
I	$Y_0 = \big(X_0 \cdot \overline{X}_1\big) + \big(\big((X_0 \oplus X_1) + \overline{P}_1\big) \cdot X_3\big)$ $Y_1 = (X_0 \cdot X_2) \oplus \big(\overline{X}_1 \cdot P_1 \cdot P_2\big) \oplus X_3$ $Y_2 = ((X_1 \oplus X_3) \cdot ((X_0 + X_2) \oplus X_1)) \oplus X_2$ $Y_3 = \big(X_1 \cdot \overline{X}_2\big) + (((X_1 \oplus X_2) + P_2) \cdot X_0)$ $P_1 = X_0 \oplus X_2$ $P_2 = X_2 \oplus X_3$	26
v_0	$Y_0 = \big(X_1 \cdot \overline{X}_2\big) + \big(\big((X_1 \oplus X_2) + \overline{X_1 \oplus X_3}\big) \cdot X_0\big)$ $Y_1 = \Big(\overline{(X_0 \oplus X_2) \cdot X_3} \cdot X_1\Big) + \overline{P}_1$ $Y_2 = (X_1 \cdot X_3) \oplus \big(\big((X_2 \oplus X_3) + \overline{X}_1\big) \cdot X_0\big) \oplus \big(X_2 \cdot \overline{X}_3\big)$ $Y_3 = ((X_1 \oplus P_1) \cdot X_2) \oplus ((X_0 \oplus X_3) \cdot X_1)$ $P_1 = X_0 + X_3$	29
v_1	$Y_0 = \overline{(X_1 + X_2) \oplus P_1}$ $Y_1 = \overline{((X_2 + X_3) \oplus (X_0 \cdot X_2)) + X_1} \oplus (X_1 \cdot X_3)$ $Y_2 = ((X_1 \oplus X_2) \cdot P_1) \oplus \overline{X}_2$ $Y_3 = (X_2 \cdot P_1) \oplus X_1$ $P_1 = X_0 \oplus X_3$	29
φ	$Y_0 = \Big(\overline{\big(X_0 \cdot \overline{X}_1\big) + (X_1 \oplus X_3)} \cdot X_2\Big) \oplus \big(\overline{X}_1 \cdot X_3\big) \oplus \overline{X}_0$ $Y_1 = \overline{((X_0 + X_3) \cdot X_1) + X_2} \oplus (X_2 \cdot X_3)$ $Y_2 = \big(\overline{X}_0 \cdot X_3\big) + \big(\big(\big(X_0 + \overline{X}_1\big) \oplus (X_0 \cdot X_3)\big) \cdot X_2\big)$ $Y_3 = (X_0 \cdot X_1) \oplus \big(\big(\big(X_2 + \overline{X}_3\big) \oplus (X_1 \cdot X_2)\big) \cdot \overline{X}_0\big)$	33
σ	$Y_0 = \big(X_0 \cdot \overline{X}_1\big) + \big(\overline{X_1 \cdot P_1} \cdot X_3\big)$ $Y_1 = (((X_2 + X_3) \oplus (X_1 \cdot X_2)) \cdot X_0) \oplus$ $\oplus ((X_2 \oplus X_3) \cdot X_1) \oplus X_3$ $Y_2 = (X_0 \cdot X_1) \oplus \big(\big((X_0 \cdot X_3) \oplus \overline{X}_1\big) \cdot X_2\big) \oplus \overline{X}_1 \oplus X_3$ $Y_3 = \big(X_2 \cdot \overline{X}_3\big) + \big(\big(\overline{X}_3 + P_1\big) \cdot \overline{X}_1\big)$ $P_1 = X_0 \oplus X_2$	31

Summary

In this paper we consider the possibility of bit-slicing the non-linear bijective mapping of GOST R 34-12.2015 «Kuznyechik» block cipher. The «Kuznyechik» permutation, when we use BPU-decomposition, program optimization of smaller non-linear elements and algebraic features of finite field multiplication, fits into 235 Boolean operations (Table 2).

It should be noted that in 2016 a method of constructing s-boxes with

minimal number of logical elements got a patent in Russian Federation [6]. The method protected by this patent allows to realize non-linear mapping of Kuznyechick cipher with complexity of 681 operations (254 ANDs and 436 XORs).

	AND	OR	NOT	XOR	Total
I	8	5	4	9	26
v_0	9	5	6	9	29
v_1	4	3	3	7	17
φ	11	6	8	7	32
σ	11	6	7	9	33
α				14	14
ω				6	6
multiplication in $GF\left(2^4\right)$	16			15	31
multiplication in $GF\left(2^4\right)$	16			15	31
branchng elimination	8	7	1		16
	83	32	29	91	Total: 235

Table 2

Acknowledgments

The authors would like to thank Dmitry Pronkin for helpful discussions on results and writing of the paper, and the anonymous reviewers for their valuable comment.

References

[1] Federal Agency on Technical Regulating and Metrology, "GOST R 34.12-2015. National standard of Russian Federation. Block ciphers.", 2015, in Russian.

[2] Alex Biryukov, Leo Perrin, and Aleksei Udovenko., "Reverse-engineering the SBox of Streebog, Kuznyechik and STRIBOBr1", 2016, `http://eprint.iacr.org/2016/071`.

[3] `https://is.muni.cz/el/1423/podzim2015/VPL405/59270121/59761464/Logic_Friday_1/?lang=en`.

[4] Yablonsky S. V., *Introduction to Discrete Mathematics*, 6th edition, Vysshaya Shkola Publishers, Moscow, 2010, 384 pp., in Russian.

[5] Savel'ev A. Y., *Introduction to Informatics*, Bauman University Publishers, Moscow, 2001, 328 pp., in Russian.

[6] Borisenko Nikolaj Pavlovich (RU), Vasinev Dmitrij Aleksandrovich (RU), Khoang Dyk Tkho (RU), *Method of Forming s-blocks with Minimum Number of Logic Elements*, Abstract of Invention, RU 2572423 C2, 2016, in Russian.

Side-Channel Attacks Countermeasure Based on Decomposed S-Boxes for Kuznyechik

Tamara Lavrenteeva and Sergey Matveev

The Penza Branch of "STC "Atlas", Russia
lwrt@mail.ru, matveev_sv@tc26.ru

Abstract

The paper describes the Russian cryptographic standard GOST R 34.12-2015 (algorithm Kuznyechik) implementations protected against side-channel attacks. Protection method is based on decomposition of the S-box (algorithm substitution) and allows of a gain in performance and required memory in comparison with universal methods of masking substitution.

Keywords: Side-Channel Attacks, masking S-box, GOST R 34.12-2015

1 Introduction

Attacks using information from side channels (Side Channel Attacks, SCA) consist in analysing the leakage - physical signals in side channels (for example, energy consumption, electromagnetic radiation) observed during the execution of a cryptographic algorithm, and using the statistical dependence of the observed signals on the intermediate values of this algorithm. Some of these values (sensitive values) are correlated with secret data, and therefore restoring information about them allows the adversary to effectively recover keys.

SCA are an important area of research in the last two decades. Starting with the work of P. Kocher,... [7],[8], intensive research has begun on SCA with regard to various cryptographic ciphers and its implementation. Simultaneously with the investigation of the SCA countermeasures were designed to resist such attacks, primarily for the widely used ciphers, such as DES, AES [2], RSA and ECDSA [6].

One of the most popular methods for counteracting side-channel attacks uses masking for the data being processed. The idea of masking is to randomize the intermediate value (the internal state) of the cryptographic algorithm: transform the intermediate value x with the random mask m and use the masked value: $x^m = x * m$. The each mask is a uniform random value. The

mask formed inside the encryption processes and changes periodically and expected unknown to the adversary. As the operation "*" are operations used in a cryptographic algorithm. Usually these are the "exclusive-or" operation, addition or multiplication by module.

The complexity of the masking implementation depends from the properties of maps. Let f be some linear with respect to the operation "*" the function x – a protected value, then $f(x^m) = f(x * m) = f(x) * f(m)$. Most often, an operation "*" is the operation $\oplus$ Boolean addition of two vectors. The calculation of the masked function in this case is easily implemented on any hardware and software platforms. In case f is a non-linear function has the property: $f(x^m) \neq f(x * m) = f(x) * f(m)$, the procedure of masking can be difficult. If a non-linear function has a algebraic structure then it can be used to simplify calculation. For example, S-boxes of AES based on the computation of the inversion in the field. The example implementation of data masking with this representation is in the work [12].

In this paper, we propose a masking method to Russian Federal standard GOST R 34.12-2015 (Kuznyechik) [1]. This method exploiting the representation S-boxes of Kuznyechik like decomposition of linear transformations and substitutions of lower dimension. We examine the single mask protection algorithm for each step of evaluations. It is sufficient to thwart first order DPA. The paper is organized as follows. In Section 2, we provide the short description of Kuznyechik, in Section 3. we give approaches to the masking of the S-box, in Section 4, we describe our masking method for the S-box of Kuznyechik, in Section 5, we give some the safety assessments of the proposed solutions.

2 The short description of Kuznechik

Kuznyechik is XLS block cipher with block length 128 bits. Kuznyechik's round function can be represented as a composition of key addition layer, an S-boxes mapping and a linear transformation layer. The sequence of the following transformations (the notation according to [1]):

E(a)= X[K_{10}]LSX[K_9]...LSX[K_1](a),

where

$K_i \in F_{2^8}^n$, i=1,$\cdots$,10, n=16 - the sequence of iterative keys

X[K](a)=$K \oplus a$

S(y)=S($a_{15}, \cdots, a_0$)=$\pi(a_{15}) \parallel \cdots \parallel \pi(a_0)$,

π - 8-bit substitution,

L(a)=R^{16}(a),

R(a)=R($a_{15}, \cdots, a_0$)=$l(a_{15}, \cdots, a_0) \parallel a_{15} \cdots \parallel y_1$,

l - a linear transformation, where the operations of addition and multiplication are performed in the field $GF(2^8)$.

Iterative keys produced by the rule:

$K_{2i+1} \parallel K_{2i+2}$,$i$=1,2,3,4.

K_1, K_2 - the master key,

$C_i = L(i)$, i=1,2,...,32,

F[k](a_1, a_0)=(LSX[k](a_1) $\oplus$ a_0, a_1).

3 The approaches to the masking of the S-box

If the function f is non-linear, for example, f is substitution, then designing an implementation has some difficulty. Several kinds of methods have been proposed before and we recall some of them hereafter. These methods are used for mapping of masked values of substitution arguments into masked output values.

Method 1. *Re–computation table.*

The core idea is to compute new lookup table with masking an input and an output with random masks. In this case, for the lookup table of S-Box $S(x) : \{0,1\}^n \to \{0,1\}^n$ and two random masks is generated new lookup table $S_{a,b}(x) = S(x \oplus a) \oplus b$, where a is the input mask, b is the output mask, x denote a sensitive variable. The new lookup table is calculated every time after a and b masks changing.

ALGORITHM 1.

Input: S(x) - S-box, $a, b \in V^n$ - mask

Output. $(S_{a,b}(x), b)$

1. for x=0 to 2^{n-1} do
2. $S_{a,b}(x) := S(x \oplus a) \oplus b$
3. end

ALGORITHM 1 requires $n2^n$ bits RAM, $2 \cdot 2^n$ logical operations, $2 \cdot 2^n$ memory transfer (read and write) and one operation for call of the S-box.

Method 2. *Compression of the lookup Table.*

To reduce the RAM requirements was proposed the compression scheme [11].

ALGORITHM 2.

ALGORITHM 2.1 The lookup table creation.

Input: $S(x)$ - S-box, where $S(x) = S_0(x) \parallel S_1(x)$ for all $x \in \{0,1\}^n$, a,$b \in V^{\frac{n}{2}}$, $c \in V^n$ - masks.

Output: $S_{a,b}(x) = S_0(x \oplus a) \oplus S_1(x \oplus b) \oplus c$

1. for x=0 to 2^{n-1} do
2. $S_{a,b}(x) := S_0(x \oplus a) \oplus S_1(x \oplus b) \oplus c$
3. end

ALGORITHM 2.2. Use the lookup table

Input: $S(x), S_{a,b}(x)$ - S-box (original and from ALGORITHM 2.1), $a, b \in V^n$, $c \in V^{\frac{n}{2}}$ - masks, masked input $x_{a,b} = x \oplus a \oplus b$, the new mask $d \in V^{\frac{n}{2}}$

Output: $(a \parallel b, c \parallel (c \oplus d))$

1. $a := S_{a,b}(x_1 \oplus b) \oplus S_1(x_{a,b})$
2. $b := S_{a,b}(x_1 \oplus a) \oplus S_0(x_{a,b}) \oplus d$
3. $c \oplus d$

Output (a|b,C|(C⊕d))

ALGORITHM 2 requires $\frac{n}{2} \cdot 2^n$ bits RAM, $4 \cdot 2^n$ logical operations and $2 \cdot 2^n$ read operation and 2^n write operations. For execution of the S-box requires 4 RAM access and 6 logical operations on binary vectors.

Method 3. *Global lookup table method.*

In this case, one general look-up table is generated for all possible masks. That is, the lookup table $S_{a,b}(x))$ are formed for all x, a, b according to ALGORITHM 1.

In this case, when changing the mask, there is no need to calculate the masked substitution again.

This algorithm requires $n \cdot 2^{3n}$ bits RAM, $2 \cdot 2^{2n}$ logical operations, $2 \cdot 2^{2n}$ read and write operations for generated lookup table. The cryptoalgorithm requires one memory access for any pair masks (a, b).

Method 4. A lookup table computed "on-the-fly"

In this case, a lookup is computed on-the-fly by using mathematical representation of the substitution. Each time the masked value $S(x) \oplus b$ may be computed from the tuple $(x \oplus a, a, b)$ and S-boxes representation algorithm. For this method the complexity of evaluation estimate as $2 \cdot 2^n$ additions of binary vectors and 2^n of memory accesses and 2^n records in memory.

Table 1 shows the complexity of evaluation and memory requirements of masking methods for S-boxes.

Method	RAM	Table creation	Re-masking	The call lookup table
Method 1	2^n	0	$2 \cdot 2^n \cdot A_n + 2^n \cdot R_n + 2n \cdot W_n$	R_n
Method 2	2^{n-1}	0	$4 \cdot 2^n \cdot A_n + 2 \cdot 2^n \cdot R_n + 2^n \cdot W_n$	$4 \cdot R_n + 6 \cdot A_n$
Method 3	2^{3n}	$2 \cdot 2^n(2 \cdot 2^n \cdot A_n + 2^n \cdot R_n + 2^n \cdot W_n)$	0	R_n
Method 4	0	0	0	$2^n \cdot R_n + 2 \cdot 2^n \cdot A_n + 2n \cdot W_n$

Table 1:

Here R_n is n-bit operation for reading from RAM, A_n is $\oplus$ operation for binary addition of n-bit vectors, W_n is n-bit operation for writing to RAM.

Obviously, the complexity of masking for all methods depends on S-boxes dimension. Therefore, if the dimension of S-box used is reduced, it can reduce the complexity of masking.

4 Masking method for the S-box of Kuznechik

Algorithm Kuznechik includes the S-box defined by a fixed 8-bit substitution π. In papers [3], [4], [9], [10] different ways of representing substitution π as a composition of linear and non-linear functions were considered. For our proposal we use a representation of the S-box in compliance with the work [4]. The substitution of π can be represented as $\pi = \omega\pi_0\alpha$, where π_0 is a nonlinear transformation, that is a composition of five nonlinear 4-bit function, and ω and α are the 8-bit linear permutation.

S-box is implemented as follows.

We denote "$\circ$" operation multiplication in finite field $GF(2^4)$ defined by the polynomial $x^4 + x^3 + 1$, "$\cdot$" – multiplication of a vector by a matrix. The algorithm for computing the value of $\pi(a)$ for $a \in V_2^8$ substitution following.

ALGORITHM 3.

1. $(l \parallel r) \leftarrow \alpha \cdot a$
2. $ll \leftarrow \begin{cases} \nu_0(l), \ if \ r = 0 \\ \nu_1(l \circ \chi(r)), \ otherwise \end{cases}$
3. $r \leftarrow \sigma(rr \circ \phi(ll))$
4. $b \leftarrow \omega \cdot (ll \parallel rr)$

In the ALGORITHM 3 $l, r, ll, rr \in F_2^4$, α, ω are the 8-bit linear permutations, χ, ν_0, ν_1, φ, σ are non-linear 4-bit functions. More details of the transformations are given in [4]. Thus, we can consider π as a composition $\pi = \omega\pi_0\alpha$, where π_0 is a substitution defined by steps 2,3, ω and α is a linear maps. Consider the implementation of substitution π_0 mapping the tuple (l, r) to the tuple (ll,rr) in a masked form. To mask of execution for the substitution π_0 it is possible to generate masking look-up tables for 4-bits non-linear function:

$\chi^{a,b}(x) = \chi(x \oplus a) \oplus b$,
$\nu_0^{a,b}(x) = \nu_0(x \oplus a) \oplus b$,
$\nu_1^{a,b}(x) = \nu_1(x \oplus a) \oplus b$,
$\phi^{a,b}(x) = \phi(x \oplus a) \oplus b$,
$\sigma^{a,b}(x) = \sigma(x \oplus a) \oplus b$.

Masking substitution π_0 maps nibbles (l, r)=($l \oplus m_1$, $r \oplus m_0$) to nibbles (ll, rr)=($ll \oplus m_3$, $rr \oplus m_5$) and implemented by the following algorithm. The algorithm uses six random uniform masks (m_0, m_1, m_2, m_3, m_4, m_5) and intermediate 8 bit z value.

ALGORITHM 4.

Input: masked values: ($l \parallel r$), random masks: (m_0, m_1, m_2, m_3, m_4, m_5), lookup tables: $\chi^{m_0,m_2}(x)$, $\nu_0^{m_1,m_3}(x)$, $\nu_1^{m1 \circ m_2,m_3}(x)$, $\varphi^{m_3,m_4}(x)$, $\sigma^{m_0 \circ m_4,m_5}(x)$

Output: ($ll^{m_3} \parallel rr^{m_5}$)

1. $x \leftarrow \nu_0^{m_1,m_3}(l^{m_1}$
2. $y \leftarrow l^{m_1}) \circ \psi^{m_0,m_2}(r^{m_0})$
3. $y \leftarrow y \oplus m_1 \circ \psi^{m_0,m_2}(r^{m_0})$
4. $y \leftarrow y \oplus m_2 \circ l^{m_1}$
5. $y \leftarrow nu_1^{m_1 \circ m_2,m_3}(y)$
6. $z \leftarrow MSB_4(16 - ((r^{m_5} + (\neg m_0 + 1)) \vee 0\text{x}10))$
7. $ll^{m_3} \leftarrow x \oplus x \cdot z \oplus y \cdot z$
8. $y \leftarrow \phi^{m_3,m_4}(ll^{m_3}) \circ r^{m_0}$
9. $y \leftarrow y \oplus m_0 \circ \phi^{m_3,m_4}(ll^{m_3})$
10. $y \leftarrow y \oplus m_4 \circ r^{m_0}$
11. $rr^{m_5} \leftarrow \sigma^{m_0 \circ m_4,m_5}$

In the algorithm 4 "+" and "−" are arithmetic addition and subtraction, the result store in the byte, $\vee$ is the bitwise "or" of two vectors, $\cdot$ is a bitwise logical "and" of two vectors, MSB_n is the operation of taking senior n bit.

Table 2 shows estimates of complexity of masking for S-boxes when it is represented as $\pi = \omega\pi_0\alpha$ for various substitution masking methods.

Method	RAM	Table creation	Re-masking	The call lookup table
1	$5\frac{n}{2}2^{\frac{n}{2}}$	0	$5(2\cdot 2^{\frac{n}{2}}A_{\frac{n}{2}}+2^{\frac{n}{2}}R_{\frac{n}{2}}+2^{\frac{n}{2}}W_{\frac{n}{2}})$	$R_{\frac{n}{2}}$
2	$5\frac{n}{2}2^{\frac{n}{2}-1}$	0	$5(4\cdot 2^{\frac{n}{2}}A_{\frac{n}{2}}+2\cdot 2^{\frac{n}{2}}R_{\frac{n}{2}}+2^{\frac{n}{2}}W_{\frac{n}{2}})$	$5(4R_{\frac{n}{2}}+6A_{\frac{n}{2}})$
3	$5\frac{n}{2}2^{3\frac{n}{2}}$	$5(2\cdot 2^{\frac{n}{2}}(2\cdot 2^{\frac{n}{2}}A_{\frac{n}{2}}+2^{\frac{n}{2}}R_{\frac{n}{2}}+2^{\frac{n}{2}}W_{\frac{n}{2}})$	0	$5\cdot R_{n/2}$
4	0	0	0	$5\cdot(2^{n/2}\cdot R_{n/2}+2\cdot 2^{n/2}\cdot A_{n/2}+2^{n/2}\cdot W_{n/2})$

Table 2:

For approximating complexity our algorithm puttin $A_n = R_n = W_n$. Table 3 shows comparison complexity masking method with (1) and without (2) our proposal. We can see that we obtain reduces the complexity more than 3 times for all methods.

Method	RAM		Table creation		Re-masking		The call lookup table	
	1	2	1	2	1	2	1	2
1	$5\cdot 2^6$	2^{11}	0	0	$5\cdot 2^6A_4$	$2^{10}A_8$	R_4	R_8
2	$5\cdot 2^5$	2^{10}	0	0	$5\cdot 2^6A_4$	$2^{10}A_8$	$50A_4$	$10A_8$
3	$5\cdot 2^{14}$	2^{27}	$5\cdot 2^{14}A_4$	$2^{27}A_8$	0	0	$5R_4$	R_8
4	0	0	0	0	0	0	$5\cdot 2^6A_4$	$2^{10}A_8$

Table 3:

5 Security Analysis

Consider one round of the encryption algorithm. On the each step of the algorithm the intermediate result computed from the some sensitive variable and masks. On the each step algorithm the intermediate result computed from the some sensitive variable and masks (m_0, m_1, m_2, m_3, m_4, m_5). All masks are independent random values. The observed input and output values at each step of the algorithm 4 are shown in Table 4.

Step	Intermediate values	Result	Step	Intermediate values	Result
1	l^{m_1}, m_3	$\nu_0(l) \oplus m_3$	7	$\nu_0(l) \oplus m_3$, $\nu_1(l \circ \chi(r)) \oplus m_3$, z	$ll \oplus m_3$
2	l^{m_1}, r^{m_0}, m_2	$l \circ \chi(r) \oplus m_1 \circ m_2 \oplus m_1 \circ \chi^{m_0,m_2}(r^{m_0}) \oplus m_2 \circ l^{m_1}$	8	m_0, m_4, r^{m_0}, ll^{m_3},	$\varphi(ll) \circ r \oplus m_0 \circ m_4 \oplus m_0 \circ \varphi^{m_3,m_4}(ll^{m_3}) \oplus m_4 \circ r^{m_0}$
3	$l \circ \chi(r) \oplus m_1 \circ m_2 \oplus m_1 \circ \chi^{m_0,m_2}(r^{m_0}) \oplus$, $m_2 \circ l^{m_1}$, $m1 \circ \chi^{m_0,m_2}(r^{m_0})$	$l \circ \chi(r) \oplus m_1 \circ m_2 \oplus m_2 \circ l^{m_1}$	9	ll^{m_3}, $m_0 \circ \varphi^{m_3,m_4}(ll^{m_3})$, $\varphi(ll) \circ r \oplus m_0 \circ m_4 \oplus m_0 \circ \varphi^{m_3,m_4}(ll^{m_3}) \oplus m_4 \circ r^{m_0}$	$\varphi(ll) \circ r \oplus m_0 \circ m_4 \oplus m_4 \circ r^{m_0}$
4	$l \circ \chi(r) \oplus m_1 \circ m_2 \oplus m_2 \circ l^{m_1}$, $m_2 \circ l^{m_1}$	$l \circ \chi(r) \oplus m_1 \circ m_2$	10	m_4, r^{m_0}, $\varphi(ll) \circ r \oplus m_0 \circ m_4 \oplus m_4 \circ r^{m_0}$	$\varphi(ll) \circ r \oplus m_0 \circ m_4$
5	$l \circ \chi(r) \oplus m_1 \circ m_2$, m_3	$\nu_1(l \circ \chi(r)) \oplus m_3$	11	m_5, $\varphi(ll) \circ r \oplus m_0 \circ (m_4)$	$\sigma\varphi(ll) \circ r \oplus m_5$
6	$\neg m_0$, r^{m_0}	$z = (0,0,0,0)$ - if $r = 0$, $z = (1,1,1,1)$ - otherwise			

Table 4:

It is easy to see that the proposed scheme provides protection against attacks of the first order.

At step 6 of the algorithm is possible to use the "zero value" attack for z variable. We estimate a possible reduction in the complexity of key recovery compared to exhaustive search for the first iteration of the algorithm. We assume that we define the first iterative key with a length of 128 bit.

Consider the following algorithm for the key recovery. For each of the 16 lookup transformation S we independently determine the appearance of zero values of r on the side channel. If we have defined, r=0, then we will try all possible values of l. In total $s_1 = 2^4$ variants. Otherwise, we will try all possible values of l and all values of $r \neq 0$. Tatal $s_2 = 2^8 - 1$ variants. The probability of an event $r = 0$ equals $p_1 = 2^{-4}$. The probability of the event $r \neq 0$ is $p_2 = (1 - 2^{-4})$.

The average complexity of the attack is

$S = \sum_{i=0}^{16} \begin{pmatrix} 16 \\ i \end{pmatrix} \left(s_1^i p_1^i + s_2^{16-i} p_2^{16-i}\right)$

Finally, we obtain $S = 2^{127.994}$. The complexity for for exhaustive search is S=2^{128}. We have reducing the complexity of the attack estimate at 1.004 times.

6 Conclusion

In this paper we demonstrate possibility to use a decomposition of the S-box of Kuznyechik for decrease complexity some of masking methods for protecting the algorithm from side channel attacks.

It is shown that in comparison with universal methods of masking, one can get a gain in performance and required memory more than 3 times. At the same time, the potential threat from implementing a zero-value attack can reduce the complexity of the key recovering by no more than 1.004 times.

The proposed method uses specific properties of the S-box decomposition. For future work, we plan to apply another decomposition methods for Kuznyechik to protect algorithm against side-channel attacks.

References

[1] *GOST R 34.12-2015. National standard of the Russian Federation. Information technology Cryptographic data security. Block Cipher*, 2015, In Russian.

[2] Akkar M.-L., Giraud C., "An Implementation of DES and AES, Secure against Some Attacks", *LNCS*, CHES 2001., **2162**, ред. Goos G, Harmanis J., van Leeuwen J., Springer, Berlin, Heidelberg, 2001, 309–318.

[3] Biryukov A., Perrin L., Udovenko A., *The Secret Structure of the S-Box of Streebog, Kuznechik and Stribob*, Crypto ePrint Archive `http://eprin.iacr.org/2015/812.pdf`.

[4] Biryukov A., Perrin L., Udovenko A., *Reverse-engineering the S-box of streebog, kuznyechik and STRIBOBr1*, Cryptology ePrint Archive `http://eprin.iacr.org/2016/071.pdf`.

[5] Chari S.,Jutla C. S., Rao J. R., Rohatgi P., "Towards sound approaches to counteract power-analysis attacks", *LNCS*, CRYPTO'99, **1666**, ред. Wiener M., Springer, Springer, Heidelberg, 1999, 398–412.

[6] Coron J., "Resistance against differential power analysis for elliptic curve cryptosystems", *LNCS*, CHES'99, **1717**, Springer, Springer, Heidelberg, 1999, 292–302.

[7] Kocher P., Jaffe J., Jun B., "Differential power analysis", *LNCS*, CRYPTO'99, **1666**, ред. Wiener M., Springer, Springer, Heidelberg, 1999, 388–397.

[8] Kocher P., "Timing attacks on implementations of Diffie-Hellmann, RSA, DSS, and other systems", *LNCS*, CRYPTO'96, **1109**, Springer, Springer, Heidelberg, 1996, 104–113.

[9] Perrin L., *Partitions in the S-Box of Streebog and Kuznyechik*, Cryptology ePrint Archive `http://eprint.iacr.org/2019/092.pdf`.

[10] Perrin L., Udovenko A., "Exponential S-Boxes: a Link Between the S-Boxes of BelT and Kuznyechik/Streebog", *IACR Transactions on Symmetric Cryptology ISSN 2519-173X*, **2016**, No 2, 99–124.

[11] Praveen Kumar Vadnala, "Time-Memory Trade-Offs for Side-Channel Resistant Implementations of Block Ciphers", *LNCS*, CT-RCA 2017, **10159**, ред. Helena Handschuh, Springer, Springer, Heidelberg, 2017, 115–130.

[12] Trichina E., Korkishko L., *Secure and efficient AES software implementation for smart cards*, Cryptology ePrint Archive `http://eprint.iacr.org/2004-149.pdf`.

On the Guaranteed Number of Activations in XS-circuits

Sergey Agievich

Research Institute for Applied Problems of Mathematics and Informatics
Belarusian State University
agievich@bsu.by

Abstract

XS-circuits describe cryptographic primitives that utilize 2 operations on binary words of fixed length: X) bitwise modulo 2 addition and S) substitution. The words are interpreted as elements of a field of characteristic 2. In this paper, we develop a model of XS-circuits according to which several instances of a simple round circuit containing only one S operation are linked together and form a compound circuit called a cascade. S operations of a cascade are interpreted as independent round oracles. When a cascade processes a pair of different inputs, some round oracles get different queries, these oracles are activated. The more activations, the higher security guarantees against differential cryptanalysis the cascade provides. We introduce the notion of the guaranteed number of activations, that is, the minimum number of activations over all choices of the base field, round oracles and pairs of inputs. We show that the guaranteed number of activations is related to the minimum distance of the linear code associated with the cascade. It is also related to the minimum number of occurrences of units in segments of binary linear recurrence sequences whose characteristic polynomial is determined by the round circuit. We provide an algorithm for calculating the guaranteed number of activations. This algorithm can also be used to deal with linear activations related to linear cryptanalysis.

Keywords: circuit, differential cryptanalysis, linear cryptanalysis, linear code, linear recurrence sequence. .

1 Introduction

XS-circuits describe cryptographic primitives that utilize 2 operations on binary words of fixed length: X) bitwise modulo 2 addition and S) substitution. A circuit may describe a block cipher when instantiating S with key-dependent round functions which usually have a complicated internal structure being circuits of the same (of smaller word length) or other types. Or a circuit may describe an encryption or authentication mode when S is a keyed permutation of a block cipher. One of the directions here is constructing wide-block and variable-input-length ciphers, that is, extending the block length of the underlying cipher to some fixed or even arbitrary length.

We interpret binary words that are processed in an XS-circuit as elements of a field of characteristic 2. A circuit becomes arithmetic if all its operations S are instantiated using only the addition (actually, X) and multiplication in the base field. Arithmetic circuits designed for symmetric cryptography are demanded in universal ZK-proof systems, especially if the circuits have low multiplicative complexity (an example of the approach can be found in [2]). We see another area of application of XS-circuits.

In this paper, we follow the model of XS-circuits proposed in [1]. According to this model, several instances of a simple round circuit containing only one S operation are linked together and form a compound circuit called a cascade. S operations of a cascade are interpreted as independent round oracles. We extensively use notions and notation from [1]. In particular, the notion of regular circuits that are in a sense the best elementary circuits and the only ones worth considering when constructing cascades.

When a cascade processes a pair of different inputs, some round oracles get different queries, these oracles are called *activated*. The more activations, the higher security guarantees against differential cryptanalysis the cascade provides.

In Section 2 we introduce the notion of the guaranteed number of activations, that is, the minimum number of activations over all choices of the base field, round oracles and pairs of inputs. In Section 3 we show that the guaranteed number of activations is related to the minimum distance of the linear code associated with the target cascade. This number is also related to the minimum number of occurrences of units in segments of binary linear recurrence sequences whose characteristic polynomial is determined by the round circuit. This is shown in Section 4. Finally, in Section 5 we provide an algorithm for calculating the guaranteed number of activations. This algorithm can also be used to deal with linear activations related to linear cryptanalysis.

Bringing the problem of lower bounding the number of activations to the context of coding theory and showing how to solve it algorithmically, we introduce a systematic approach for constructing sound cryptographic mappings. Interestingly, another systematic approach of this kind, the so-called Wide trail strategy, also relates to coding theory. This approach was proposed in [5, 6] and was implemented in numerous block ciphers including AES and Kuznyechik (see [3] for a fairly complete list).

The Wide trail strategy allows to achieve a high activation rate, close to 1/2, when MDS (Maximum Distance Separable) codes are used to build a diffusion layer. The drawback of the strategy is that the layer becomes quite

complicated and usually has to be implemented through a table lookup. For comparison, the diffusion layer of an XS-circuit can be made very simple. However, S operations of the circuit cannot be applied in parallel although this is allowed by the Wide trail strategy.

2 Preliminaries

Let (a, B, c) be a regular XS-circuit of order n (see [1] for definitions and further details). We assume that the circuit is in the first canonical form, that is, a is a nonzero column vector, B is a Frobenius cell, $c = (0, \ldots, 0, 1)$. Denote by b the last column of B. All the vectors a, b, c are binary of dimension n.

Instantiating the circuit over a field $\mathbb{F}$ of characteristic 2 and substituting an oracle $S\colon \mathbb{F} \to \mathbb{F}$ for the operation S, we get the mapping

$$(a, B, c)[S]\colon \mathbb{F}^n \to \mathbb{F}^n,\ (x_1, x_2, \ldots, x_n) \mapsto (x_2, x_3, \ldots, x_n, x_{n+1}),$$
$$x_{n+1} = (x_1, x_2, \ldots, x_n)b + S((x_1, x_2, \ldots, x_n)a).$$

Let $(a, B, c)^t$ be the t-round cascade built by connecting t instances of (a, B, c). The cascade utilizes t operations S. Instantiating these operations by oracles $S_1, \ldots, S_t$, we obtain the mapping $(a, B, c)^t[S_1, \ldots, S_t]$. It may be described algorithmically as follows: having received an input $(x_1, x_2, \ldots, x_n)$, the sequence

$$x_{\tau+n} = (x_\tau, \ldots, x_{\tau+n-1})b + S_\tau((x_\tau, \ldots, x_{\tau+n-1})a), \quad \tau = 1, 2, \ldots, t,$$

is calculated and the vector $(x_{t+1}, \ldots, x_{t+n})$ is returned as the output.

Example 1 (GFN1)**.** *The* GFN1 *family of* XS*-circuits was introduced in [14]. The circuit of dimension $n \geq 2$ has the second canonical form: $a = (1, 0, \ldots, 0)^T$, B is a Frobenius cell with $b = a$, $c = a^T$. Replacing (a, B, c) with*

$$(B^{-1}a, B^{-1}BB, cB) = ((0, \ldots, 0, 1)^T, B, (0, \ldots, 0, 1)),$$

we obtain the first canonical form for which

$$x_{\tau+n} = x_\tau + S_\tau(x_{\tau+n-1}), \quad \tau = 1, 2, \ldots.$$

□

Let us suppose now that the cascade processes not one but two inputs simultaneously. From there, $(x_1, x_2, \ldots, x_n)$ is the X-difference of input vectors

and $(x_{\tau+1}, \ldots, x_{\tau+n})$ is the difference of the τth round outputs. See [1, Section 4] for further details. There differences are denoted using the symbol Δ but here we simplify the notation.

The difference u_τ at the input of S_τ has the form $(x_\tau, \ldots, x_{\tau+n-1})a$. The corresponding output difference v_τ can be written as $(x_\tau, \ldots, x_{\tau+n-1})b + x_{\tau+n}$. Due to the bijectivity of S_τ, the equality $u_\tau = 0$ holds if and only if $v_\tau = 0$. In other words,

$$(x_\tau, \ldots, x_{\tau+n-1})a = 0 \Leftrightarrow (x_\tau, \ldots, x_{\tau+n-1})b + x_{\tau+n} = 0.$$

Let us construct a matrix $G = G(n, a, b, t)$ of dimensions $(t+n) \times 2t$. Its columns go in pairs, the τth pair has the form:

$$\left.\begin{matrix} 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \end{matrix}\right\} \tau - 1$$
$$\left.\begin{matrix} a & b \\ 0 & 1 \end{matrix}\right\} n + 1$$
$$\left.\begin{matrix} 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \end{matrix}\right\} t - \tau$$

With this,

$$(x_1, x_2, \ldots, x_{t+n})G = (u_1, v_1, u_2, v_2, \ldots, u_t, v_t).$$

We require that in each pair (u_τ, v_τ) both elements are either zero or nonzero together. Denote by $\mathcal{W}$ the set of all vectors

$$w = (u_1, v_1, \ldots, u_t, v_t) = xG, \quad x \in \mathbb{F}^{t+n},$$

for which the requirement holds. The set $\mathcal{W}$ is completely determined by the base field $\mathbb{F}$, the vectors a, b and the number of rounds t. The zero vector obviously belongs to $\mathcal{W}$.

We call the situation when $(u_\tau, v_\tau) \neq (0, 0)$ the *activation* of S_τ. Let $\mathrm{wt}_2(w)$ be the total number of activations, that is, nonzero pairs (u_τ, v_τ), in the vector w.

For $t \geq n$ we are interested in the quantity

$$d(\mathcal{W}) = \min_{w \in \mathcal{W}, w \neq 0} \mathrm{wt}_2(w).$$

It is the minimum number of activations when applying the mappings $(a, B, c)^t[S_1, \ldots, S_t]$ to pairs of different vectors from $\mathbb{F}^n$. Note that the minimization covers all admissible tuples $(S_1, \ldots, S_t)$ and all admissible input pairs.

For $t < n$ we set $d(\mathcal{W}) = 0$. This reflects the fact that as long as the number of rounds is less than the dimension of the circuit, it is possible to avoid activations by manipulating the initial diffirence $(x_1, \ldots, x_n) \neq 0$ (see [1, Section 8]).

The quantity $d(\mathcal{W})$ can also be denoted as $d(\mathbb{F}, n, a, b, t)$ implying that $\mathcal{W}$ is uniquely determined by the parameters $(\mathbb{F}, n, a, b, t)$. Let

$$d(n, a, b, t) = \min_{\mathbb{F}} d(\mathbb{F}, n, a, b, t),$$

where the minimum is taken over all fields of characteristic 2. Any such field is an extension of $\mathbb{F}_2$ and therefore

$$d(n, a, b, t) \leq d(\mathbb{F}, n, a, b, t) \leq d(\mathbb{F}_2, n, a, b, t).$$

The cascade $(a, B, c)^t$ guarantees at least $d(n, a, b, t)$ activations regardless of the choice of $\mathbb{F}$, round oracles and input pairs. We call $d(n, a, b, t)$ the *guaranteed* number of activations.

3 Connection to the linear codes

The set $\mathcal{W}$ is a subset of the vector space

$$\mathcal{C} = \{xG \colon x \in \mathbb{F}^{t+n}\} \subseteq \mathbb{F}^{2t}.$$

The following lemma means that for $t \geq n$ the space $\mathcal{C}$ has dimension $t + n$ and, therefore, it is a linear code with the parameters $[2t, t + n]$.

Lemma 1. *Let vectors a and b define a regular* XS*-circuit of the first canonical form of dimension n. If $t \geq n$, then the matrix $G = G(n, a, b, t)$ has full rank:* $\operatorname{rank} G = t + n$.

Proof. Let us associate with the first two columns of G the polynomials $a(\lambda) = \sum_{i=0}^{n-1} a_i \lambda^i$ and $f_B(\lambda) = \lambda^n + \sum_{i=0}^{n-1} b_i \lambda^i$. Here a_i and b_i are coordinates of a and b respectively. We follow the notation introduced in [1, Section 7]. Note that Theorem 9 of the cited paper states that for a regular XS-circuit the polynomials $a(\lambda)$ and $f_B(\lambda)$ are coprime.

The monomial λ^i in $a(\lambda)$ marks the position in the first column in which the coefficient a_i is located. The same holds for $f_B(\lambda)$ and the second column. In general, the τth pair of columns is described by the polynomials $\lambda^{\tau-1} a(\lambda)$ and $\lambda^{\tau-1} f_B(\lambda)$.

The first $2t$ columns of G are linearly dependent if there exist nonzero polynomials $p(\lambda)$ and $q(\lambda)$ whose degrees are less than t and which satisfy

$$p(\lambda)a(\lambda) + q(\lambda)f_B(\lambda) = 0.$$

For $t = n$, since $a(\lambda)$ and $f_B(\lambda)$ are coprime, there are no suitable polynomials $p(\lambda)$, $q(\lambda)$ and the matrix G has full rank.

The first linear dependence appears in G at $t = n + 1$ when choosing $p(\lambda) = f_B(\lambda)$ and $q(\lambda) = a(\lambda)$. The penultimate column becomes dependent on the previous ones. But the last column remains independent, since it is the only one containing 1 in the last row. Thus, $\operatorname{rank} G = 2n + 1 = t + n$ and G is again full-ranked.

The argument can be repeated: each new pair of columns adds 1 to the rank of G. Full rank is preserved, which was to be proven. □

The minimum distance of $\mathcal{C}$ is the quantity

$$d(\mathcal{C}) = \min_{w \in \mathcal{C}, w \neq 0} \mathrm{wt}(w),$$

where $\mathrm{wt}(w)$ is the Hamming weight of w. According to the Singleton bound (see, for example, [10]),

$$d(\mathcal{C}) \leq 2t + 1 - (t + n) = t - n + 1.$$

Since $\mathrm{wt}(w)/2 \leq \mathrm{wt}_2(w) \leq \mathrm{wt}(w)$ and $\mathcal{W} \subseteq \mathcal{C}$, it holds that

$$d(\mathcal{C})/2 \leq d(\mathcal{W}) \leq d(\mathcal{C}).$$

In particular, $d(\mathcal{W}) \leq t - n + 1$. This estimate means that over $t \geq n$ rounds we cannot guarantee more than $t - n + 1$ activations. Further we are interested in lower bounds for $d(\mathcal{W})$.

Let $t \geq n$ and, therefore, $\operatorname{rank} G = t + n$ by Lemma 1. Suppose that when processing a nonzero input difference using some round oracles, activations occur only in rounds whose numbers belong to a set $\mathcal{T} \subseteq \{1, 2, \ldots, t\}$. We call $\mathcal{T}$ the *activation profile*. Following this profile, let us divide G into two parts: G_0 and G_1. The matrix G_1 consists of pairs of columns whose numbers are in $\mathcal{T}$ and G_0 consists of the remaining columns. By construction, there exists a nonzero vector $x \in \mathbb{F}^{t+n}$ such that $xG_0 = 0$ and xG_1 does not contain zeros. This means that the partition (G_0, G_1) is feasible in the sense of the following definition.

Definition. *Let G_0 and G_1 be matrices composed of different pairs of columns of G. The partition (G_0, G_1) is feasible if*

1) $\operatorname{rank} G_0 < t + n$;

2) $\operatorname{rank}(G_0 \mid g) > \operatorname{rank} G_0$ *for each column g of G_1.*

Indeed, if $\operatorname{rank} G_0 = t + n$, then from $xG_0 = 0$ it follows that $x = 0$ which contradicts the construction. And if $\operatorname{rank}(G_0 \mid g) = \operatorname{rank} G_0$, then from $xG_0 = 0$ it follows that $xg = 0$. The latter means that xG_1 contains zero, again a contradiction.

In the following lemma, we show that feasibility of a partition (G_0, G_1) is not only necessary but also a sufficient condition for the feasibility of the underlying activation profile.

Lemma 2. *Let vectors a and b define a regular* XS*-circuit of the first canonical form of dimension n. Let $t \geq n$ and k be the maximum number of pairs of columns in the matrix G_0 where the maximum is taken over all feasible partitions (G_0, G_1) of $G = G(n, a, b, t)$. Then*

$$d(n, a, b, t) = t - k.$$

Proof. Let (G_0, G_1) be a feasible partition of G. It is necessary to prove that there exists an extension $\mathbb{F}$ of the field $\mathbb{F}_2$ and a vector $x \in \mathbb{F}^{t+n}$ such that $xG_0 = 0$ and xG_1 does not contain zeros.

The set

$$L = \{xG_1 \colon x \in \mathbb{F}^{t+n}, xG_0 = 0\} \subseteq \mathbb{F}^{2(t-k)}$$

is a vector space of dimension $r = t + n - \operatorname{rank} G_0$. It can be written as

$$L = \{yP \colon y \in \mathbb{F}^r\},$$

where P is a binary matrix of dimensions $r \times 2(t - k)$. The matrix P does not contain a zero column due to the second restriction on the feasibility of the partition (G_0, G_1).

Suppose that L does not contain a vector without zero coordinates. Then we choose an arbitrary vector $yP \in L$, build an extension $\mathbb{F}'$ of the field $\mathbb{F}$ and extend y to a vector y' of the same dimension but over $\mathbb{F}'$. We construct y' in such way that a particular zero coordinate of yP becomes nonzero in $y'P$ while nonzero coordinates of yP remain nonzero in $y'P$. After constructing the pair $(\mathbb{F}', y')$ we interpret it as $(\mathbb{F}, y)$ and repeat the extension until we get the vector yP without zeros. It remains to show how to extend y to y'.

Define $\mathbb{F}'$ as an extension of $\mathbb{F}$ of degree 2. Without loss of generality, let elements of $\mathbb{F}'$ be $(m + 1)$-bit words $\alpha = \alpha_1 \ldots \alpha_m \alpha_{m+1}$ and $\alpha \in \mathbb{F}$ if and only if $\alpha_{m+1} = 0$. Let the addition in $\mathbb{F}'$ be the usual XOR. The extension of y consists in setting the last (zero) bits of its coordinates. Let β be a vector composed of these bits. Since P does not contain zero columns, it is possible to choose β so that a particular coordinate of βP is nonzero. The corresponding coordinate of $y'P$ is also nonzero. Moreover, if a certain

coordinate yP is nonzero, then the corresponding coordinate $y'P$ remains nonzero. That was to be proven. □

Remark 1. *The minimum distance of the code* $\mathcal{C} = \{xG\}$ *can also be defined as* $d(\mathcal{C}) = t - k$*, where* k *is the maximum number of columns in* G_0 *and the maximum is taken over all feasible partitions* (G_0, G_1) *of* G *(see, for example, [8, Theorem 1.4.5]). The difference is in changing the partitioning restrictions. Now* G_i *not necessarily consists of pairs of related columns, the requirement* $\operatorname{rank} G_0 < t + n$ *is preserved, but the requirement* $\operatorname{rank}(G_0 \mid g) > \operatorname{rank} G_0$ *becomes redundant.*

Remark 2. *Let* $\operatorname{rank} G_0 = t + n - 1$*. Then in the proof above, the vector space* L *has dimension* 1 *and the matrix* P *becomes the row vector* $(1, 1, \ldots, 1)$*. This means that with* $\mathbb{F} = \mathbb{F}_2$ *there exists a nonzero* $x \in \mathbb{F}^{t+n}$ *such that* $xG_0 = 0$ *and* $xG_1 = (1, 1, \ldots, 1)$*. In other words, the activation profile associated with the partition* (G_0, G_1) *is feasible over* $\mathbb{F}_2$*. Moreover, as we see below, this profile is a segment of a linear recurrence sequence over* $\mathbb{F}_2$*.*

4 The case $\mathbb{F} = \mathbb{F}_2$

In the case $\mathbb{F} = \mathbb{F}_2$, the condition $u_\tau = 0 \Leftrightarrow v_\tau = 0$ is eqiuvalent to $u_\tau = v_\tau$. With this,

$$x_{\tau+n} = (x_\tau, \ldots, x_{\tau+n-1})(a + b), \quad \tau = 1, 2, \ldots, t,$$

that is, the sequence $(x_1, \ldots, x_{t+n})$ is a segment of a nonzero linear recurrence sequence (LRS) over $\mathbb{F}_2$. The characteristic polynomial of the sequence is

$$f(\lambda) = \lambda^n + f_{n-1}\lambda^{n-1} + \ldots + f_1\lambda + f_0, \quad (f_0, f_1, \ldots, f_{n-1}) = a + b.$$

The vectors $(x_\tau, \ldots, x_{\tau+n-1})$, $\tau = 1, 2, \ldots$, stand as states of the linear feedback shift register (LFSR) associated with $f(\lambda)$. When choosing the first bits of LFSR states, we get the sequence (x_τ), and when choosing the linear combinations $(x_\tau, \ldots, x_{\tau+n-1})a$, we get the sequence (u_τ). The latter sequence is also a LRS with the same characteristic polynomial $f(\lambda)$.

The sequence (u_τ) is nonzero. Indeed, the underlying XS-circuit is regular and for any nonzero input difference $(x_1, \ldots, x_n)$ at least one activation must occur over the first n rounds (see the discussion before Theorem 10 in [1]) which means that $(u_1, \ldots, u_n) \neq (0, \ldots, 0)$.

For the same reason, $f_0 = 1$ and the sequences (x_τ), (u_τ) are purely periodic. Indeed, otherwise, a nonzero input difference $(x_1, \ldots, x_n) =$

$(1, 0, \ldots, 0)$ induces a zero difference after $n - 1$ rounds, which is impossible due to the regularity.

The number of activations over t rounds is the number of nonzero elements (units) in the segment $(u_1, \ldots, u_t)$. We can use known results on the number of occurrences of particular elements in segments of LRS. Let r be the least period of (u_τ), R be the order of f (the maximum least period of nonzero LRS with the characteristic polynomial f). Then according to Theorems 8.82 and 8.85 from [9], the number of activation is at least

$$\frac{t}{2} - 2^{n/2-1} \left(\frac{r}{R}\right)^{1/2} \left(t_0 + \frac{2}{\pi} \log r + \frac{2}{5} + \frac{t_1}{r}\right).$$

Here t_0 and t_1 are respectively the quotient and remainder when dividing t by r. If $t_1 = 0$, then only the term t_0 can be left in the last brackets.

It makes sense to apply the estimate above only for large n, t and r. In practice, these parameters are small and the minimum number of activations can be found by exhaustive search over all LRS profiles $(u_1, \ldots, u_t)$ in time of order $2^n t$.

Example 2 (SMS4)**.** *The* SMS4 *circuit is used in the block cipher of the same name (which is often shortened to SM4). See [7] for details of the cipher and [1] for details of the circuit.*

The circuit is already in the first canonical form, its dimension is 4, the characteristic polynomial $f(\lambda) = \lambda^4 + \lambda^3 + \lambda^2 + \lambda + 1$*. The polynomial* $f(\lambda)$ *is irreducible of order* 5*. Therefore, the least period of* (u_τ) *equals 5.*

The minimum number of activations is achieved on the start segments of the following LRS:

$$0, 0, 0, 1, 1,\ \ 0, 0, 0, 1, 1,\ \ \ldots$$

If $t = 5t_0 + t_1$, $0 \le t_1 < 5$*, then this number is*

$$\begin{cases} 2t_0, & t_1 = 0, 1, 2, 3, \\ 2t_0 + 1, & t_1 = 4. \end{cases}$$

□

Example 3 (GFN1, continued)**.** *Let us continue Example 1 and consider the* GFN1 *circuit of dimension* n *in the first canonical form. For this circuit,* $a = (0, 0, \ldots, 0, 1)^T$, $b = (1, 0, \ldots, 0, 0)^T$ *and* $f(\lambda) = \lambda^n + \lambda^{n-1} + 1$.

For $n = 2, 3, 4$*, the polynomial* $f(\lambda)$ *is primitive. The least period of* (u_τ) *equals* $r = 2^n - 1$ *and every full period* $(u_1, \ldots, u_r)$ *contains exactly* 2^{n-1} *units. Therefore,*

$$d(\mathbb{F}_2, n, a, b, 2^n - 1) = 2^{n-1}$$

and the activation rate over $2^n - 1$ *rounds can potentially achieve the value* $2^{n-1}/(2^n - 1) > 1/2$. *This value is indeed achieved for* $n = 2, 3$ *but, as we show later, not for* $n = 4$. □

5 The algorithm

The following algorithm summarizes our constructions and reasoning.

Algorithm GNA (the guaranteed number of activations)

Input: (n, a, b, t), where a and b are binary vectors of dimension n that define a regular XS-circuit in the first canonical form, t is a number of rounds.

Output: $d(n, a, b, t)$, the guaranteed number of activations over t rounds of the input circuit.

Steps:

1. If $t < n$, then return 0. If $t = n$, return 1.
2. Construct the matrix $G = G(n, a, b, t)$ as explained in Section 3. The dimensions of G are $(t + n) \times 2t$, $\operatorname{rank} G = t + n$. The columns of G are grouped in pairs.
3. Calculate $d(\mathbb{F}_2, n, a, b, t)$ as described in Section 4 and set $k \leftarrow t - d(\mathbb{F}_2, n, a, b, t)$.
4. Make a list of all possible partitions of G into submatrices G_0 and G_1 such that G_0 contains exactly $k + 1$ pairs of columns of G.
5. For each partition (G_0, G_1):
   (a) if $\operatorname{rank} G_0 \geq t + n - 1$, then continue (go to the end of the loop);
   (b) if there is a column g in G_1 such that $\operatorname{rank}(G_0 \mid g) = \operatorname{rank} G_0$, then continue;
   (c) set $k \leftarrow k + 1$ and go to Step 4.
6. Return $t - k$.

Theorem. *The algorithm GNA is correct.*

Proof. A direct consequence of Lemma 2 and Remark 2.

In Step 5a of the algorithm, we skip the case $\operatorname{rank} G_0 = t + n - 1$ because in this case the activation profile associated with the partition (G_0, G_1) is feasible over $\mathbb{F}_2$ and the initial bound $d(\mathbb{F}_2, n, a, b, t)$ for $d(n, a, b, t)$ cannot be strengthened. □

The algorithm GNA gives us the guaranteed number of *differential* activations. We can easily adapt the algorithm to deal with *linear* activations (see [1, Section 9]). To do this, we pass from (a, B, c) to the dual circuit (c^T, B^T, a^T) and determine vectors a' and b' that define its first canonical form (we only need to determine a', since $b' = b$). The quantity $\mathrm{GNA}(n, a', b', t)$ is the guaranteed number of linear activations.

Example 4 (SMS4, continued)**.** *For* SMS4 *its dual has the same first canonical form. So the guaranteed numbers of differential and linear activations are the same. The outputs of GNA against* SMS4 *for* $t \leq 32$ *coincide with the estimates of Example 2. Thus, the activation rate close to* $2/5$ *is achieved. In particular, the guaranteed number of activations over 32 rounds (exactly the case of the block cipher SMS4/SM4) is 12.*

Note that here we are processing the abstract SMS4 *circuit, not its instantiation in SMS4/SM4. In this instantiation,* S *operations are constructed using round keys, table S-boxes, rotations and XORs of binary words. Lower bounds on the number of active S-boxes (not activations / active rounds) in SMS4/SM4 can be found in [11, 12, 13].*

Iterating over $\binom{t}{k+1}$ *partitions in Step 4 of GNA can be simplified. For example, in the case of* SMS4*, if any 4 of 5 consecutive pairs of columns fall into* G_0*, then the corresponding partition is not feasible and can be immediately rejected. Indeed, the 5 consecutive pairs of columns are linearly dependent while 4 pairs are not (it follows from the same reasoning as in the proof of Lemma 1). Therefore, a pair not included in* G_0 *contains a column* g *which is linearly expressed through the columns of* G_0 *and, therefore, the second condition of feasibility is violated.* □

Example 5 (GFN1, continued)**.** *An* GFN1 *circuit of arbitrary dimension is self-dual:* $(a, B, c) = (c^T, B^T, a^T)$*. Therefore, a bound on differential activations is also a bound on linear activations.*

For the circuit of dimension $n = 4$*, GNA gives* 7 *activations over* 15 *rounds. It is one less than estimated in Example 3 through LRS profiles. The optimal activation profile found by GNA looks as follows:*

$$0001111010\mathbf{0100}.$$

It differs from the related LRS profile

$$0001111010\mathbf{11001}$$

starting from the 10th round. The LRS profile gives 3 activations after the fork while the optimal profile gives only 2 activations. □

Example 6 (activation times)**.** *The ith activation time, ρ_i, is the minimum number of rounds that guarantees i activations (see [1, Section 8]). In the next table, we present the values ρ_i for* GFN1 *of dimension $n = 4$ and for* SMS4*. We calculate ρ_i using the GNA algorithm.*

i	1	2	3	4	5	6	7	8	9	10	11	12
ρ_i(GFN1)	4	7	8	10	12	13	14	17	20	22	23	25
ρ_i(SMS4)	4	5	9	10	14	15	19	20	24	25	29	30

The time ρ_7(GFN1) $= 14$ given in the table refines Proposition 5 of [4]. □

Acknowledgments: The author thanks Egor Lawrenov for fruitful discussions and computer experiments, the results of which are used in the above examples. The author also thanks the anonymous referees of CTCRYPT 2020 for their valuable feedback.

References

[1] Agievich S. XS-circuits in block ciphers. Mat. Vopr. Kriptogr., 10, 33–41 (2019).

[2] Albrecht M., Grassi L., Rechberger C., Roy A., Tiessen T. MiMC: efficient encryption and cryptographic hashing with minimal multiplicative complexity. In: Cheon J., Takagi T. (eds) Advances in Cryptology — ASIACRYPT 2016. Lecture Notes in Computer Science, vol 10031. Springer, Berlin, Heidelberg, 191–219 (2016).

[3] Avanzi R. A Salad of Block Ciphers. Cryptology ePrint Archive, Report 2016/1171 (2016). Avail. at: `https://eprint.iacr.org/2016/1171`.

[4] Bogdanov A., Shibutani K. Generalized Feistel Networks Revisited. Des. Codes Cryptogr., 66, 75–97 (2013).

[5] Daemen J. Cipher and Hash Function Design Strategies Based on Linear and Differential Cryptanalysis. K.U. Leuven, Doctoral Dissertation (1995). Avail. at: `https://cs.ru.nl/~joan/papers/JDA_Thesis_1995.pdf`.

[6] Daemen J., Rijmen V. The Wide Trail Design Strategy. In: Honary B. (eds) Cryptography and Coding 2001. Lecture Notes in Computer Science, vol 2260. Springer, Berlin, Heidelberg, 222–238 (2001).

[7] Diffie W., Ledin G. SMS4 Encryption Algorithm for Wireless Networks. Cryptology ePrint Archive, Report 2008/329 (2008). Avail. at: `https://eprint.iacr.org/2008/329`.

[8] Huffman W.C., Pless V. Fundamentals of Error-Correcting Codes. Cambridge Univ. Press, Cambridge (2003).

[9] Liddl R., Niederreiter H. Finite Fields. Cambridge University Press, New York, NY, USA (1997).

[10] MacWilliams F.J., Sloane N.J.A. The Theory of Error-Correcting Codes. Elsevier Science (1977).

[11] Wu S., Wang M. Security Evaluation against Differential Cryptanalysis for Block Cipher Structures. Cryptology ePrint Archive, Report 2011/551 (2011). Avail. at: `https://eprint.iacr.org/2011/551`.

[12] Zhang J., Wu W., Zheng Y. Security of SM4 Against (Related-Key) Differential Cryptanalysis. In: Bao F., Chen L., Deng R., Wang G. (eds) Information Security Practice and Experience. ISPEC 2016. Lecture Notes in Computer Science, vol 10060. Springer, Cham, 65–78 (2016).

[13] Zhang M., Liu J., Wang X. The Upper Bounds on Differential Characteristics in Block Cipher SMS4. Cryptology ePrint Archive, Report 2010/155 (2010). Avail. at: `https://eprint.iacr.org/2010/155`.

[14] Zheng Y., Matsumoto T., Imai H. On the Construction of Block Ciphers Provably Secure and Not Relying on Any Unproved Hypotheses. In: Brassard G. (eds) Advances in Cryptology – CRYPTO'89 Proceedings. Lecture Notes in Computer Science, vol 435. Springer, New York, NY, 461–480 (1990).

Construction of orthomorphic MDS matrices with primitive characteristic polynomial

Oliver Coy Puente and Reynier Antonio de la Cruz Jiménez

Institute of Cryptography, Havana University, Cuba
o.coypuente@gmail.com, djr.antonio537@gmail.com

Abstract

Matrices having the Maximum Distance Separable property (MDS matrices) are a vital component for the design of symmetric-key algorithms to achieve the diffusion property. In recent years, there has been a lot of work on the construction and characterization of MDS matrices with a low implementation cost in the context of the so-called lightweight schemes. However, many authors do not pay attention to the influence of reducibility of the proposed MDS matrices and as a consequence an adversary can exploit the nontrivial invariant subspaces associated to these mappings. In this article, we propose some methods for constructing linear matrices of size 4×4 and 6×6 with primitive characteristic polynomial that preserves the MDS property having better resistance against the so-called invariant subspaces attacks.

Keywords: MDS-matrix, recursive matrix, companion matrix, Feistel Network, invariant subspace, linear orthomorphism.

1 Introduction

Modern block ciphers are often iterations of several rounds. Each round (which must depend on the key) consists of a confusion layer and a diffusion layer. The confusion layers are usually formed by local nonlinear mappings (S-Boxes) while the diffusion layers are formed by global linear mappings mixing the output of the different S-Boxes. The security of a diffusion layer is measured by its differential branch number and the linear branch number. The larger the two branch numbers are, the stronger a diffusion layer is. The diffusion layers with the optimal branch numbers are called being maximum distance separable (MDS). But a branch number describes diffusion property of an MDS matrix with respect to the canonical basis only. There are several linear mappings of block ciphers with reducible characteristic polynomial (see, for example, [6, 9]) and such mappings have nontrivial invariant subspaces, moreover, the group generated by linear mapping and the additive group of vector space is an imprimitive group. In this situation, if a nonlinear bijective transformation does not diffuse systems of imprimitivity of this

group, then block cipher may be vulnerable, for example, to homomorphism attack [7].

From practical point of view, is not only desirable that an MDS matrix can be implemented efficiently both in software/hardware but also when encryption and decryption implementations are required and the inverse of the MDS matrix will have to be implemented as well except for Feistel and Lai-Massey structures, where the inverse of the internal function is not required for decryption. However, constructing an MDS matrix with low implementation cost (as to suit lightweight schemes) is a nontrivial task.

Although, several articles published by Burov D. A., Pogorelov B. A. and Pudovkina M. A. are devoted to the question on the influence of the linear mapping reducibility on the security of block ciphers [7, 8, 13, 14, 15], in this work we continued the research topic started in [16], considering the problem of building an special kind of linear mappings with primitive characteristic polynomial and acceptable implementation cost having the following advantages, such as no non-zero fixed points and better resistance against the so-called invariant subspaces attacks.

This article is structured as follows: We begin with preliminaries in Section 2, proving some useful results and then we give some basic concepts about MDS matrices and XOR-count metric. Some methods for constructing orthomorphic MDS matrices with primitive characteristic polynomial are given in Section 3,4 and 5 respectively, for all these matrices its implementation cost is analysed in Section 6. Our work is concluded in Section 7.

2 Preliminaries

Let be $k \geqslant 2$,Q =GF(q)=GF(p)[x]/$q(x)$ the finite field of $q = p^n$ elements, where $q(x)$ is a es irreducible polynomial of degree n over GF(p) and p is a prime number, by Q^* we denote multiplicative group of Q, i.e., $Q^* = Q \setminus \{0\}$. The set of all vectors of dimension k over Q is denoted by Q^k. The set of all matrices of dimension $k \times k$ over Q is denoted by $Q_{k,k}$ and by $Q^*_{k,k}$ we denote the set of all invertible matrices of size $k \times k$ over Q. Throughout the article, we shall use the following operations and notations:

0	- the null element of Q;
e	- the neutral element of the multiplicative group Q^*;
$w_H(\vec{a})$	- the Hamming weight of a vector $\vec{a} \in Q^k$, i.e. the number of its nonzero coordinates;
$I_{k\times k}$	- the identity matrix of $Q_{k,k}$;

$O_{k\times k}$ - the zero matrix of $Q_{k,k}$;
$(\vec{v})^\top$ - the transpose of a vector $\vec{v} \in Q^k$;
$\operatorname{rank} A$ - the rank of the matrix $A \in Q_{k,k}$;
$|A|$ - the determinant of the matrix of $A \in Q_{k,k}$;
$a(x) \mid b(x)$ - the polynomial $a(x)$ divides the polynomial $b(x)$;
$a(x) \nmid b(x)$ - the polynomial $a(x)$ is not a divisor of the polynomial $b(x)$.

2.1 Linear algebra

Definition 1. *[1] Let be* $f(x) = f_0 - f_1x - f_2x^2 - \ldots - f_{k-2}x^{k-2} - f_{k-1}x^{k-1} - x^k \in Q[x]$, *the matrix* $S_f \in Q_{k,k}$, *defined as*

$$S_f = \begin{pmatrix} 0 & 0 & \cdots & 0 & 0 & f_0 \\ e & 0 & \cdots & 0 & 0 & f_1 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & e & 0 & f_{k-2} \\ 0 & 0 & \cdots & 0 & e & f_{k-1} \end{pmatrix}_{k\times k}$$

is called companion matrix of the polynomial $f(x)$.

For companion matrices of irreducible polynomials it is known that $GF(q^k)$ is isomorphic to $Q(S_f)$, where $Q(S_f) = ((Q, S_f), +, \cdot)$ and $(Q, S_f) = \left\{ \sum_{i=0}^{k-1} a_i S_f^i : (a_0, \ldots, a_{k-1}) \in Q^k \right\}$ (see, for example, [2]).

As we can see in the next definition, the companion matrix is a particular case of the $\mathcal{P}$ - companion matrix − a new concept which is defined as follows.

Definition 2. *Let be* $f(x) = f_0 - f_1x - f_2x^2 - \ldots - f_{k-2}x^{k-2} - f_{k-1}x^{k-1} - x^k \in Q[x]$, *the matrix* $S_f(\mathcal{P}_{(k-1)\times(k-1)}) \in Q_{k,k}$, *defined as*

$$S_f(\mathcal{P}) = \left(\begin{array}{ccc|c} 0 & \cdots & 0 & f_0 \\ \hline & & & f_1 \\ & & & \vdots \\ & \mathcal{P}_{(k-1)\times(k-1)} & & \vdots \\ & & & f_{k-2} \\ & & & f_{k-1} \end{array}\right)_{k\times k}$$

is called $\mathcal{P}$ *- companion matrix of the polynomial* $f(x)$, *where* $\mathcal{P}_{(k-1)\times(k-1)}$ *is a permutation matrix.*

In [1], Definition 3, Chapter 11 is defined the order of an element. Analogously, we will define the multiplicative order of matrix of the multiplicative group $Q(S_f)^*$.

Definition 3. *The smallest integer* $r \in \{1, \ldots, q^k - 1\}$ *such that* $A^r = I_{k\times k}$ *is called multiplicative order of the matrix* $A \in Q_{k,k}$ *in the multiplicative group* $Q(S_f)^*$ *and denoted by* $\mathsf{Ord}(A)$.

Definition 4. *[1] The characteristic polynomial of the matrix* $A \in Q_{k,k}$, *denoted by* $\chi_A(x)$, *is defined as follows*

$$\chi_A(x) = |x \cdot I_{k\times k} - A|.$$

Definition 5. *[3] Let* $g(x)$ *be a polynomial over* Q *of degree* $\deg(g(x)) \geq 1$ *and non-zero constant term. The period of* $g(x)$ *, denoted by* $T(g)$, *is defined to be the least positive integer* l *such that* $g(x) \mid (x^l - 1)$.

Definition 6. *[1] The polynomial* $m_A(x) \in Q[x]$ *is called the minimal polynomial of the matrix* $A \in Q_{k,k}$ *if and only if* $m_A(A) = O_{k\times k}$, *and for any* $g(x) \in Q[x]$ *such that* $g(A) = O_{k\times k}$, $\deg m_A(x) \leq \deg(g)$.

It is well known (see [1], Theorem 26, Chapter 25) that $\chi_{S_f}(S_f) = O_{k\times k}$ and $\chi_{S_f}(x) = f(x)$. This facts can be used to determine the reducibility of the polynomial $\chi_{S_f^t}(x)$ for some $t \in \mathbb{N}$.

Proposition 1. *Let be* $Q = GF(q)$ *and* $f(x) \in Q[x]$ *an irreducible polynomial of degree* k *over* Q. *Then for any integer* $i \in \{1, \ldots, k-1\}$ *the following equality holds*

$$\chi_{S_f^{q^i}}(x) = f(x).$$

Proof. We have that $f(x)$ is an irreducible polynomial over Q, then $S_f, S_f^q, \ldots, S_f^{q^{k-1}}$ are its roots in the field $Q(S_f)$ and for any integer $i \in \{1, \ldots, k-1\}$ we have that $\chi_{S_f^{q^i}}(x) | f(x)$. Now, using the irreducibility of the polynomial $f(x)$ we conclude that $\chi_{S_f^{q^i}}(x) = f(x)$. □

Proposition 2. *Let be* $Q = GF(q)$ *and* $f(x) \in Q[x]$ *a primitive polynomial of degree* k *over* Q. *Then for any integer* $t \in \{1, \ldots, q^k - 1\}$ *such that* $(t, q^k - 1) = 1$, *the polynomial* $\chi_{S_f^t}(x)$ *is primitive over* Q.

Proof. We have that $f(x)$ is a primitive polynomial over Q, then S_f and S_f^t are primitive elements over $Q(S_f)$. We need to prove that $\chi_{S_f^t}(x)$ is primitive over Q. If $\chi_{S_f^t}(x)$ is reducible, then there exist an irreducible divisor $g(x)$ of the polynomial $\chi_{S_f^t}(x)$, such that $g(S_f^t) = O_{k\times k}$ and $T(g) \mid T(\chi_{S_f^t})$ and this is a contradiction because $T(g) = q^k - 1$ and $T(\chi_{S_f^t}) < q^k - 1$. Then $\chi_{S_f^t}(x)$ is irreducible and because S_f^t is a primitive element of $Q(S_f)$ we conclude that the polynomial $\chi_{S_f^t}(x)$ is primitive (see [1], Theorem 29, Chapter 25). □

Definition 7. *Let be $f(x) \neq x^l \in Q[x], l \in \mathbb{N}$ an arbitrary polynomial and $\{\alpha_1, \ldots, \alpha_s\}$ the set of all nonzero roots of the polynomial $f(x)$ over Q. We will denote by $O(f)$ the least common multiple of multiplicative orders of the elements α_i:*

$$O(f) = lcm(\mathsf{ord}\alpha_1, \ldots, \mathsf{ord}\alpha_s).$$

It is known that if $f(x)$ is an irreducible polynomial of degree k over Q then for $i \in \overline{1,k}$ $O(f) = \mathsf{ord}\alpha_i$, where α_i belongs to some factorization field of $f(x)$ (see [1]).

Proposition 3. *Let be $f(x) \in Q[x]$ a primitive polynomial over Q, then for any integer $t < q^{\frac{k}{2}}$:*

1. *$m_{S_f^t}(x)$ is an irreducible polynomial over Q;*
2. *$deg(m_{S_f^t}(x)) = k$.*

Proof. We have that $f(x) \in Q[x]$ is a primitive polynomial over Q, then $Q(S_f)^* = \langle S_f \rangle$ and $\mathsf{Ord}(S_f) = q^k - 1$. The first statement is a direct consequence of the lemma 16 given in ([4], p. 41) and to prove the second statement we observe that

$$m = deg(m_{S_f^t}(x)) = \min\left\{s : \mathsf{Ord}(S_f^t) | (q^s - 1)\right\} =$$

$$= \min\left\{s : \frac{O(f)}{gcd(O(f), t)} | (q^s - 1)\right\} = \min\left\{s : \frac{q^k - 1}{gcd(q^k - 1, q^s - 1)} \Big| t\right\} =$$

$$= \min\left\{s : \frac{q^k - 1}{q^{gcd(k,s)} - 1} \Big| t\right\}.$$

If $m < k$, then $(k, m) \leqslant \frac{k}{2}$, which is equivalent to

$$\frac{q^k - 1}{q^{gcd(k,m)} - 1} \geqslant \frac{q^k - 1}{q^{\frac{k}{2}} - 1} > q^{\frac{k}{2}} \text{ and } \frac{q^k - 1}{q^{gcd(k,m)} - 1} \nmid t.$$

□

Notice that the results of propositions 1, 2 and 3 allow us to describe some integers t, for which the linear mapping $\mathcal{L} : Q^k \to Q^k$, defined as $\mathcal{L}(\vec{a}) = \vec{a} \cdot S_f^t$, where $\vec{a} \in Q^k$, does not have invariant subspaces when $f(x) \in Q[x]$ is a primitive polynomial.

In what follows, we shall consider the field Q of even characteristic, i.e., $p = 2$. In this case we denote by 1 the neutral element of the multiplicative group Q^*, and by $\oplus$ the addition in Q.

Definition 8. *The linear mapping $\mathcal{L} : Q^k \to Q^k$, defined as $\mathcal{L}(\vec{a}) = \vec{a} \cdot A$, where $\vec{a} \in Q^k, A \in Q^*_{k,k}$ is called a linear orthomorphism if the matrix $A \oplus I_{k\times k}$ is invertible over Q.*

Proposition 4. *For all $\vec{a} \in Q^k$ and any invertible matrix $A \in Q_{k,k}$ the linear transformation $\mathcal{L} : Q^k \to Q^k$, defined as $\mathcal{L}(\vec{a}) = \vec{a} \cdot A$ is a linear orthomorphism if and only if $\mathcal{L}$ has no non-zero fixed points.*

Proof. It is well known from linear algebra that the number of non-zero fixed points of $\mathcal{L}$ is $2^{n(k-\text{rank}(A\oplus I_{k,k}))} - 1$. From here we obtain that the linear transformation $\mathcal{L}$ has no non-zero fixed points if and only if when $\text{rank}(A \oplus I_{k,k}) = k$ which means that $\mathcal{L}$ is a linear orthomorphism. □

Proposition 5. *Let be A an invertible matrix of size $k \times k$ over Q. Then for any $\alpha \in Q^*$ the matrix $\alpha \cdot I_{k\times k} \oplus A$ is invertible over Q if and only if $(x \oplus \alpha) \nmid \chi_A(x)$.*

Proof. It is easy to see that the following holds $(x \oplus \alpha)|\chi_A(x)$ if and only if α is a root of $\chi_A(x)$, which is equivalent to $0 = \chi_A(\alpha) = |\alpha \cdot I_{k\times k} \oplus A|$. □

Corollary 1. *If the polynomial $\chi_{S_f}(x) \in Q[x]$ is primitive over Q, then for any $\alpha \in Q^*$ the matrix $\alpha \cdot I_{k\times k} \oplus S^t_f \in Q_{k,k}$ is invertible for any integer t such that the following conditions holds:*

1. *$t < 2^{\frac{k}{2}}$;*
2. *$t \in \{1, \ldots, 2^k - 1\}$ with $(t, 2^k - 1) = 1$.*

Proof. The proof follows from the propositions 2, 3 and 5. □

2.2 MDS matrices

Definition 9. *[5] The branch number ρ of matrix $A \in Q_{k,k}$ is defined as*

$$\rho(A) = \min_{\vec{a}\neq\vec{0}}\{w_H(\vec{a}) + w_H(\vec{a}A)\}.$$

Definition 10. *[5, 10] The matrix $A \in Q_{k,k}$ is called MDS if $\rho(A) = k + 1$*

.

Lemma 1. *[10] The matrix $A \in Q_{k,k}$ is an MDS matrix if and only if all its quadratic submatrices are invertible over Q.*

Lemma 2. *[10] The matrix $A \in Q_{4,4}$ with all elements nonzero is an MDS matrix if and only if it is an invertible matrix with the inverse matrix having all elements nonzero and all its submatrices of dimension 2×2 are invertible over Q.*

Definition 11. *A matrix $A \in Q_{k,k}$ is said to be an orthomorphic MDS matrix if both matrices A and $A \oplus I_{k\times k}$ have the MDS property over Q.*

2.3 XOR-count

In [12] the authors proposed quantify the complexity of implementing the linear layer of a block cipher by counting the number of bitwise XOR operations, necessary to implement the multiplication of any vector by the linear mappings used as building block to achieve the diffusion property. In this articles we shall use this metric to assess the cost of the proposed constructions.

Definition 12. *[12] The XOR-count of an element $\alpha \in Q$ is the number of XOR operations required to implement the multiplication of α by arbitrary element $\beta \in Q$.*

Throughout the article we shall denote the XOR-count of an element $\alpha \in Q$ by $\mathsf{XOR}(\alpha)$. It's not difficult to check that $\mathsf{XOR}(0) = \mathsf{XOR}(1) = 0$. The XOR-count of the column j of the matrix $M = (m_{i,j})_{k\times k}$ is calculated by the following formula given in [12]:

$$\sum_{i=1}^{k} \mathsf{XOR}(m_{i,j}) + (l_j - 1) \cdot n,$$

where l_j is the number of non-zero elements of the j–th column. Then the XOR-count of matrix $M = (m_{i,j}) \in Q_{k,k}$ we can calculate by the formula

$$\mathsf{XOR}(M) = \sum_{j=1}^{k} \sum_{i=1}^{k} \mathsf{XOR}(m_{i,j}) + n \cdot \sum_{j=1}^{k} (l_j - 1). \tag{1}$$

Let be $f(x) = a_0 \oplus a_1 x \oplus a_2 x^2 \oplus ... \oplus a_{k-1} x^{k-1} \oplus x^k$ and $\{c_1, \ldots, c_s\}$ the set of all non-zero different coefficients of polynomial $f(x)$. In [11] is showed that

$$\mathsf{XOR}(S_f) = \sum_{j=1}^{s} \mathsf{XOR}(c_j) + (l_f - 1) \cdot n,\ \mathsf{XOR}(S_f^t) = t \cdot \mathsf{XOR}(S_f), \tag{2}$$

for all $t > 0$, where l_f is the number of all non-zero different coefficients of the last column of the matrix S_f.

Is easy to see that

$$\mathsf{XOR}(S_f) = \mathsf{XOR}(S_f(\mathcal{P}_{(k-1)\times(k-1)})), \tag{3}$$

for any permutations matrix $\mathcal{P}_{(k-1)\times(k-1)}$.

Choosing the irreducible polynomial $x^8 \oplus x^7 \oplus x^6 \oplus x \oplus 1$ over $GF(2)$ we create the field $Q = GF(2^8) = GF(2)[x]/x^8 \oplus x^7 \oplus x^6 \oplus x \oplus 1$ and throughout

the article we shall work only in this field. The values compiled in table 1 correspond to the XOR-count of finite field elements (written in hexadecimal form). From this table the XOR-count can be calculated as follows, for an element $\beta = \mathsf{0x20} \in GF(2^8)$ we obtaint that $\mathsf{XOR}(\beta) = \mathsf{16}$.

XOR	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.a	.b	.c	.d	.e	.f
0.	**0**	0	3	9	5	11	10	14	7	11	12	18	14	20	13	21
1.	**12**	18	11	19	13	17	18	24	17	23	22	26	12	20	23	29
2.	**16**	**22**	**21**	**25**	**11**	**19**	**22**	**28**	**17**	**23**	**16**	**24**	**18**	**22**	**23**	**29**
3.	**20**	24	25	31	27	33	26	34	11	19	22	28	24	30	29	33
4.	**20**	24	23	29	25	31	26	34	11	19	20	26	22	28	29	33
5.	**18**	24	25	29	15	23	24	30	19	25	20	28	22	26	25	31
6.	**24**	30	25	33	27	31	30	36	29	35	36	40	26	34	35	41
7.	**10**	18	19	25	21	27	28	32	25	29	28	34	30	36	31	39
8.	**25**	21	26	20	24	22	31	27	30	26	33	31	27	21	36	32
9.	**11**	5	20	16	22	18	25	23	22	20	29	25	31	27	32	26
a.	**19**	17	26	22	28	24	29	23	14	8	23	19	25	21	28	26
b.	**21**	17	24	22	18	12	27	23	22	18	23	17	21	19	28	24
c.	**27**	23	32	30	26	20	33	29	28	24	31	25	29	27	34	30
d.	**31**	29	36	32	38	34	41	35	26	20	33	29	35	31	40	38
e.	**9**	3	16	12	18	14	23	21	20	18	25	21	27	23	30	24
f.	**25**	21	28	22	26	24	31	27	30	26	35	33	29	23	36	32

Table 1: XOR-count of elements of the field Q

3 Constructing orthomorphic MDS matrices from companion matrices

In this section we study the possibility of constructing orthomorphic MDS matrices from companion matrices.

It is noteworthy that if the polynomial $f(x) = f_0 \oplus f_1 x \oplus f_2 x^2 \oplus ... \oplus f_{k-2}x^{k-2} \oplus f_{k-1}x^{k-1} \oplus x^k \in Q[x]$ has the coefficient $f_0 = 1$ then

$$S_f^k = \begin{pmatrix} 1 & * & \cdots & * & * \\ f_1 & * & \cdots & * & * \\ f_2 & * & \cdots & * & * \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ f_{k-2} & * & \cdots & * & * \\ f_{k-1} & * & \cdots & * & * \end{pmatrix}_{k\times k},$$

and this mean that

$$I_{k\times k} \oplus S_f^k = \begin{pmatrix} 0 & * & \cdots & * & * \\ f_1 & * & \cdots & * & * \\ f_2 & * & \cdots & * & * \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ f_{k-2} & * & \cdots & * & * \\ f_{k-1} & * & \cdots & * & * \end{pmatrix}_{k\times k}.$$

Hence, $f_0 \neq 1$ is a necessary condition for the matrix S_f^k to be an orthomorphic MDS matrix.

Now we propose a method for constructing 4×4 orthomorphic MDS matrices with primitive characteristic polynomial from the companion matrix of the polynomial

$$f(x) = x^4 \oplus \gamma^{-1}x^3 \oplus \gamma^t x^2 \oplus \gamma^{-1}x \oplus \gamma \in Q[x]. \tag{4}$$

Proposition 6. *Let be $Q = GF(2^8)$, $\gamma \in Q$ a primitive element, $f(x) \in Q[x]$ a polynomial of the form (4). If $t \in \{2, \ldots, 254\}$ is a prime number and $f(x)$ is an irreducible polynomial over Q, then the matrix S_f^4 is an orthomorphic MDS matrix.*

Proof. By definition 11, it is necessary to show that the matrices S_f^4 and $I_{4\times4} \oplus S_f^4$ are MDS matrices. Let's show that $L = S_f^4$ is an MDS matrix. From lemmas 1 and 2 that is sufficient to check that all elements of the matrices $L = (l_{i,j})_{4\times4}$ and $L^{-1} = (l'_{i,j})_{4\times4}$ are nonzero, and that all minors of dimension 2×2 are nonzero too. The matrix L has the following form

$$L = \begin{pmatrix} \gamma & 1 & \gamma^{t+1} \oplus \gamma^{-1} & \gamma^{-2} \oplus 1 \\ \gamma^{-1} & \gamma \oplus \gamma^{-2} & \gamma^{t-1} \oplus \gamma^{-3} \oplus 1 & \gamma^{t+1} \oplus \gamma^{-4} \oplus \gamma^{-2} \oplus \gamma^{-1} \\ \gamma^t & \gamma^{t-1} \oplus \gamma^{-1} & \gamma^{2t} \oplus \gamma^{t-2} \oplus \gamma^{-2} \oplus \gamma & \gamma^{t-3} \oplus \gamma^{-3} \\ \gamma^{-1} & \gamma^t \oplus \gamma^{-2} & \gamma^{-3} \oplus \gamma^{-1} & \gamma^{2t} \oplus \gamma^{t-2} \oplus \gamma^{-4} \oplus \gamma \end{pmatrix},$$

the matrix L^{-1} is of the form

$$L^{-1} = \begin{pmatrix} \gamma^{-8} \oplus \gamma^{t-5} \oplus \gamma^{2t-2} \oplus \gamma^{-1} & \gamma^{-6} \oplus \gamma^{-2} & \gamma^{-4} \oplus \gamma^{t-1} & \gamma^{-2} \\ \gamma^{t-7} \oplus \gamma^{-6} \oplus \gamma^{-3} & \gamma^{t-5} \oplus \gamma^{-4} \oplus \gamma^{-1} \oplus \gamma^{2t-2} & \gamma^{-2} \oplus \gamma^{t-3} & \gamma^{t-1} \\ \gamma^{-8} \oplus \gamma^{-5} \oplus \gamma^{-4} \oplus \gamma^{t-2} & \gamma^{-6} \oplus \gamma^{-3} \oplus \gamma^{t-3} & \gamma^{-4} \oplus \gamma^{-1} & \gamma^{-2} \\ \gamma^{-7} \oplus \gamma^{-3} & \gamma^{-5} \oplus \gamma^{t-2} & \gamma^{-3} & \gamma^{-1} \end{pmatrix}.$$

We shall denote by $\Theta = \{\theta[i], i \in \overline{1, 36}\}$ the set of all minors of the matrix L of dimension 2×2. Using computer calculations we find that

$$\begin{aligned} \Theta \quad = \{ \quad & 1,\ \gamma,\ \gamma^2,\ \gamma^2 \oplus \gamma^{-1},\ \gamma^{t-2} \oplus \gamma^{-1} \oplus \gamma^{-2},\ \gamma^{t-1} \oplus 1,\ \gamma^{t-1} \oplus \gamma^{-1}, \\ & \gamma^{t-1} \oplus \gamma^{-1} \oplus 1,\ \gamma^t \oplus 1,\ \gamma^t \oplus \gamma,\ \gamma^t \oplus \gamma^{-3} \oplus 1,\ \gamma^t \oplus \gamma^{-2} \oplus \gamma, \\ & \gamma^{t+1},\ \gamma^{t+1} \oplus \gamma^{-2},\ \gamma^{t+2},\ \gamma^{t+2} \oplus 1, \\ & \gamma^{2t-2} \oplus \gamma^{t-2} \oplus \gamma^{t-1} \oplus \gamma^{-4} \oplus \gamma^{-1} \oplus \gamma^2,\ \gamma^{2t-1} \oplus \gamma^{t-1} \oplus \gamma^{-3} \oplus 1, \\ & \gamma^{2t-1} \oplus \gamma^t \oplus \gamma^{-3} \oplus 1,\ \gamma^{2t-1} \oplus \gamma^{t-3} \oplus \gamma^t \oplus \gamma^{-3} \oplus \gamma^{-2} \oplus 1, \\ & \gamma^{2t} \oplus \gamma^{-2},\ \gamma^{2t} \oplus \gamma^t \oplus \gamma^{-2} \oplus \gamma,\ \gamma^{2t} \oplus \gamma^{t+1} \oplus \gamma^{-2} \oplus \gamma, \\ & \gamma^{2t} \oplus \gamma^{t-3} \oplus \gamma^t \oplus \gamma^{-3} \oplus \gamma^{-2} \oplus \gamma,\ \gamma^{2t+1} \oplus \gamma^{-1}, \\ & \gamma^{2t+1} \oplus \gamma^{t-2} \oplus \gamma^2 \oplus \gamma^{-1},\ \gamma^{2t+1} \oplus \gamma^{t-1} \oplus \gamma^2 \oplus \gamma^{-1}, \\ & \gamma^{2t+1} \oplus \gamma^{t-1} \oplus \gamma^{t-2} \oplus \gamma^{-4} \oplus \gamma^{-1},\ \gamma^{2t+2} \oplus 1, \\ & \gamma^{3t-1} \oplus \gamma^{2t-1} \oplus \gamma^{-2} \oplus 1,\ \gamma^{3t-1} \oplus \gamma^{2t} \oplus \gamma^{-3} \oplus \gamma \\ & \gamma^{3t} \oplus \gamma^{t+1} \oplus \gamma^{-2} \oplus \gamma^{-1},\ \gamma^{3t} \oplus \gamma^{2t-2} \oplus \gamma^{t+1} \oplus \gamma^{-4} \oplus \gamma^{-1}, \\ & \gamma^{3t+1} \oplus \gamma^{t+2} \oplus \gamma^{-1} \oplus 1,\ \gamma^{3t+1} \oplus \gamma^{2t-2} \oplus \gamma^{t+2} \oplus \gamma^{-4} \oplus \gamma^{-1}, \\ & \gamma^{4t} \oplus \gamma^{2t-2} \oplus \gamma^{-4} \oplus \gamma^2\}. \end{aligned}$$

By direct computer calculations we have checked that if $t \in \{2, \ldots, 254\}$ is a prime number then all these elements are nonzero. Similarly, we obtain that the matrix $I_{4\times4} \oplus S_f^4$ is MDS too. □

Analogously, to the previous case we propose a construction of orthomorphic MDS matrices of size 6×6 with primitive characteristic polynomial from the companion matrix of the polynomial

$$g(x) = x^6 \oplus \gamma_1 x^5 \oplus \gamma_2 x^4 \oplus \gamma_3 x^3 \oplus \gamma_2 x^2 \oplus \gamma_1 x \oplus \gamma \in Q[x]. \tag{5}$$

Proposition 7. *Let be $\gamma \in Q = GF(2^8)$ a primitive element and $g(x) \in Q[x]$ a polynomial of the form (5). If $(\gamma_1, \gamma_2, \gamma_3) \in \{(\gamma^{-3}, \gamma^{-4}, \gamma^4), ((\gamma \oplus 1)^{-1}, \gamma^3, \gamma^2), ((\gamma \oplus 1)^{-1}, \gamma^{-1} \oplus 1, \gamma), (\gamma^{-2} \oplus 1, \gamma^{-1}, \gamma^{-1} \oplus 1)\}$, then the matrix S_g^6 is an orthomorphic MDS matrix.*

Proof. Similarly, to the proof of the proposition 6. □

When both characteristic polynomials $f(x)$ and $g(x)$ are primitive, then from propositions 2, 3 we have that for any $\vec{a} \in Q^4$ and $\vec{b} \in Q^6$ the following linear mappings $F^4(\vec{a}) = \vec{a} \cdot S_f^4$, $G^6(\vec{b}) = \vec{b} \cdot S_g^6$ does not have invariant subspaces (see, for example,[1]).

Let be $Q = GF(2^8)$. In tables 3 and 4 we compile some matrices of the form S_f^4 and S_g^6 with primitive polynomials f and g from (4) and (5) respectively.

(γ, t)	S_f	S_f^4
(02, 7)	$\begin{pmatrix} 00 & 00 & 00 & 02 \\ 01 & 00 & 00 & E1 \\ 00 & 01 & 00 & 80 \\ 00 & 00 & 01 & E1 \end{pmatrix}$	$\begin{pmatrix} 02 & 01 & 22 & 90 \\ E1 & 93 & E8 & 06 \\ 80 & A1 & C4 & B8 \\ E1 & 11 & 48 & E0 \end{pmatrix}$
(10, 101)	$\begin{pmatrix} 00 & 00 & 00 & 10 \\ 01 & 00 & 00 & B5 \\ 00 & 01 & 00 & 71 \\ 00 & 00 & 01 & B5 \end{pmatrix}$	$\begin{pmatrix} 10 & 01 & A9 & 2E \\ B5 & 3F & DD & 59 \\ 71 & 07 & B8 & CC \\ B5 & 5E & DB & 48 \end{pmatrix}$
(FD, 109)	$\begin{pmatrix} 00 & 00 & 00 & FD \\ 01 & 00 & 00 & 9F \\ 00 & 01 & 00 & 32 \\ 00 & 00 & 01 & 9F \end{pmatrix}$	$\begin{pmatrix} FD & 01 & BB & E0 \\ 9F & 1C & BD & CB \\ 32 & 8D & 48 & A6 \\ 9F & D3 & 31 & 38 \end{pmatrix}$

Table 3: Some matrices of the form S_f^4 and it's building blocks.

$(\gamma, \gamma_1, \gamma_2, \gamma_3)$	S_g	S_g^6
(02, 90, E1, E0)	$\begin{pmatrix} 00 & 00 & 00 & 00 & 00 & 02 \\ 01 & 00 & 00 & 00 & 00 & 90 \\ 00 & 01 & 00 & 00 & 00 & E1 \\ 00 & 00 & 01 & 00 & 00 & E0 \\ 00 & 00 & 00 & 01 & 00 & E1 \\ 00 & 00 & 00 & 00 & 01 & 90 \end{pmatrix}$	$\begin{pmatrix} 02 & E3 & AA & F2 & 08 & FF \\ 90 & B6 & 32 & 5C & F7 & 50 \\ E1 & D8 & 7D & EF & 5E & B8 \\ E0 & 39 & 46 & D9 & E9 & 8F \\ E1 & A8 & F2 & 9B & DB & A6 \\ 90 & 55 & 79 & 04 & 9E & 83 \end{pmatrix}$
(04, B4, 91, 90)	$\begin{pmatrix} 00 & 00 & 00 & 00 & 00 & 04 \\ 01 & 00 & 00 & 00 & 00 & B4 \\ 00 & 01 & 00 & 00 & 00 & 91 \\ 00 & 00 & 01 & 00 & 00 & 90 \\ 00 & 00 & 00 & 01 & 00 & 91 \\ 00 & 00 & 00 & 00 & 01 & B4 \end{pmatrix}$	$\begin{pmatrix} 04 & 95 & B9 & 57 & 40 & 06 \\ B4 & 2A & 4D & FE & 46 & AE \\ 91 & 99 & 75 & C5 & FA & 7E \\ 90 & 08 & 79 & 98 & D1 & 22 \\ 91 & BD & 57 & F1 & 9C & E9 \\ B4 & BF & 65 & 10 & E0 & 72 \end{pmatrix}$
(A0, 9E, 27, 26)	$\begin{pmatrix} 00 & 00 & 00 & 00 & 00 & A0 \\ 01 & 00 & 00 & 00 & 00 & 9E \\ 00 & 01 & 00 & 00 & 00 & 27 \\ 00 & 00 & 01 & 00 & 00 & 26 \\ 00 & 00 & 00 & 01 & 00 & 27 \\ 00 & 00 & 00 & 00 & 01 & 9E \end{pmatrix}$	$\begin{pmatrix} A0 & 87 & F1 & 84 & 4E & 36 \\ 9E & 40 & 21 & 88 & 78 & 3B \\ 27 & E9 & 0A & 34 & 28 & 53 \\ 26 & CE & 64 & E8 & 69 & 8E \\ 27 & 51 & 84 & 71 & 48 & 42 \\ 9E & C7 & F7 & FD & 8D & 3D \end{pmatrix}$

Table 4: Some matrices of the form S_g^6 and it's building blocks.

4 Constructing orthomorphic MDS matrices from $\mathcal{P}$ - companion matrices

Let be $h(x) = x^k \oplus \bigoplus_{i=0}^{k-1} h_i x^i$, $h^{\downarrow} = (h_1, \ldots, h_{k-1})^{\top}$, $f^{\downarrow} = (f_1, \ldots, f_{k-1})^{\top} = \mathcal{P}_{(k-1)\times(k-1)} \cdot h^{\downarrow}$, where $\mathcal{P}_{(k-1)\times(k-1)} = (p_{ij})_{(k-1)\times(k-1)}$ is a permutation matrix. In this section we study the possibility of constructing orthomorphic MDS matrices from $\mathcal{P}$ - companion matrices of the polynomial $h(x)$. Permutations matrices for which $\chi_{S_h(\mathcal{P}_{(k-1)\times(k-1)})}(x) = f(x)$ are of particular interest, where $f(x) = x^k \oplus \bigoplus_{i=0}^{k-1} f_i x^i$, and $f_0 = h_0$. If $f(x)$ is an irreducible polynomial over Q, then from proposition 3 follows that for any integer $t < q^{\frac{k}{2}}$ the matrix $(S_h(\mathcal{P}_{(k-1)\times(k-1)}))^t$ has an irreducible characteristic polynomial. In this case the coefficients of the polynomial $h(x)$ can be defined according to the following rule

$$h_i = \begin{cases} 1, & i = k \\ \bigoplus_{j=1}^{k-1} \hat{p}_{ij} f_j, & i \in \overline{1, k-1} \\ f_0, & i = 0 \end{cases},$$

where $\mathcal{P}^{-1} = (\hat{p}_{ij})_{(k-1)\times(k-1)}$.

For $k = 4$ we propose a method for constructing 4×4 orthomorphic MDS

matrices with primitive characteristic polynomial of the form (4) using the so-called $\mathcal{P}$ - companion matrices of the polynomial h, which is defined as

$$h(x) = x^4 \oplus \gamma^{-1}x^3 \oplus \gamma^{-1}x^2 \oplus \gamma^t x \oplus \gamma. \tag{6}$$

Proposition 8. *Let be $Q = GF(2^8)$, $\gamma \in Q$ a primitive element, $f(x) \in Q[x]$ a polynomial of the form (4) and*

$$\mathcal{P}_{3\times 3} = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}.$$

If $t \in \{2, \ldots, 254\}$ is a prime number and $f(x)$ is an irreducible polynomial over Q, then the matrix $(S_h(\mathcal{P}_{3\times 3}))^4$ is an orthomorphic MDS matrix, where $h(x) = x^4 \oplus \gamma^{-1}x^3 \oplus \gamma^{-1}x^2 \oplus \gamma^t x \oplus \gamma$.

Proof. Similarly, to the proof of the proposition 6. □

It is noteworthy that the use of $\mathcal{P}$ - companion matrices have some effect over simple companion matrices when looking for MDS matrices. For example, the matrix $(S_h(\mathcal{P}_{3\times 3}))^4$ have the MDS property over $Q = GF(2^8) = GF(2)[x]/x^8 \oplus x^4 \oplus x^3 \oplus x^2 \oplus 1$ for $t = 29$, while the matrix S_h^4 does not exhibits this useful property.

The high level of view of a round of transformations $\vec{a} \cdot S_h^4$ and $\vec{a} \cdot (S_h(\mathcal{P}_{3\times 3}))^4$ is given in Figures 1 and 2 respectively, where $\vec{a} \in Q^4$ and $h(x) \in Q[x]$, $\mathcal{P}_{3\times 3}$ are defined in proposition 8. These tranformations share the same XOR-count metric (see subsection 2.3).

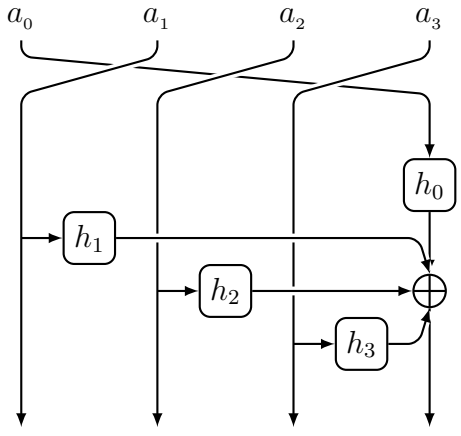


Fig. 1: High level of view of a round of transformation $\vec{a} \cdot S_h^4$

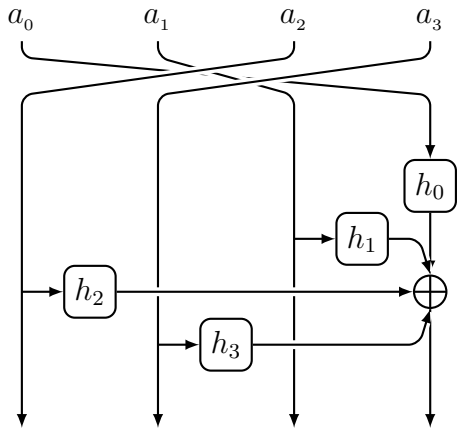


Fig. 2: High level of view of a round of transformation $\vec{a} \cdot (S_h(\mathcal{P}_{3\times 3}))^4$

For $k = 6$ we have constructed orthomorphic MDS over $Q = GF(2^8)$ matrices with primitive characteristic polynomial using the $\mathcal{P}$ - companion

matrix of the polynomial $h(x) = x^6 \oplus \bigoplus_{i=0}^{5} h_i x^i$, where for the primitive element γ the coefficients $h_i \in \{1, \gamma, \gamma^{-1}, \gamma^2, \gamma^{-2}, \gamma^3\},\ i \in \overline{0,5}$ due to the low XOR-count metric of these elements. In table 5 we compile some instances of the $\mathcal{P}$ - companion matrix defined by the polynomial $h(x)$.

$h(x) = x^6 \oplus \bigoplus_{i=0}^{5} h_i x^i$	$\mathcal{P}_{5\times5}$	$S_h(\mathcal{P}_{5\times5})$	$(S_h(\mathcal{P}_{5\times5}))^6$
$x^6 \oplus 04x^5 \oplus 04x^4 \oplus 91x^3 \oplus E1x^2 \oplus 91x \oplus 04$	$\begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 00 & 00 & 00 & 00 & 00 & 04 \\ 00 & 00 & 00 & 01 & 00 & 91 \\ 01 & 00 & 00 & 00 & 00 & E1 \\ 00 & 00 & 00 & 00 & 01 & 91 \\ 00 & 00 & 01 & 00 & 00 & 04 \\ 00 & 01 & 00 & 00 & 00 & 04 \end{pmatrix}$	$\begin{pmatrix} 04 & 1B & 10 & C2 & 41 & 7E \\ 91 & 1E & 90 & E5 & B4 & 51 \\ E1 & F9 & 06 & C8 & B1 & 64 \\ 91 & 8F & 05 & E2 & 40 & 0D \\ 04 & D3 & F1 & 73 & 47 & 87 \\ 04 & FE & 81 & 76 & D1 & 60 \end{pmatrix}$
$x^6 \oplus 04x^5 \oplus 01x^4 \oplus 08x^3 \oplus 04x^2 \oplus 91x \oplus 04$	$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 00 & 00 & 00 & 00 & 00 & 04 \\ 00 & 00 & 01 & 00 & 00 & 91 \\ 00 & 00 & 00 & 00 & 01 & 04 \\ 01 & 00 & 00 & 00 & 00 & 08 \\ 00 & 00 & 00 & 01 & 00 & 01 \\ 00 & 01 & 00 & 00 & 00 & 04 \end{pmatrix}$	$\begin{pmatrix} 04 & 0F & D3 & 10 & 41 & 8F \\ 91 & 1A & 5C & 05 & A0 & BD \\ 04 & D9 & 76 & 11 & 4D & D8 \\ 08 & CD & 24 & 24 & 92 & D2 \\ 01 & 57 & D6 & 0C & A5 & 9E \\ 04 & 53 & 73 & 81 & 44 & 95 \end{pmatrix}$
$x^6 \oplus E1x^5 \oplus E1x^4 \oplus 08x^3 \oplus 91x^2 \oplus 08x \oplus 04$	$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 00 & 00 & 00 & 00 & 00 & 04 \\ 00 & 00 & 01 & 00 & 00 & 08 \\ 00 & 00 & 00 & 01 & 00 & 91 \\ 00 & 00 & 00 & 00 & 01 & 08 \\ 01 & 00 & 00 & 00 & 00 & E1 \\ 00 & 01 & 00 & 00 & 00 & E1 \end{pmatrix}$	$\begin{pmatrix} 04 & 7A & E0 & 21 & 02 & BA \\ 08 & 2D & 51 & E3 & 95 & 54 \\ 91 & E3 & D9 & 52 & A1 & 8A \\ 08 & C9 & AC & D7 & E5 & C9 \\ E1 & 7E & 3D & AF & 95 & FC \\ E1 & CF & FF & 38 & 99 & AB \end{pmatrix}$

Table 5: Some matrices of the form $(S_h(\mathcal{P}_{5\times5}))^6$ and it's building blocks.

Proposition 9. *Let be $Q = GF(2^8)$. The matrices of the form $(S_h(\mathcal{P}_{5\times5}))^6$ from table 5 are orthomorphic MDS matrices.*

Proof. Similarly, to the proof of the proposition 6. □

5 Constructing orthomorphic MDS matrices of the form $(\gamma_1 \cdot \mathcal{R}_{k,k}^{(1)} \oplus \cdots \oplus \gamma_{\lceil \frac{k}{2} \rceil} \cdot \mathcal{R}_{k,k}^{(\lceil \frac{k}{2} \rceil)} \oplus \mathcal{R}_{k,k}^{(\lceil \frac{k}{2} \rceil+1)})^k$

In this section we study how to construct orthomorphic MDS matrices of the form

$$\left(\mathcal{M}\begin{pmatrix} \gamma_1 & \cdots & \gamma_{\lceil \frac{k}{2} \rceil} \\ \mathcal{R}_{k,k}^{(1)} & \cdots & \mathcal{R}_{k,k}^{(\lceil \frac{k}{2} \rceil)} \ \mathcal{R}_{k,k}^{(\lceil \frac{k}{2} \rceil+1)} \end{pmatrix}\right)^k, \tag{7}$$

with primitive characteristic polynomial, where

$$\mathcal{M}\begin{pmatrix} \gamma_1 & \cdots & \gamma_{\lceil \frac{k}{2} \rceil} \\ \mathcal{R}_{k,k}^{(1)} & \cdots & \mathcal{R}_{k,k}^{(\lceil \frac{k}{2} \rceil)} \ \mathcal{R}_{k,k}^{(\lceil \frac{k}{2} \rceil+1)} \end{pmatrix} = \left(\gamma_1 \cdot \mathcal{R}_{k,k}^{(1)} \oplus \cdots \oplus \gamma_{\lceil \frac{k}{2} \rceil} \cdot \mathcal{R}_{k,k}^{(\lceil \frac{k}{2} \rceil)} \oplus \mathcal{R}_{k,k}^{(\lceil \frac{k}{2} \rceil+1)}\right),$$

all binay non-zero matrices $\mathcal{R}_{k,k}^{(1)}, \ldots \mathcal{R}_{k,k}^{(\lceil \frac{k}{2} \rceil)}, \mathcal{R}_{k,k}^{(\lceil \frac{k}{2} \rceil+1)} \in GF(2)_{k,k}$, and in order to achieve an efficiently implementation the finite fields elements $\gamma_1, \ldots, \gamma_{\lceil \frac{k}{2} \rceil}$ belong to the set $\{1, \alpha, \alpha^{-1}, \alpha^2, \alpha^{-2}, \alpha^3, \alpha^{-3}\}$ where α is a primitive element. The reason behind such modification relies on the fact that if the matrix

$\mathcal{M}\begin{pmatrix} \gamma_1 & \cdots & \gamma_{\lceil \frac{k}{2} \rceil} \\ \mathcal{R}_{k,k}^{(1)} & \cdots & \mathcal{R}_{k,k}^{(\lceil \frac{k}{2} \rceil)} \; \mathcal{R}_{k,k}^{(\lceil \frac{k}{2} \rceil+1)} \end{pmatrix}$ is sparse and contain more 1's than finite field elements $\gamma_1, \ldots, \gamma_{\lceil \frac{k}{2} \rceil}$ then for small values of k if the resulting matrix of the form (7) have the MDS property we can achieve a low implementation cost.

For $k = 4, 6$ we have performed a search based on random generation of matrices $\mathcal{R}_{k,k}^{(1)}, \ldots \mathcal{R}_{k,k}^{(\lceil \frac{k}{2} \rceil)}, \mathcal{R}_{k,k}^{(\lceil \frac{k}{2} \rceil+1)}$ with few number of 1's and as a result have obtained the following matrices given in tables 6 and 7 having an efficient implementation.

(γ_1, γ_2)	$\left(\mathcal{M}\begin{pmatrix} \gamma_1 & \gamma_2 \\ \mathcal{R}_{4,4}^{(1)} & \mathcal{R}_{4,4}^{(2)} \; \mathcal{R}_{4,4}^{(3)} \end{pmatrix}\right)^4$	$\mathcal{R}_{4,4}^{(1)}$	$\mathcal{R}_{4,4}^{(2)}$	$\mathcal{R}_{4,4}^{(3)}$	$\mathcal{M}\begin{pmatrix} \gamma_1 & \gamma_2 \\ \mathcal{R}_{4,4}^{(1)} & \mathcal{R}_{4,4}^{(2)} \; \mathcal{R}_{4,4}^{(3)} \end{pmatrix}$
(01, E1)	$\begin{pmatrix} 70 & 90 & E1 & 01 \\ A9 & E1 & 91 & 48 \\ 90 & E1 & E1 & E1 \\ A9 & 91 & 90 & 70 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$	$\begin{pmatrix} 00 & 01 & 00 & 01 \\ 00 & 00 & 00 & E1 \\ 01 & 00 & 00 & 00 \\ E1 & 00 & 01 & 00 \end{pmatrix}$
(01, 91)	$\begin{pmatrix} 24 & B4 & 91 & 01 \\ BC & 91 & B5 & 2D \\ B4 & 91 & 91 & 91 \\ BC & B5 & B4 & 24 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$	$\begin{pmatrix} 00 & 01 & 00 & 01 \\ 00 & 00 & 00 & 91 \\ 01 & 00 & 00 & 00 \\ 91 & 00 & 01 & 00 \end{pmatrix}$
(A9, E1)	$\begin{pmatrix} 5D & B4 & A9 & 01 \\ 5E & E1 & B5 & 5A \\ B4 & A9 & E1 & A9 \\ 5E & BC & B4 & 5D \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$	$\begin{pmatrix} 00 & 01 & 00 & 01 \\ 00 & 00 & 00 & E1 \\ 01 & 00 & 00 & 00 \\ A9 & 00 & 01 & 00 \end{pmatrix}$

Table 6: Some matrices of the form (7) and it's building blocks for $k = 4$.

$(\gamma_1, \gamma_2, \gamma_3)$	$\left(\mathcal{M}\begin{pmatrix} \gamma_1 & \gamma_2 & \gamma_3 \\ \mathcal{R}_{6,6}^{(1)} & \mathcal{R}_{6,6}^{(2)} & \mathcal{R}_{6,6}^{(3)} \; \mathcal{R}_{6,6}^{(4)} \end{pmatrix}\right)^6$	$\mathcal{R}_{6,6}^{(1)}$	$\mathcal{R}_{6,6}^{(2)}$	$\mathcal{R}_{6,6}^{(3)}$	$\mathcal{R}_{6,6}^{(4)}$	$\mathcal{M}\begin{pmatrix} \gamma_1 & \gamma_2 & \gamma_3 \\ \mathcal{R}_{6,6}^{(1)} & \mathcal{R}_{6,6}^{(2)} & \mathcal{R}_{6,6}^{(3)} \; \mathcal{R}_{6,6}^{(4)} \end{pmatrix}$
(02, 08, 02)	$\begin{pmatrix} AF & 01 & 10 & 05 & 48 & 05 \\ 88 & 86 & 8A & 90 & 80 & 20 \\ F5 & 4D & EB & 15 & 55 & 82 \\ 9A & 18 & C8 & 87 & 41 & 95 \\ 20 & 10 & 90 & 02 & 86 & 0A \\ 48 & 48 & 44 & 10 & 45 & AF \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 00 & 00 & 01 & 00 & 00 & 01 \\ 00 & 00 & 00 & 00 & 00 & 02 \\ 00 & 00 & 00 & 00 & 01 & 08 \\ 01 & 01 & 00 & 00 & 00 & 00 \\ 02 & 00 & 00 & 00 & 00 & 00 \\ 08 & 00 & 00 & 01 & 00 & 00 \end{pmatrix}$
(08, 02, 08)	$\begin{pmatrix} AF & 02 & 10 & 05 & 90 & 05 \\ 44 & 86 & 45 & 48 & 80 & 10 \\ F5 & 9A & EB & 15 & AA & 82 \\ 9A & 30 & C8 & 87 & 82 & 95 \\ 10 & 10 & 48 & 01 & 86 & 05 \\ 48 & 90 & 44 & 10 & 8A & AF \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 00 & 00 & 01 & 00 & 00 & 01 \\ 00 & 00 & 00 & 00 & 00 & 01 \\ 00 & 00 & 00 & 00 & 02 & 08 \\ 01 & 02 & 00 & 00 & 00 & 00 \\ 01 & 00 & 00 & 00 & 00 & 00 \\ 08 & 00 & 00 & 01 & 00 & 00 \end{pmatrix}$
(A9, E1, 91)	$\begin{pmatrix} F8 & 49 & 91 & 86 & 09 & A8 \\ 1C & 13 & 66 & 09 & B5 & A9 \\ 07 & 8F & E4 & 01 & F7 & 79 \\ DD & 2A & 1D & B3 & 8F & 69 \\ A9 & A9 & E5 & C5 & 98 & 43 \\ A8 & 09 & 1C & A9 & 66 & 73 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 00 & 00 & 01 & 00 & 00 & 01 \\ 00 & 00 & 00 & 00 & 00 & 01 \\ 00 & 00 & 00 & 00 & 01 & A9 \\ A9 & 01 & 00 & 00 & 00 & 00 \\ E1 & 00 & 00 & 00 & 00 & 00 \\ 91 & 00 & 00 & 01 & 00 & 00 \end{pmatrix}$

Table 7: Some matrices of the form (7) and it's building blocks for $k = 6$.

Proposition 10. *Let be* $Q = GF(2^8)$. *The matrices of the form* $\left(\mathcal{M}\begin{pmatrix} \gamma_1 & \gamma_2 \\ \mathcal{R}_{4,4}^{(1)} & \mathcal{R}_{4,4}^{(2)} \; \mathcal{R}_{4,4}^{(3)} \end{pmatrix}\right)^4$ *from table 6 are orthomorphic MDS matrices.*

Proof. Similarly, to the proof of the proposition 6. □

Proposition 11. *Let be* $Q = GF(2^8)$. *The matrices of the form* $\left(\mathcal{M}\begin{pmatrix} \gamma_1 & \gamma_2 & \gamma_3 \\ \mathcal{R}_{6,6}^{(1)} & \mathcal{R}_{6,6}^{(2)} & \mathcal{R}_{6,6}^{(3)} \; \mathcal{R}_{6,6}^{(4)} \end{pmatrix}\right)^6$ *from table 7 are orthomorphic MDS matrices.*

Proof. Similarly, to the proof of the proposition 6. □

All characteristic polynomials of the matrices from proposition 10 and 11 are primitive over Q. The high level of view of a round of transformations

$\vec{a} \cdot \left(\mathcal{M}\left(\begin{smallmatrix} 01 & E1 \\ \mathcal{R}_{4,4}^{(1)} & \mathcal{R}_{4,4}^{(2)} \mathcal{R}_{4,4}^{(3)} \end{smallmatrix}\right)\right)^4$ and $\vec{b} \cdot \left(\mathcal{M}\left(\begin{smallmatrix} 02 & 08 & 02 \\ \mathcal{R}_{6,6}^{(1)} & \mathcal{R}_{6,6}^{(2)} & \mathcal{R}_{6,6}^{(3)} \mathcal{R}_{6,6}^{(4)} \end{smallmatrix}\right)\right)^6$ are given in figures 3 and 4, respectively. As we can see these output are the result of a recursive transformations, which can be implemented efficiently. The high level of view of the others matrices displayed in tables 6 and 7 can be represented in a similar fashion to the previous ones.

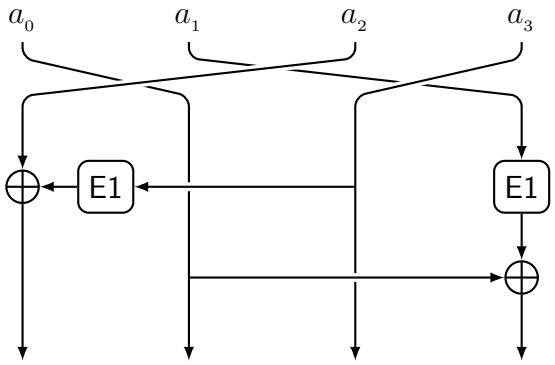


Fig. 3: High level of view of a round of transformation $\vec{a} \cdot \left(\mathcal{M}\left(\begin{smallmatrix} 01 & E1 \\ \mathcal{R}_{4,4}^{(1)} & \mathcal{R}_{4,4}^{(2)} \mathcal{R}_{4,4}^{(3)} \end{smallmatrix}\right)\right)^4$.

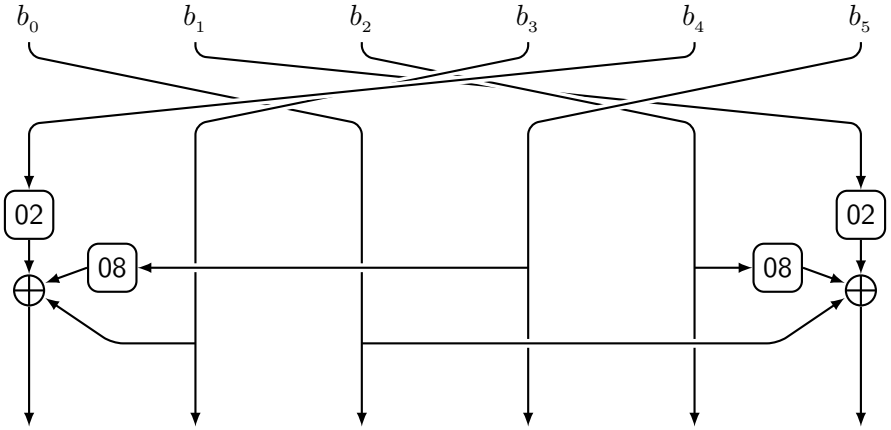


Fig. 4: High level of view of a round of transformation $\vec{b} \cdot \left(\mathcal{M}\left(\begin{smallmatrix} 02 & 08 & 02 \\ \mathcal{R}_{6,6}^{(1)} & \mathcal{R}_{6,6}^{(2)} & \mathcal{R}_{6,6}^{(3)} \mathcal{R}_{6,6}^{(4)} \end{smallmatrix}\right)\right)^6$.

6 XOR-count of some orthomorphic MDS matrices

Let be $f(x),\ g(x) \in Q[x]$ two polynomials of the form (4) and (5), respectively. From (2) we have that for $(\gamma, t) = (02, 7)$ and $(\gamma, \gamma_1, \gamma_2, \gamma_3) = (02, 90, E1, E0)$, $\mathsf{XOR}(S_f^4) = 220$ and $\mathsf{XOR}(S_g^6) = 396$, respectively.

From (3) we have that $\mathsf{XOR}(S_f^4) = \mathsf{XOR}((S_h(\mathcal{P}_{3\times3}))^4) = 220$, where $t = 2,\ h(x) \in Q[x]$ is a polynomial of the form (6) for $\gamma = 02$ and $\mathcal{P}_{3\times3}$ the permutation matrix of the proposition 8. From table 5 we have that $\mathsf{XOR}((S_h(\mathcal{P}_{5\times5}))^6) \in \{318, 342, 360\}$ for any permutation matrix $\mathcal{P}_{5\times5}$, where $h(x) \in \{x^6 \oplus B4x^5 \oplus 08x^4 \oplus 01x^3 \oplus B4x^2 \oplus 08x \oplus B4,\ x^6 \oplus 01x^5 \oplus 04x^4 \oplus$

$\mathsf{02}x^3 \oplus \mathsf{04}x^2 \oplus \mathsf{5A}x \oplus \mathsf{04},\ x^6 \oplus \mathsf{5A}x^5 \oplus \mathsf{5A}x^4 \oplus \mathsf{02}x^3 \oplus \mathsf{04}x^2 \oplus \mathsf{5A}x \oplus \mathsf{02}\}$, respectively.

From (1), tables 6 and 7 we have that

$$\mathsf{XOR}\left(\left(\mathcal{M}\left(\begin{smallmatrix}\mathsf{01} & \mathsf{E1}\\ \mathcal{R}^{(1)}_{4,4} & \mathcal{R}^{(2)}_{4,4}\ \mathcal{R}^{(3)}_{4,4}\end{smallmatrix}\right)\right)^4\right) = 88, \quad \mathsf{XOR}\left(\left(\mathcal{M}\left(\begin{smallmatrix}\mathsf{02} & \mathsf{08} & \mathsf{02}\\ \mathcal{R}^{(1)}_{6,6} & \mathcal{R}^{(2)}_{6,6} & \mathcal{R}^{(3)}_{6,6}\ \mathcal{R}^{(4)}_{6,6}\end{smallmatrix}\right)\right)^6\right) = 312,$$

$$\mathsf{XOR}\left(\left(\mathcal{M}\left(\begin{smallmatrix}\mathsf{01} & \mathsf{91}\\ \mathcal{R}^{(1)}_{4,4} & \mathcal{R}^{(2)}_{4,4}\ \mathcal{R}^{(3)}_{4,4}\end{smallmatrix}\right)\right)^4\right) = 104, \quad \mathsf{XOR}\left(\left(\mathcal{M}\left(\begin{smallmatrix}\mathsf{08} & \mathsf{02} & \mathsf{08}\\ \mathcal{R}^{(1)}_{6,6} & \mathcal{R}^{(2)}_{6,6} & \mathcal{R}^{(3)}_{6,6}\ \mathcal{R}^{(4)}_{6,6}\end{smallmatrix}\right)\right)^6\right) = 312,$$

$$\mathsf{XOR}\left(\left(\mathcal{M}\left(\begin{smallmatrix}\mathsf{A9} & \mathsf{E1}\\ \mathcal{R}^{(1)}_{4,4} & \mathcal{R}^{(2)}_{4,4}\ \mathcal{R}^{(3)}_{4,4}\end{smallmatrix}\right)\right)^4\right) = 104, \quad \mathsf{XOR}\left(\left(\mathcal{M}\left(\begin{smallmatrix}\mathsf{A9} & \mathsf{E1} & \mathsf{91}\\ \mathcal{R}^{(1)}_{6,6} & \mathcal{R}^{(2)}_{6,6} & \mathcal{R}^{(3)}_{6,6}\ \mathcal{R}^{(4)}_{6,6}\end{smallmatrix}\right)\right)^6\right) = 324.$$

where the matrices $\mathcal{R}^{(1)}_{4,4}, \mathcal{R}^{(2)}_{4,4}, \mathcal{R}^{(3)}_{4,4}$ and $\mathcal{R}^{(1)}_{6,6}, \mathcal{R}^{(2)}_{6,6}, \mathcal{R}^{(3)}_{6,6}, \mathcal{R}^{(4)}_{6,6}$ are given in table 6 and 7, respectively.

7 Conclusion and Future works

In this work we have introduced a new concept of the $\mathcal{P}$ - companion matrix wich not only generalize the notion of companion matrix but also can be used to extend the class of MDS mappings that can be obtained using this kind of matrices. Also we have presented some new constructions based on the use of recursive schemes for generating a special kind of MDS matrices (here called orthomorphic MDS matrices) of dimension 4×4 and 6×6, respectively. The main advantage of ours MDS orthomorphic matrices is the absence of invariant subspaces, due to the irreducibility of their characteristic polynomials. It is well known that the presence of such subspaces in block ciphers can be exploited by an adversary in order to distinguish it from a random permutation. Finally, for the proposed matrices we have analyzed the XOR-count metric and the obtained results shows that these matrices could be attractive for the so-called lightweight schemes offering a good trade offs between security and implementation. In the future, we aim to further optimise the search for constructing orthomorphic MDS matrices of size 8×8 using our methods.

Acknowledgements. The authors are very grateful to Oleg V. Kamlovskiy and the anonymous reviewers of CTCrypt'2020 for their useful comments and valuable observations, which helped to improve the final version of this article.

References

[1] Glukhov M.M., Elizarov V.P., Nechaev A.A., *Algebra: Uchebnik. - izdanie vtoroe, ispravlennoe i dopolnennoe. [Algebra: Textbook. - second edition, revised and supplemented.]*, Izdatelstvo Lan, Sankt-Peterburg, Moskva, Krasnodar, 2015, In Russian.

[2] Lidl R., Niederreiter H., *Introduction to finite fields and their applications*, Cambridge University Press, London, New York, New Rochelle, Melbourne, Sydney, 1986.

[3] Zhe-Xian Wan., *Lectures on Finite Fields and Galois Rings*, Beijing, 2003.

[4] Zierler N., "Linear recurring sequences", J. Soc. Indust. Appl. Math., 1959, 31–48.

[5] Augot D.,Finiasz M., "Direct construction of recursive MDS diffusion layers using shortened BCH codes", International Workshop on Fast Software Encryption, 2014, 3–17.

[6] Barreto P., Rijmen V., "The Khazad Legacy-Level Block Cipher", First Open NESSIE Workshop, 2000, 79.p.

[7] Burov D. A., Pogorelov B. A., "The influence of linear mapping reducibility on the choice of round constants.", 2017, 51–64..

[8] Burov D. A., Pogorelov B. A., "The permutation group insight on the diffusion property of linear mappings", 2018, 47–58.

[9] GOST R 34.12-2015. Information technology. Cryptographic protection of information. Block ciphers Moscow: Standartinform, 2015 (in Russian)..

[10] Gupta K.C., Ray I.G., "On constructions of MDS matrices from companion matrices for lightweight cryptography", International Conference on Availability, Reliability, and Security, 2013, 29–43.

[11] Toh D., Teo J., Khoo K., Sim S.M., "Lightweight MDS serial-type matrices with minimal fixed XOR count", International Conference on Cryptology in Africa, 2018, 51–71.

[12] Sarkar S., Sim S. M., "A deeper understanding of the XOR count distribution in the context of lightweight cryptography", International Conference on Cryptology in Africa, 2016, 167–182.

[13] Pogorelov B. A., Pudovkina M. A., "On the distance from permutations to imprimitive groups for a fixed system of imprimitivity", *Discrete Math. Appl., 24:2*, 2014, 95–108.(in Russian).

[14] Pogorelov B. A., Pudovkina M. A, "Factor structures of transformations", *Mathematical Aspects of Cryptography, 3:3*, 2012, 81-104.(in Russian).

[15] Pogorelov B. A., Pudovkina M. A., "Combinatorial characterization of XL-layers", *Mathematical Aspects of Cryptography, 4:3*, (2013), 99–129 (in Russian).

[16] O. Coy Puente, R. A. De La Cruz Jiménez, "Some methods for constructing MDS matrices over finite fields", Prikl. Diskr. Mat., 2019, 5–18 (in Russian).

[17] *http://www.sagemath.org. — Sage Mathematics Software Version 8.1*, 2018.

Construction of MDS matrices combining the Feistel, Misty and Lai-Massey schemes

Ramses Rodriguez Aulet and Reynier Antonio de la Cruz Jiménez

Institute of Cryptography, Havana University, Cuba.
ramsesrusia@yahooo.com, djr.antonio537@gmail.com

Abstract

In Cryptography maximum distance separable (MDS) matrices are an important structural element to provide the diffusion property in the block ciphers, stream ciphers and hash functions. To discover new kind of transformations that can generate a series of new MDS matrices which could be used in practice is not a trivial task. In this article we propose new methods for constructing MDS matrices of size 4×4 combining the well-known Feistel, Misty and Lai-Massey structures.

Keywords: Diffusion, Involutory matrix, Almost involutory matrix, MDS matrix .

1 Introduction

In the work [3] Claude Shannon defines confusion and diffusion as two properties necessary for constructing strong cryptographic functions; these properties are also required for hash functions. One strategy to obtain maximum diffusion and avoid linear and differential attacks is to use global linear mappings with optimal diffusion, combined with the local nonlinear mappings (S-Boxes) (see,[8, 9, 10]). The linear transformations choose by designers should be able to spread the internal dependencies as much as possible. Hence, designers commonly used optimal diffusion matrices called Maximum Distance Separable (because they are related to a Maximum Distance Separable code) matrices to maximise the diffusion ability of the diffusion layer. Example of their use can be found no only in the design of block ciphers like AES, TwoFish, KHAZAD, Picaro, etc, but also in the Hash function PHOTON [13], Whirlpool, Grostl, and even in stream ciphers (MUGI).

From practical point of view, is not only desirable that an MDS matrix can be implemented efficiently both in software/hardware but also when encryption and decryption implementations are required and the inverse of the MDS matrix will have to be implemented as well (except for Feistel and Lai-Massey structures, where the inverse of the internal function is not required

for decryption). For this reason it is of a great significance that one can use exactly (or almost exactly) the same linear transformarion for encryption and decryption. One strategy to achieve this goal is employing involutory MDS matrices and we can found several ciphers like Anubis, Khazad , Iceberg or Prince that using this approach have the same implementation for encryption and decryption.

The construction of MDS matrices, is not an easy problem to solve. There are several ways for constructing such matrices, for instances: using the Cauchy and Hadamard matrices [16]. In [1] it is shown that it is possible to build involutive binary matrices with a high degree of diffusion by exploiting the properties of the Feistel network and in [15] it is shown that using the general Feistel networks it is possible to build MDS matrices on finite fields, in this case the authors do not build involutory MDS matrices. The aim of this article is to build involutory or almost involutory MDS matrices combining the Feistel, Misty and Lai-Massey schemes.

This article is structured as follows: In Section 1 we give the basic definitions and some results about MDS matrices. Some constructions which can generate MDS matrices and linear orthomorphisms are presented in Section 2. Some examples of MDS matrices obtained by our approach is given in Section 3. We provide an implementation of a concrete matrix in Section 4. A comparison with the state-oh-the-art is performed in Section 5. Our work is concluded in Section 6.

2 Preliminaries and Basic Definitions

Let be $P = GF(2^t) = GF(2)[x]/g(x)$ finite field with 2^t elements, for some irreducible polynomial $g(x)$ of degree t. The vector space of dimension n over P is denoted by P^n. We use the notation $P_{n,n}$ for the ring of $n \times n$ matrices over finite field P. Throughout the article, we shall use the following operations and notations:

1	- the neutral element of the multiplicative group P^*;
$\oplus$	- addition in GF(2^t);
$w_H(\vec{a})$	- the Hamming weight of a vector $\vec{a} \in P^n$, i.e. the number of its nonzero coordinates;
$\omega(\mathcal{M})$	- the number of 1's in the matrix $\mathcal{M}$;
Ψ^{-1}	- the inverse transformation to some invertible mapping Ψ;
$I_{n,n}$	- the identity matrix of $P_{n,n}$.
$O_{n,n}$	- the zero matrix of $P_{n,n}$.

$|A|$ - the determinant of the matrix of $A \in P_{n,n}$.

In what follows, for the sake of simplicity, a matrix over finite field P will be written in hexadecimal notation.

Definition 1. *An transformation $\varphi : P^n \to P^n$ is called involutive, if $\forall \alpha \in P^n$ the following equality hold $\varphi(\varphi(\alpha)) = \alpha$.*

Clearly, if φ is an involutive transformation then for any $\phi : P^n \to P^n$ the transformation $\hat{\varphi} = \phi \circ \varphi \circ \phi^{-1}$ will be an ivolution too.

Definition 2. *An transformation $\varphi : P^n \to P^n$ is called linear transformation, if the following relation holds*

$$\forall, \vec{\alpha}, \vec{\beta} \in P^n, a_1, a_2 \in P : \varphi(a_1\vec{\alpha} + a_2\vec{\beta}) = a_1\varphi(\vec{\alpha}) + a_2\varphi(\vec{\beta}), \qquad (1)$$

It is shown in [6] that the composition of linear transformations is again a linear transformation.

Definition 3. *Let be $\vec{\alpha} = (\alpha_1, \ldots, \alpha_n)$ a basis of the vector space P^n. The matrix $A_{\vec{\alpha}}(\varphi) \in P_{n,n}$ defined as follows*

$$A_{\vec{\alpha}}(\varphi) = (\varphi(\alpha_1)^{\downarrow}_{\vec{\alpha}}, \ldots, \varphi(\alpha_1)^{\downarrow}_{\vec{\alpha}}) \qquad (2)$$

is called the matrix associated with the linear transformation φ in the basis $\vec{\alpha}$.

Definition 4. *The branch number ρ of matrix $A \in P_{n,n}$ is defined as*

$$\rho(A) = \min_{\vec{a} \neq \vec{0}} \{w_H(\vec{a}) + w_H(\vec{a}A)\}. \qquad (3)$$

Definition 5. *A matrix $A \in P_{n,n}$ is called maximal distance separable (MDS) matrix if $\rho(A) = n + 1$.*

Theorem 1. *Matrix A is an MDS matrix if and only if every sub-matrix is non-singular.*

Proposition 1. *[11] Any 4×4 matrix over P with all entries non zero is an MDS matrix if and only if it is a full rank matrix with the inverse matrix having all entries non zero and all of its 4×4 submatrices are full rank.*

For efficient implementation of an MDS matrix in software, it is desirable to have maximum number of 1's in the matrix. In [12], authors studied this property and constructed some matrices achieving the maximum number of 1's. Here we restate the definition of the number of occurrences of one, which we will use in our constructions.

Definition 6. *Let be $A = (a_{ij})_{n\times n}$ an arbitrary matrix over P. The number of occurrences of one in A denoted by $\mathcal{N}_1(A)$ is the the number of (i, j) pairs such that a_{ij} is equal to one.*

It is well known from [12] that for any MDS matrix $A \in P_{4,4}$ we have $\mathcal{N}_1(A) = 9$ and $\mathcal{N}_1(A) = 16$ when A is an MDS matrix of $P_{6,6}$.

Definition 7. *Let be $A = (a_{ij})_{n\times n}$ an arbitrary matrix over P. We say that A has the almost involutory property if*

1. $A^{-1} \neq A$;
2. *All coefficients of A can be found in A^{-1} too.*

For example, let be $P = GF(2^4)/x^4 \oplus x \oplus 1$ and $\mathcal{M}_{2\times 2} = \left(\begin{smallmatrix} 0\text{x}1 & 0\text{x}C \\ 0\text{x}C & 0\text{x}E \end{smallmatrix}\right) \in P_{2,2}$. It can be easy checked that $\mathcal{M}^{-1}_{2\times 2} = \left(\begin{smallmatrix} 0\text{x}E & 0\text{x}C \\ 0\text{x}C & 0\text{x}1 \end{smallmatrix}\right) \in P_{2,2}$ and the coefficients of $\mathcal{M}_{2\times 2}$ are present in $\mathcal{M}^{-1}_{2\times 2}$ too, so this matrix has the almost involutory property. Other example of a matrix having the almost involutory property can be found in the linear layer of the block cipher Kuznyechik which can be expresed as a power of the companion matrix of the follwowing polynomial $h(y) = y^{16} \oplus 0\text{x}94y^{15} \oplus 0\text{x}20y^{14} \oplus 0\text{x}85y^{13} \oplus 0\text{x}10y^{12} \oplus 0\text{x}C2y^{11} \oplus 0\text{x}C0y^{10} \oplus 0\text{x}01y^9 \oplus 0\text{x}FBy^8 \oplus 0\text{x}01y^7 \oplus 0\text{x}C0y^6 \oplus 0\text{x}C2y^5 \oplus 0\text{x}10y^4 \oplus 0\text{x}85y^3 \oplus 0\text{x}20y^2 \oplus 0\text{x}94y \oplus 0\text{x}01$ over $P = GF(2^8)/x^8 \oplus x^7 \oplus x^6 \oplus x \oplus 1$.

We can see that involutory and almost involutory MDS matrices can be useful when implementing the inverse of an SPN cipher, because the inverse of these kind of matrices can also be implemented efficiently.

Proposition 2. *If $A \in P_{n,n}$ is an involutory MDS matrix and $\Pi \in P_{n,n}$ is permutation matrix then the matrix $A\Pi$ and ΠA are almost involutory MDS.*

Proof. Let be A involutive MDS matrix and Π permutation matrix then $\Pi{-}1$ is permutation matrix. Then we have that $(A\Pi)(\Pi^{-1}A) = A(\Pi\Pi^{-1})A = AA = I_{n,n}$ and taking into account that Π is permutation matrix we obtain that AS is an MDS matrix which has the almost involutory property. □

Definition 8. *The characteristic polynomial of a linear transformation of a matrix $A \in P_{n,n}$, denoted by $\chi_A(x)$, is defined as follow*

$$\chi_A(x) = |I_{n,n}x \oplus A|. \tag{4}$$

In work [4] the authors showed the possibility of invariant attacks on the cipher type XSL-network (Khazad, Kuznyechik) where $(x + 1)$ divide the characteristic polynomial of the lineal transformation. For this reason we

will study the characteristic polynomial of those matrices generate by our constructions.

Every $n \times n$ matrix over P can be written as an $(tn) \times (tn)$ matrix over $GF(2)$. When considering a hardware implementation, it is natural to consider only matrices over $GF(2)$. Measurements of implementation costs will then only involve the number of bit-operations (XORs) needed. It is an interesting question to evaluate the efficiency of a given matrix. The following metrics are useful for estimating the hardware cost of a linear operation.

1. **Direct XOR Count**. Given a matrix $\mathcal{M} \in GF(2)_{t\times n,t\times n}$, the direct XOR count $\mathsf{DXC}(\mathcal{M})$ of $\mathcal{M}$ is $\omega(\mathcal{M}) - nt$. This metric corresponds to counting the number of gates used in a naive implementation of the linear mapping $\mathcal{M}$.

2. **Global Optimization**. For a matrix $\mathcal{M} \in GF(2)_{t\times n,t\times n}$, it is posible to obtain an estimation of its cost in hardware by finding a good linear straight-line program corresponding to $\mathcal{M}$ with state-of-the-art automatic tools based on certain SLP[1] heuristic [2], and this metric is denoted as $\mathsf{SLP}(\mathcal{M})$.

3 Constructing MDS matrices combining the Feistel, Misty and Lai-Massey transformations

Let be $n = 2k$ an even number, in what follows $\vec{x} = (\vec{x_1}||\vec{x_2})$ where $\vec{x_1} = (x_1, \ldots, x_k)$ and $\vec{x_2} = (x_{k+1}, \ldots, x_{2k})$. For any $\mathcal{L} \in P_{n,n}$ using the well-known Lai-Massey and Feistel schemes we define the following transformation as follows;

Lai-Massey-like transformation:

$$\varphi_1(\vec{x}) = \big(\vec{x_1} \oplus \mathcal{L}(\vec{x_1} \oplus \vec{x_2})\big) \big|\big| \big(\vec{x_2} \oplus \mathcal{L}(\vec{x_1} \oplus \vec{x_2})\big). \tag{5}$$

Feistel-like transformation:

$$\varphi_2(\vec{x}) = \big(\vec{x_1} \oplus \mathcal{L}(\vec{x_2})\big) \big|\big| \vec{x_2}. \tag{6}$$

Misty-like transformation:

$$\varphi_3(\vec{x}) = \mathcal{L}(\vec{x_1} \oplus \vec{x_2}) \big|\big| \vec{x_2}. \tag{7}$$

it is not difficult to see that the transformations given by relations (5) and (6) are involutions and the inverse of the Misty-like transformation given by (7) is defined by the following formula: $\varphi_3^{-1}(\vec{x}) = \big(\mathcal{L}^{-1}(\vec{x_1}) \oplus \vec{x_2}\big) \big|\big| \vec{x_2}$.

[1]Note that this is so far the most accurate estimation that is practical for 32×32 binary matrices.

Using the matrix given by relation (2), canonical basis of P^4

$$\vec{\alpha}_4 = \{(1,0,0,0),(0,1,0,0),(0,0,1,0),(0,0,0,1)\},$$

and the previous transformations, we construct the following matrices of dimension $n \times n$, $n = 4$, as follows

Construction of $\mathcal{M}_{n\times n}^{\Phi_A}$

Let $\Phi_A = \varphi_2 \circ \varphi_1 \circ \varphi_2$. Then $\mathcal{M}_{4\times 4}^{\Phi_A} = A_{\vec{\alpha_4}}(\Phi_A)$;

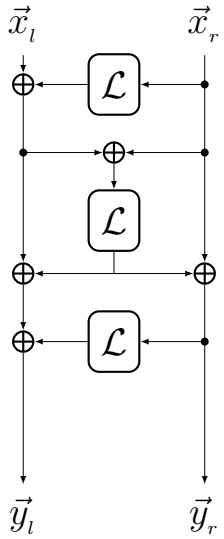


Fig. 1: Structure of Φ_A.

Construction of $\mathcal{M}_{n\times n}^{\Phi_B}$

Let $\Phi_B = \varphi_1 \circ \varphi_2 \circ \varphi_1$. Then $\mathcal{M}_{4\times 4}^{\Phi_B} = A_{\vec{\alpha_4}}(\Phi_B)$;

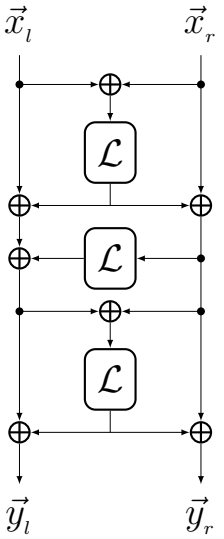


Fig. 2: Structure of Φ_B.

Construction of $\mathcal{M}_{n\times n}^{\Phi_C}$

Let $\Phi_C = \varphi_3 \circ \varphi_2 \circ \varphi_3^{-1}$. Then $\mathcal{M}_{4\times 4}^{\Phi_C} = A_{\vec{\alpha_4}}(\Phi_C)$;

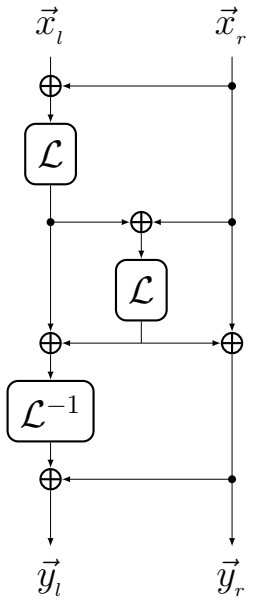


Fig. 3: Structure of Φ_C.

Let $n = 4$ and $f_1(x) = x^2 \oplus x \oplus 1$, $f_2(x) = x^4 \oplus x^3 \oplus 1$, $f_3(x) = x^4 \oplus x^3 \oplus x^2 \oplus x \oplus 1$ —some polynomials over P. In what follows, we shall work over the finite field $P = GF(2^8)$.

Proposition 3. *Let be an element $a \in P^*$, $a \neq 1$ for which $f_1(a) \neq 0$. The matrix $\mathcal{M}_{4\times4}^{\Phi_A}$ of transformation Φ_A with $\mathcal{L} = \begin{pmatrix} a & 1 \\ 1 & a \end{pmatrix}$ is an involutory MDS matrix.*

Proof. The matrix $\mathcal{M}_{n\times n}^{\Phi_A} \in P_{4,4}$, for $\mathcal{L} = \begin{pmatrix} a & 1 \\ 1 & a \end{pmatrix} \in P_{2,2}$, has the following form

$$\mathcal{M}_{4\times4}^{\Phi_A} = \begin{pmatrix} a^2 \oplus a & 1 & a & 1 \\ 1 & a^2 \oplus a & 1 & a \\ a^3 & a^2 & a^2 \oplus a & 1 \\ a^2 & a^3 & 1 & a^2 \oplus a \end{pmatrix} \tag{8}$$

Taking into account that $\mathcal{M}_{4\times4}^{\Phi_A} = (\mathcal{M}_{4\times4}^{\Phi_A})^{-1}$ we only need to check according with proposition 1 that all minors of order 2 of $\mathcal{M}_{4\times4}^{\Phi_A}$ are nonzero over P. These minors are on the following set

$\{1, a^5 \oplus a^4 \oplus a^2, a, a^2, a^3, a^4, a^4 \oplus 1, a^2 \oplus a, a^3 \oplus a^2 \oplus 1, a^4 \oplus a^2 \oplus 1, a^6 \oplus a^4, a^2 \oplus 1, a^3 \oplus a^2 \oplus a\}$

whose factors are

$\{1, a^2 \cdot (a^3 \oplus a^2 \oplus 1), a, a^2, a^3, a^4, (a \oplus 1)^4, a \cdot (a \oplus 1), (a^3 \oplus a^2 \oplus 1), (a^2 \oplus a \oplus 1)^2, (a \oplus 1)^2 \cdot a^4, (a \oplus 1)^2, a \cdot (a^2 \oplus a \oplus 1)\}$

The polynomial $x^3 \oplus x^2 \oplus 1$ is irreducible over field P. Therefore for any nonzero $a \in P$ such that

$$\begin{aligned} \alpha &\neq 0, \\ \alpha \oplus 1 &\neq 0, \\ \alpha^2 \oplus \alpha \oplus 1 &\neq 0, \end{aligned}$$

the matrix $\mathcal{M}_{4\times4}^{\Phi_A}$ is an involutory MDS matrix over P. □

Proposition 4. *Let be an element $a \in P^*$, $a \neq 1$, for which $f_1(a) \neq 0$ then the matrix $\mathcal{M}_{4\times4}^{\Phi_B}$ of transformation Φ_B with $\mathcal{L} = \begin{pmatrix} 1 & 1 \\ a & 1 \end{pmatrix}$ is an involutory MDS.*

Proof. The matrix $\mathcal{M}_{n\times n}^{\Phi_B} \in P_{4,4}$, for $\mathcal{L} = \begin{pmatrix} 1 & 1 \\ a & 1 \end{pmatrix} \in P_{2,2}$, has the following form

$$\mathcal{M}_{4\times4}^{\Phi_B} = \begin{pmatrix} 1 & a\oplus 1 & a\oplus 1 & a\oplus 1 \\ a^2\oplus a & 1 & a^2\oplus a & a\oplus 1 \\ a & a & 1 & a\oplus 1 \\ a^2 & a & a^2\oplus a & 1 \end{pmatrix} \tag{9}$$

So, using the fact that $\mathcal{M}_{4\times4}^{\Phi_B} = (\mathcal{M}_{4\times4}^{\Phi_B})^{-1}$ we only need to check in correspondence with proposition 1 that all minors of order 2 of $\mathcal{M}_{4\times4}^{\Phi_B}$ are nonzero over P. These minors are on the following set

$\{1, a^3, a, a\oplus 1, a^2\oplus 1, a^3\oplus 1, a^3\oplus a^2, a^2\oplus a, a^2\oplus a\oplus 1, a^3\oplus a\oplus 1, a^3\oplus a^2\oplus 1, a^3\oplus a, a^2, a^3\oplus a^2\oplus a, a^3\oplus a^2\oplus a\oplus 1\}$

whose factors are

$\{1, a^3, a, (a\oplus 1), (a\oplus 1)^2, (a\oplus 1)\cdot(a^2\oplus a\oplus 1), (a\oplus 1)\cdot a^2, a\cdot(a\oplus 1), (a^2\oplus a\oplus 1), (a^3\oplus a\oplus 1), (a^3\oplus a^2\oplus 1), a\cdot(a\oplus 1)^2, a^2, a\cdot(a^2\oplus a\oplus 1), (a\oplus 1)^3\}$

The polynomials $x^3\oplus x^2\oplus 1$ and $x^3\oplus x\oplus 1$ are irreducible over field P. Therefore for any nonzero $a\in P$ such that

$$\begin{aligned} \alpha &\neq 0, \\ \alpha\oplus 1 &\neq 0, \\ \alpha^2\oplus\alpha\oplus 1 &\neq 0, \end{aligned}$$

the matrix $\mathcal{M}_{4\times4}^{\Phi_B}$ is an involutory MDS matrix over P. □

Proposition 5. *Let be an element $a\in P^*$, $a\neq 1$ for which $f_i(a)\neq 0$ where $i=1,2,3$ then the matrix $\mathcal{M}_{4\times4}^{\Phi_C}$ of transformation Φ_C with $\mathcal{L}=\begin{pmatrix} a & 1 \\ a & a^2 \end{pmatrix}$ is an involutory MDS matrix.*

Proof. Similar to the previuos ones. □

3.1 Constructing MDS matrices which are linear orthomorphisms

Definition 9. *The linear mapping $\mathcal{L}: P^k\to P^k$, defined as $\mathcal{L}(\vec{a})=\vec{a}\cdot A$, where $\vec{a}\in P^k, A\in P_{k,k}^*$ is called a linear orthomorphism if the matrix $A\oplus I_{k\times k}$ is invertible over P.*

For the following permutations matrices

$$\Pi_1=\begin{pmatrix} 0&0&1&0\\0&0&0&1\\1&0&0&0\\0&1&0&0\end{pmatrix}, \Pi_2=\begin{pmatrix} 0&0&0&1\\1&0&0&0\\0&1&0&0\\0&0&1&0\end{pmatrix} \text{ and } \Pi_3=\begin{pmatrix} 0&0&0&1\\0&1&0&0\\1&0&0&0\\0&0&1&0\end{pmatrix}.$$

The corresponding resulting matrices $\mathcal{M}_{4\times4}^{\Phi_A}\circ\Pi_1, \mathcal{M}_{4\times4}^{\Phi_B}\circ\Pi_2$ and $\mathcal{M}_{4\times4}^{\Phi_C}\circ\Pi_3$ are linear orthomorphisms over P, where

$$\mathcal{M}_{4\times4}^{\Phi_*}, *\in\{A,B,C\},$$

are defined in propositions 3,4, and 5 respectively.

It can be checked that all these otrhomorphic MDS matrices have irreducible characteristic polynomial. So we can conclude that the main advantage of ours MDS orthomorphic matrices is the absence of invariant subspaces, due to the irreducibility of their characteristic polynomials. It is well known that the presence of such subspaces in block ciphers can be exploited by an adversary in order to distinguish it from a random permutation.

4 Some examples

In Table 1 we list certain properties of some MDS matrices generated by ours constructions over the finite field $P = GF(2^8)/x^8 \oplus x^7 \oplus x^6 \oplus x \oplus 1$.

Matrix M	$\mathcal{L}$	Involutory	Almost involotory	$\mathcal{N}_1(M)$	$\chi_M(x)$	$\chi_M(x)$ is irreducible over P	Factorization of $\chi_M(x)$
$\mathcal{M}_{4\times4}^{\Phi_A} = \begin{pmatrix} 0x08 & 0x04 & 0x06 & 0x01 \\ 0x04 & 0x08 & 0x01 & 0x06 \\ 0x06 & 0x01 & 0x02 & 0x01 \\ 0x01 & 0x06 & 0x01 & 0x02 \end{pmatrix}$	$\begin{pmatrix} 0x02 & 0x01 \\ 0x01 & 0x02 \end{pmatrix}$	No	Yes	6	$x^4 \oplus 0x0Fx^2 \oplus 0x01$	No	$(x^2 \oplus 0x0Fx \oplus 0x01)^2$
$\mathcal{M}_{4\times4}^{\Phi_A} = \begin{pmatrix} 0x06 & 0x01 & 0x08 & 0x04 \\ 0x01 & 0x06 & 0x04 & 0x08 \\ 0x02 & 0x01 & 0x06 & 0x01 \\ 0x01 & 0x02 & 0x01 & 0x06 \end{pmatrix}$	$\begin{pmatrix} 0x02 & 0x01 \\ 0x01 & 0x02 \end{pmatrix}$	yes	no	6	$(x \oplus 0x1)^4$	No	$(x \oplus 0x1)^4$
$\mathcal{M}_{4\times4}^{\Phi_A} = \begin{pmatrix} 0x9B & 0x01 & 0x56 & 0x43 \\ 0x01 & 0x47 & 0x43 & 0x01 \\ 0x56 & 0x43 & 0x56 & 0x04 \\ 0x43 & 0x01 & 0x04 & 0x01 \end{pmatrix}$	$\begin{pmatrix} 0x04 & 0x01 \\ 0x56 & 0x04 \end{pmatrix}$	No	Yes	5	$x^4 \oplus 0x8Bx^3 \oplus 0x1Bx^2 \oplus 0x8Bx \oplus 0x01$	Yes	–
$\mathcal{M}_{4\times4}^{\Phi_B} = \begin{pmatrix} 0x01 & 0x06 & 0x02 & 0x04 \\ 0x03 & 0x01 & 0x02 & 0x02 \\ 0x03 & 0x06 & 0x01 & 0x06 \\ 0x03 & 0x03 & 0x03 & 0x01 \end{pmatrix}$	$\begin{pmatrix} 0x01 & 0x01 \\ 0x02 & 0x01 \end{pmatrix}$	Yes	No	4	$(x \oplus 0x1)^4$	No	$(x \oplus 0x1)^4$
$\mathcal{M}_{4\times4}^{\Phi_B} = \begin{pmatrix} 0x06 & 0x02 & 0x04 & 0x01 \\ 0x01 & 0x02 & 0x02 & 0x03 \\ 0x06 & 0x01 & 0x06 & 0x03 \\ 0x03 & 0x03 & 0x01 & 0x03 \end{pmatrix}$	$\begin{pmatrix} 0x01 & 0x01 \\ 0x02 & 0x01 \end{pmatrix}$	No	Yes	4	$x^4 \oplus x^3 \oplus 0x0Fx^2 \oplus 0x02x \oplus 0x1$	No	$(x \oplus 0x6f)(x^3 \oplus 0x6Ex^2 \oplus 0xD2x \oplus 0x24)$
$\mathcal{M}_{4\times4}^{\Phi_B} = \begin{pmatrix} 0x01 & 0x02 & 0x8F & 0x01 \\ 0x02 & 0x08 & 0x01 & 0x0C \\ 0x8F & 0x01 & 0x8F & 0x03 \\ 0x01 & 0x0C & 0x03 & 0x0C \end{pmatrix}$	$\begin{pmatrix} 0x01 & 0x04 \\ 0x8E & 0x01 \end{pmatrix}$	No	Yes	5	$x^4 \oplus 0x8Ax^3 \oplus 0xFAx^2 \oplus 0x8Ax \oplus 0x01$	Yes	–
$\mathcal{M}_{4\times4}^{\Phi_C} = \begin{pmatrix} 0x05 & 0x0e & 00x7 & 0x0c \\ 0x07 & 0x017 & 0x06 & 0x13 \\ 0x06 & 0x0c & 0x05 & 0x0e \\ 0x06 & 0x12 & 0x07 & 0x17 \end{pmatrix}$	$\begin{pmatrix} 0x02 & 0x01 \\ 0x02 & 0x04 \end{pmatrix}$	Yes	No	0	$(x \oplus 0x1)^4$	No	$(x \oplus 0x1)^4$
$\mathcal{M}_{4\times4}^{\Phi_C} = \begin{pmatrix} 0x07 & 0x0e & 0x0c & 0x05 \\ 0x06 & 0x17 & 0x13 & 0x07 \\ 0x05 & 0x0c & 0x0e & 0x06 \\ 0x07 & 0x12 & 0x17 & 0x06 \end{pmatrix}$	$\begin{pmatrix} 0x02 & 0x01 \\ 0x02 & 0x04 \end{pmatrix}$	No	Yes	0	$x^4 \oplus 0x18x^3 \oplus 0x1Ax^2 \oplus 0x1Ax \oplus 0x1$	No	$(x \oplus 0x37)(x \oplus 0xCC)(x^2 \oplus 0xE3x \oplus 0xF)$
$\mathcal{M}_{4\times4}^{\Phi_C} = \begin{pmatrix} 0x87 & 0xCB & 0x93 & 0x17 \\ 0x28 & 0x1D & 0x7B & 0x36 \\ 0xCB & 0x87 & 0x16 & 0x93 \\ 0x1D & 0x29 & 0x36 & 0x7B \end{pmatrix}$	$\begin{pmatrix} 0x84 & 0x4C \\ 0x4D & 0x35 \end{pmatrix}$	No	Yes	0	$x^4 \oplus F7x^3 \oplus 65x^2 \oplus F7x \oplus 1$	Yes	–

Table 1: Properties of some matrices generated by ours constructions.

As we can see from Table 1, MDS matrices for different linear components $\mathcal{L}$ can be obtained. This Table also shows some information about the involutory and almost involutory properties, the number of ones of the displayed matrices and the factorisation of its characteristic polynomials over P.

5 Implementation of concrete $\mathcal{M}_{4\times4}^{\Phi_A}$

In this section we provide both software/hardware implementations for a candidate matrix which has been found using a search algorithm over the

structure of $\mathcal{M}_{4\times4}^{\Phi_A}$ with the following different matrices $\mathcal{L}_i \in P_{2,2}, i = 1, 2, 3$, where

$$\mathcal{L}_1 = \begin{pmatrix} 0\text{x}02 & 0\text{x}02 \\ 0\text{x}E1 & 0\text{x}03 \end{pmatrix}, \mathcal{L}_2 = \begin{pmatrix} 0\text{x}E1 & 0\text{x}03 \\ 0\text{x}E1 & 0\text{x}02 \end{pmatrix}, \mathcal{L}_3 = \mathcal{L}_1.$$

Then, the resulting matrix $\mathcal{M}_{4\times4}^{\Phi_A} \in P_{4,4}$, denote by $\mathcal{M}_{\mathsf{RR}}$ for simplicity, has the following form

$$\mathcal{M}_{\mathsf{RR}} = \begin{pmatrix} 0\text{x}01 & 0\text{x}07 & 0\text{x}E1 & 0\text{x}03 \\ 0\text{x}E1 & 0\text{x}04 & 0\text{x}E1 & 0\text{x}02 \\ 0\text{x}01 & 0\text{x}03 & 0\text{x}E0 & 0\text{x}01 \\ 0\text{x}01 & 0\text{x}E8 & 0\text{x}90 & 0\text{x}E5 \end{pmatrix} \in P_{4,4}, \tag{10}$$

where $P = GF(2^8)/x^8 \oplus x^7 \oplus x^6 \oplus x \oplus 1$.

5.1 Software implementation

The following code describe a way for implementing the multiplication by the matrix $\mathcal{M}_{\mathsf{RR}}$ in $\mathsf{C}++$ program language.

```
static uint8_t xtime(uint8_t x)
{
  return ((x<<1) ^ (((x>>7) & 1) * 0xC3));
}

static uint8_t xtimes(uint8_t x, int ts)
{
    while (ts-- > 0) {
        x = xtime(x);
    }

    return x;
}

static uint8_t mult(uint8_t x, uint8_t y)
{
    return ((((y >> 0) & 1) * xtimes(x, 0)) ^
            (((y >> 1) & 1) * xtimes(x, 1)) ^
            (((y >> 2) & 1) * xtimes(x, 2)) ^
            (((y >> 3) & 1) * xtimes(x, 3)) ^
            (((y >> 4) & 1) * xtimes(x, 4)) ^
            (((y >> 5) & 1) * xtimes(x, 5)) ^
            (((y >> 6) & 1) * xtimes(x, 6)) ^
            (((y >> 7) & 1) * xtimes(x, 7)) );
}

static uint32_t M (uint32_t x){
```

```
  uint8_t a = x, b = x>>8, c = x>>16, d = x>>24, e1, e2;

  d ^= mult(a, 0xE1) ^ mult(b, 0x2);
  c ^= mult((a^b), 0x02) ^ a;
  e1 = mult((a^b^c^d), 0xE1);
  e2 = mult((a^b^c^d), 0x02) ^ (b^d);
  d ^= e1;
  c ^= e2;
  b ^= e1;
  a ^= e2;
  d ^= mult(a, 0xE1) ^ mult(b, 0x2);
  c ^= mult((a^b), 0x02) ^ a;

  return ((((((uint32_t)d<<8)|c)<<8)|b)<<8)|a;
}
```

As it can be seen from the code, the number of field multiplications is decreased up to 8 in comparison with the direct multiplication of $\vec{x} \in P^4$ by $\mathcal{M}_{\mathsf{RR}}$. Other matrices generated by our methods can be implemented in a similar way.

5.2 Hardware implementation

The multiplication operation by the MDS matrix given in (10) has the following binary representation over $GF(2)_{32,32}$.

$$\mathcal{M}_{\mathsf{RR}} = \left(\begin{array}{cccc|cccc|cccc|cccc}
1&0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&1&1&0&0&0&0&0&0&1&0&0&0&0&0&0&1\\
0&1&0&0&0&0&0&0&1&1&0&0&0&0&1&1&0&0&1&0&0&0&0&0&1&1&0&0&0&0&0&1\\
0&0&1&0&0&0&0&0&1&1&1&0&0&0&0&1&0&0&0&1&0&0&0&0&0&1&1&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&1&1&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&1&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&1&1&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&1&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&1&1&1&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&1&0&0\\
0&0&0&0&0&0&1&0&0&0&0&0&1&1&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&1&1&1\\
0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0\\
\hline
1&1&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&1\\
0&0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&1\\
0&0&0&1&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0\\
1&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0\\
1&0&0&0&0&0&0&1&0&0&0&0&1&0&1&1&1&0&0&0&0&0&0&1&0&0&0&0&0&1&0&1\\
1&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1\\
\hline
1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&1&1&0&0&0&0&0&1&0&1&1&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&1&1&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&1&1&0&0&1&0&0&0&0&1&1&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&1&0&0&0&0&0&0&1&1&1&1&0&0&0&0&0&1&1&0&0&0&0&0&1&1&0\\
0&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1\\
\hline
1&0&0&0&0&0&0&0&0&1&0&0&0&1&1&0&0&1&1&0&0&0&0&0&0&1&1&0&0&0&1&1\\
0&1&0&0&0&0&0&0&0&1&1&0&0&1&0&1&0&1&0&1&0&0&0&0&0&0&1&0&0&0&1&0\\
0&0&1&0&0&0&0&0&0&0&1&1&0&0&1&0&1&0&1&0&0&0&0&0&1&0&0&1&0&0&0&1\\
0&0&0&1&0&0&0&0&1&0&0&1&1&0&0&1&0&0&0&1&0&1&0&0&0&1&0&0&1&0&0&0\\
0&0&0&0&1&0&0&0&0&1&0&0&1&1&0&0&1&0&0&0&1&0&1&0&0&0&1&0&0&1&0&0\\
0&0&0&0&0&1&0&0&1&0&1&0&0&1&1&0&0&1&0&0&0&0&1&0&1&1&0&0&1&0&0&1\\
0&0&0&0&0&0&1&0&1&0&0&1&0&1&0&1&0&1&0&1&0&0&0&0&1&0&1&0&0&0&1&0\\
0&0&0&0&0&0&0&1&1&0&0&0&1&1&0&0&1&0&0&0&0&1&1&0&0&0&0&1&0&1&1&0
\end{array}\right) \tag{11}$$

From this representation we obtain that $\mathsf{DXC}(\mathcal{M}_{\mathsf{RR}}) = 199$ and using a linear straight-line program corresponding to $\mathcal{M}_{\mathsf{RR}}$ with automatic tools

proposed in [2] we have found an implementation of this matrix (given in Table 2) which require only 80 bitwise XORs.

#	Operation	#	Operation	#	Operation	#	Operation
1	$t_0 = x_{14} \oplus x_{31}$	21	$y_{28} = y_{11} \oplus t_{19}$	41	$t_{40} = x_{10} \oplus t_{37}$	61	$y_{30} = t_{58} \oplus t_{59}$
2	$t_1 = x_{13} \oplus x_{30}$	22	$t_{21} = x_5 \oplus x_{21}$	42	$y_{18} = t_{39} \oplus t_{40}$	62	$t_{61} = x_{14} \oplus t_{57}$
3	$t_2 = x_8 \oplus x_{15}$	23	$y_{12} = t_4 \oplus t_{21}$	43	$t_{42} = x_{24} \oplus t_0$	63	$y_{29} = y_{12} \oplus t_{61}$
4	$t_3 = x_9 \oplus x_{26}$	24	$y_{21} = t_{19} \oplus t_{21}$	44	$y_9 = t_{39} \oplus t_{42}$	64	$t_{63} = t_0 \oplus t_{12}$
5	$t_4 = x_{10} \oplus x_{27}$	25	$t_{24} = x_{21} \oplus t_5$	45	$y_{26} = t_{35} \oplus y_9$	65	$y_{13} = t_{58} \oplus t_{63}$
6	$t_5 = x_{11} \oplus x_{28}$	26	$y_5 = y_{21} \oplus t_{24}$	46	$t_{45} = x_{18} \oplus t_{10}$	66	$t_{65} = t_1 \oplus t_{40}$
7	$t_6 = x_{17} \oplus x_{24}$	27	$t_{26} = x_{12} \oplus t_{24}$	47	$y_2 = y_{18} \oplus t_{45}$	67	$y_{25} = t_{55} \oplus t_{65}$
8	$t_7 = x_{12} \oplus x_{29}$	28	$y_{20} = t_{16} \oplus t_{26}$	48	$t_{47} = x_1 \oplus x_{17}$	68	$t_{67} = x_{17} \oplus y_{17}$
9	$t_8 = x_{16} \oplus t_7$	29	$t_{28} = x_{20} \oplus t_4$	49	$t_{48} = x_9 \oplus t_{45}$	69	$y_1 = t_{42} \oplus t_{67}$
10	$t_9 = x_0 \oplus t_0$	30	$y_4 = y_{20} \oplus t_{28}$	50	$y_{17} = t_{47} \oplus t_{48}$	70	$t_{69} = x_{15} \oplus t_{53}$
11	$t_{10} = x_{25} \oplus t_2$	31	$t_{30} = x_3 \oplus x_{19}$	51	$t_{50} = x_{15} \oplus t_{33}$	71	$y_{14} = t_7 \oplus t_{69}$
12	$t_{11} = x_{23} \oplus t_1$	32	$y_{10} = t_{10} \oplus t_{30}$	52	$y_{16} = x_0 \oplus t_{50}$	72	$t_{71} = t_9 \oplus t_{59}$
13	$t_{12} = x_{16} \oplus t_9$	33	$y_{27} = t_{26} \oplus y_{10}$	53	$t_{52} = x_6 \oplus x_{22}$	73	$y_{24} = t_{48} \oplus t_{71}$
14	$y_{15} = t_1 \oplus t_{12}$	34	$t_{33} = x_8 \oplus t_6$	54	$t_{53} = t_{11} \oplus t_{14}$	74	$y_{31} = y_{14} \oplus t_{71}$
15	$t_{14} = x_7 \oplus y_{15}$	35	$y_0 = t_{12} \oplus t_{33}$	55	$y_{23} = x_0 \oplus t_{53}$	75	$t_{74} = y_{16} \oplus t_{52}$
16	$y_7 = t_9 \oplus t_{14}$	36	$t_{35} = x_{11} \oplus t_{28}$	56	$t_{55} = x_{15} \oplus t_{47}$	76	$t_{75} = t_{63} \oplus t_{74}$
17	$t_{16} = x_4 \oplus x_{20}$	37	$y_{19} = t_30 \oplus t_{35}$	57	$y_8 = t_{12} \oplus t_{55}$	77	$y_{22} = t_{61} \oplus t_{75}$
18	$y_{11} = t_3 \oplus t_{16}$	38	$t_{37} = x_{19} \oplus t_3$	58	$t_{57} = t_{11} \oplus t_{33}$	78	$t_{77} = t_{18} \oplus t_{52}$
19	$t_{18} = x_{22} \oplus t_8$	39	$y_3 = y_{19} \oplus t_{37}$	59	$t_{58} = t_5 \oplus t_{52}$	79	$t_{78} = t_{11} \oplus t_{77}$
20	$t_{19} = x_{13} \oplus t_{18}$	40	$t_{39} = x_2 \oplus x_{18}$	60	$t_{59} = t_1 \oplus t_{50}$	80	$y_6 = x_{31} \oplus t_{78}$

Table 2: An implementation of $\mathcal{M}_{\mathsf{RR}}$ with 80 XORs.

6 Comparing our MDS matrices with the state-of-the-art

In Table 3 we compare our matrices with others by different methods in the public literature. We can see that the the implementations cost in hardware of the linear transformations obtained by our approach is comparable with state-of-the-art. Moreover, we can obtain a trade offs between software and hardware implementations for some matrices produced by our techniques.

Matrix	Involutory	Almost involutory	SLP
$\mathcal{M}_{\mathsf{AES}}$ [17]	✗	✗	97
$\mathcal{M}_{\mathsf{KLSW}}$ [14]	✓	✗	84
$\mathcal{M}_{\mathsf{SSCZL}}$ [19]	✓	✓	80
$\mathcal{M}_{\mathsf{SG}}$ [5]	✗	✗	78
$\mathcal{M}_{\mathsf{MM}}$ [18]	✗	✓	83
$\mathcal{M}_{\mathsf{RR}}$ [this work]	✓	✓	80

Table 3: A comparison with the sate-of-the-art.

7 Conclusion and Future Work

In this work we have presented some new schemes based on the well-known Feistel, Misty and Lai-Massey structures for constructing MDS matrices of size $n = 2k, k = 2$ over field $GF(2^8)$. Combining these structures we provide involutory and almost involutory MDS matrices which can be implemented efficiently. We have found some matrices having the MDS property which are very actractive for the so-called lightweight schemes. Also, we have obtained 6x6 MDS matrix combining these structures and we leave such matrices for future work. In the future, we aim to further optimise the search for constructing MDS matrices of size $2k, k \geq 4$ using our aproach. Our results can be generalized for any finite field, although it is necessary to say that we can not construct MDS matrices over $GF(2^2)$ with the structures presented in this work.

Acknowledgements. The authors are very grateful to the anonymous reviewers of CTCrypt'2020 for their useful comments and valuable observations, which helped to improve the final version of this article.

References

[1] Adnan B. Mustafa C. and Mehmet O. Feistel Like Construction of Involutory Binary Matrices With High Branch Number. Cryptology ePrint Archive, Report 2016/751.

[2] Boyar J., Matthews P., Peralta R.: Logic minimization techniques with applications to cryptology. J. Cryptology, 26(2):280–312, 2013.

[3] C. Shannon. Communication theory of secrecy systems. Bell System Technical Journal, 28(4), 1949

[4] Dmitry Burov, Boris, Pogorelov. The influence of linear mapping reducibility on choice of round constants. CTCcrypt 216

[5] Duval S. and Leurent G.: MDS Matrices with Lightweight Circuits. In FSE, volume 2018, pages 48-78. Springer, 2018.

[6] Glukhov M. M., Elizarov V. P., Nechaev A. A. Algebra. LAN. 2015. 595 p. (In Russian)

[7] Hong X., Lin T. Xuejia L. On the recursive construction of MDS matrices for ligtweight Cryptography

[8] H. M. Heys, and S. E. Tavares,The Design of Substitution-Permutation Networks Resistatnt to Diferential and Linear Cryptanalysis, Proceedings of 2nd ACM Conference on Computer and Communications Security, Fairfax, Virginia, pp. 148-155, 1994.

[9] H. M. Heys, and S. E. Tavares,The Design of Product Ciphers Resistatnt to Diferential and Linear Crypt-analysis, Journal Of Cryptography, Vol. 9, No. 1, pp. 1-19, 1996

[10] H. M. Heys, and S. E. Tavares,Avalanche Characteristics of Substitution-Permutation Encryption Networks.

[11] Gupta, K.C., Ray, I.G.: On Constructions of MDS Matrices from Companion Matrices for Lightweight Cryptography. In: Cuzzocrea, A., Kittl, C., Simos, D.E., Weippl, E., Xu, L. (eds.) CD-ARES Workshops 2013. LNCS, vol. 8128, pp. 29–43. Springer, Heidelberg (2013).

[12] Junod P. and Vaudenay S.: Perfect Diffusion Primitives for Block Ciphers Building Efficient MDS Matrices, Selected Areas in Cryptography 2004: Waterloo, Canada, August 9-10,2004. Revisited papers,LNCS. Springer-Verlag. Journal Information Security Practice and Experience, Springer pp 552-563. 2014.

[13] Jian G., Thomas P. and Axel P. The PHOTON Family of Lightweight Hash Functions. Cryptology ePrint Archive, Report 2011/609.

[14] Kranz H., Leander G., Stoffelen K., and Wiemer F. Shorter Linear Straight-Line Programs for MDS Matrices. In FSE, volume 2017, pages 188-211. Springer, 2017.

[15] Mahdi S. and Mohsen M. Construction of Lightweight MDS Matrices from Generalized Feistel Structures. Cryptology ePrint Archive, Report 2018/1072.

[16] Mahdi S., Mohammad D., Hamid M. and Behnaz O. On construction of involutory MDS matrices from Vandermonde Matrices in $GF(2^q)$. Springer. Published November 2011.

[17] NIST. Advanced Encryption Standard. Federal Information Processing Standard (FIPS) 197,November 2001.

[18] Sajadieh M., and Mousavi M.: Construction of Lightweight MDS Matrices from Generalized Feistel Structures. Cryptology ePrint Archive, Report 2018/1072.

[19] Shun Li1, Siwei Sun1, Chaoyun Li Zihao Wei1 and Lei Hu1: Constructing Low-latency Involutory MDS Matrices with Lightweight Circuits. In FSE. Springer, 2019.

Constructing permutations, involutions and orthomorphisms with almost optimal cryptographic parameters

Reynier Antonio de la Cruz Jiménez

Institute of Cryptography, Havana University, Cuba.
djr.antonio537@gmail.com

Abstract

Nonlinear bijective transformations (the so-called S-Boxes) are crucial components in the design of many symmetric ciphers. To construct permutations having cryptographic properties close to the optimal ones is not a trivial task. In this work we propose a new construction based on the well-known Lai-Massey structure for generating permutations of dimension $n = 2k, k \geq 2$. The main cores of our constructions are: the inversion in GF(2^k), an arbitrary k-bit non-bijective function (which has no preimage for 0) and any k-bit permutation. Combining these components with the finite field multiplication, we provide new 8-bit permutations with high values of its basic cryptographic parameters. Also, we show that our approach can be used for constructing involutions and orthomorphisms that have strong cryptographic properties.

Keywords: S-Box, permutation, involution, orthomorphism, Boolean function, nonlinear multiplications.

1 Introduction

Modern block ciphers are often iterations of several rounds. Each round (which must depend on the key) consists of a confusion layer and a diffusion layer. The confusion layers are usually formed by local nonlinear mappings (S-Boxes) while the diffusion layers are formed by global linear mappings mixing the output of the different S-Boxes. Block ciphers can be built using a well-known structure such as a Feistel network (and its variants) (see, e.g. [1]), a Substitution-Permutation network (SPN) [39], or a Lai-Massey structure [47]. Cryptographic properties of S-Boxes deal with the application of several logical attacks on ciphers, namely linear attack [24], the differential attack [24], the higher order differential attack [27] and algebraic attack [8] (which is not yet efficient but represents some threat and should keep in mind by designers of next generation of block ciphers). For this reason S-Boxes must

satisfy various criteria for providing high level of protection against such attacks.

Besides the linear, differential and algebraic attacks, today the most prominent attacks on the cryptographic algorithms are based on supervision of physical processes in cryptographic device. In literature, this kind of attack has received the name of side-channel attacks (SCAs). Examples of such attacks are: Simple Power Analysis (SPA) [25], Differential Power Analysis (DPA) [25], Timing Analysis (TA) [26] , Correlation Power Analysis (CPA) [5], Mutual Information Attack (MIA)[12]. S-Boxes represent the most vulnerable part in an implementation when considering side-channel adversary and it is not a trivial task to construct S-Boxes having good resistive properties both towards classical cryptanalysis as well side-channel attacks.

The known methods for constructing S-Boxes can be divided into four main classes: algebraic constructions, pseudo-random generation, heuristic techniques and constructions from small to large S-Boxes. Each approach has its advantages and disadvantages respectively. In this article we propose (using the last approach) a new construction based on the Lai-Massey structure for generating ordinary permutations, involutions and orthomorphisms with strong cryptographic properties and therefore study the resilience of such construction against side-channel attacks in terms of its masking complexity.

This article is structured as follows: In Section 1 we give the basic definitions. In Section 2, we present our design criteria. In section 3 we present a new class of permutations which can be used for constructing ordinary S-Boxes, involutions and orthomorphisms with high values of its basic cryptographic parameters. In this section, we also derive some properties of the suggested class of permutations. In Section 4 we give some examples of 8-bit S-Boxes constructed by our approach. The masking complexity of our S-Boxes is bounded in Section 5. Our work is concluded in Section 6.

2 Basic definitions and notations

Let V_n be n-dimensional vector space over the field GF(2) and furthermore, we denote $V_n^* = V_n \setminus \{0\}$. By $S(V_n)$ we denote the symmetric group on set of 2^n elements. The finite field of size 2^n is denoted by $\mathrm{GF}(2^n)$, where $\mathrm{GF}(2^n)$=$\mathrm{GF}(2)[\xi]/g(\xi)$, for some irreducible polynomial $g(\xi)$ of degree n. We use the notation $\mathbb{Z}/2^n$ for the ring of the integers modulo 2^n. The set of all binary bijective linear maps of size $n \times n$ is denoted by $\mathsf{GL}_n(\mathrm{GF}(2))$. Given a natural number l, throughout the article we shall use the following operations

and notations:

$a\|\|b$	- concatenation of the vectors a, b of V_l, i.e., a vector from V_{2l} ;
0	- the null vector of V_l;
$\oplus$	- bitwise eXclusive-OR. Addition in $\mathrm{GF}(2^l)$;
$\langle a, b\rangle$	- the scalar product of vectors $a = (a_{l-1}, \ldots, a_0), b = (b_{l-1}, \ldots, b_0)$ of V_l and is equal to $\langle a, b\rangle = a_{l-1}b_{l-1} \oplus \ldots \oplus a_0 b_0$;
$w_H(a)$	- the Hamming weight of a binary vector $a \in V_l$, i.e., the number of its nonzero coordinates;
$\otimes$	- finite field multiplication ;
$\Lambda \circ \Psi$	- a composition of mappings, where Ψ is the first to operate;
Ψ^{-1}	- the inverse transformation to some bijective mapping Ψ
$\chi(\Phi_1, \Phi_2)$	- the Hamming distance between $\Phi_1, \Phi_2 \in S(V_l)$;
$\mathsf{ord}(a)$	- the multiplicative order of the element $a \in \mathrm{GF}(2^l)$.

There are bijective mappings between $\mathbb{Z}/2^n, V_n$ and $\mathrm{GF}(2^n)$ defined by the correspondences:

$$\left[a_{n-1} \cdot 2^{n-1} + \ldots + a_0\right] \leftrightarrow (a_{n-1}, \ldots, a_0) \leftrightarrow \left[a_{n-1} \otimes \xi^{n-1} \oplus \ldots \oplus a_0\right].$$

Using these mapping in what follows we make no difference between vectors of V_n and the corresponding elements in and $\mathbb{Z}/2^n$ and $\mathrm{GF}(2^n)$.

We define the indicator function as follows

$$\mathsf{Ind}(x, y) = \begin{cases} 1, & \text{если} \quad x = y; \\ \\ 0, & \text{если} \quad x \neq y. \end{cases}$$

The function $\mathsf{Ind}(x, y)$ is a Boolean function, hence by $\overline{\mathsf{Ind}}(x, y)$ we denote its logical negation, i.e.,

$$\overline{\mathsf{Ind}}(x, y) \begin{cases} 0, & \text{если} \quad x = y; \\ \\ 1, & \text{если} \quad x \neq y. \end{cases}$$

Now, we introduce some basic concepts needed to describe and analyze S-Boxes with respect to linear, differential, algebraic attacks. For this purpose, we consider an n-bit S-Box Φ as a vector of Boolean functions:

$$\Phi = (f_{n-1}, \ldots, f_0), f_i : V_n \to V_1, i = 0, 1, \ldots n-1. \tag{1}$$

For some fixed $i = 0, 1, \ldots, n-1$, every Boolean function f_i can be written as a sum over V_1 of distinct t-order products of its arguments, $0 \leq t \leq n-1$; this is called the algebraic normal form of f_i. Functions f_i are called coordinate

Boolean functions of the S-Box Φ and it is well known that most of the desirable cryptographic properties of Φ can be defined in terms of their linear combinations (also-called S-Box component Boolean functions).

Definition 1. *For $a, b \in V_n$ the Walsh transform $\mathcal{W}_\Phi(a,b)$ of an n-bit S-Box Φ is defined as*

$$\mathcal{W}_\Phi(a,b) = \sum_{x \in V_n} (-1)^{\langle b, \Phi(x) \rangle \oplus \langle a, x \rangle}. \tag{2}$$

Definition 2. *The nonlinearity of an n-bit S-Box Φ, denoted by $\mathcal{NL}(\Phi)$, is defined as*

$$\mathcal{NL}(\Phi) = 2^{n-1} - \frac{1}{2} \cdot \max_{b \neq 0, a \in \mathrm{GF}(2^n)} |\mathcal{W}_\Phi(a,b)|. \tag{3}$$

From a cryptographic point of view S-Boxes with small values of Walsh coefficients offer better resistance against linear attacks.

Definition 3. *The differential uniformity (also called δ-uniformity) of an n-bit S-Box Φ, denoted by δ_Φ, is defined as*

$$\delta_\Phi = \max_{a \neq 0, b \in \mathrm{GF}(2^n)} \Delta_\Phi(a,b), \tag{4}$$

where $\Delta_\Phi(a,b) = \#\{x \in \mathrm{GF}(2^n) | \, \Phi(x \oplus a) \oplus \Phi(x) = b\} = \sum_{x \in V_n} \mathsf{Ind}(\Phi(x \oplus a) \oplus \Phi(x), b)$.

The resistance offered by an S-Box against differential attacks is related by the highest value of δ, for this reason S-Boxes must have a small value of δ-uniformity for a sufficient level of protection against this type of attacks.

Definition 4. *The minimum (maximum) algebraic degree of an S-Box Φ, denoted by $\deg(\Phi)$, is the minimum (maximum) among all maximum numbers of variables of the terms in the algebraic normal form of $(\langle a, \Phi(x) \rangle)$ for all possible values x and $a \neq 0$:*

$$\deg(\Phi)_{(\mathsf{min})} = \min_{a \neq 0 \in V_n} \deg\big(\langle a, \Phi(x) \rangle\big), \tag{5}$$

$$\deg(\Phi)_{(\mathsf{max})} = \max_{a \neq 0 \in V_n} \deg\big(\langle a, \Phi(x) \rangle\big). \tag{6}$$

It is well-known that the minimum (maximum) algebraic degree of any permutation $\Phi \in S(V_n)$ is upper bounded by $n-1$. In general, S-Boxes should have high minimum (maximum) degree because S-Boxes with low degree are susceptible to algebraic attack, higher-order differential, interpolation, cube attacks etc.

Definition 5. *The univariate polynomial representation of an n-bit S-Box* Φ *over* $\mathrm{GF}(2^n)$*, is defined in a unique fashion as*

$$\Phi(X) = \sum_{i=0}^{2^n-1} \nu_i X^i, \nu_i \in \mathrm{GF}(2^n), \tag{7}$$

where coefficients $\nu_i, i = 0, \ldots, 2^n - 1$ can be obtained from the n-bit S-Box Φ by applying Lagrange's Interpolation theorem (see, for example, [6]).

Definition 6. *Let* U *be a non-empty subset of* $\mathrm{GF}(2^{2n})$*, then the annihilating set of* U *is defined as*

$$\{p \in \mathrm{GF}(2)[z_1, \ldots, z_{2n}] \big| p(U) = 0\}.$$

Definition 7. *The algebraic immunity of* U *is defined as*

$$\mathcal{AI}(U) = \min\left\{ \deg p \,\middle|\, 0 \neq p \in \mathrm{GF}(2)[z_1, \ldots, z_{2n}], p(U) = 0 \right\}. \tag{8}$$

Definition 8. *The graph algebraic immunity of n-bit S-Box* Φ*, denoted by* $\mathcal{AI}_{gr}(\Phi)$*, is defined as*

$$\mathcal{AI}_{gr}(\Phi) = \min\left\{ \deg p \,\middle|\, 0 \neq p \in \mathrm{GF}(2)[z_1, \ldots, z_{2n}], p(gr(\Phi)) = 0 \right\}, \tag{9}$$

where $gr(\Phi) = \{(x, \Phi(x)) | x \in \mathrm{GF}(2^n)\} \subseteq \mathrm{GF}(2^{2n})$.

Thus we focus on the graph algebraic immunity of S-Box Φ and also on the parameter $r_{\Phi}^{(AI_{gr}(\Phi))}$ referred to as the number of all the independent equations in input and output values of the S-Box Φ, i.e., equations of the form $p(x, \Phi(x)) = 0$, for all $x \in \mathrm{GF}(2^n)$.

Definition 9. *An element* $a \in V_n$ *is called a fixed point of an n-bit S-Box* Φ *if* $\Phi(a) = a$.

Definition 10. *Two n-bit S-Boxes* Φ_1 *and* Φ_2 *are linear (resp. affine) equivalent if there exist linear (resp. affine) mappings* A_1, A_2*, such that* $\Phi_2 = A_2 \circ \Phi_1 \circ A_1$.

It is well-known that the following cryptographic parameters: δ-uniformity, nonlinearity and minimum (maximum) algebraic degree remains invariant under linear (resp. affine) equivalence.

3 General S-Box Design Criteria

Our goal, is to find permutations constructed by smaller ones that satisfy the following criteria (which in what follows are called almost optimal):

1. Maximum value of minimum degree;
2. Maximum graph algebraic immunity with the minimum number of equations;
3. Minimum value of δ-uniformity limited by parameter listed above;
4. Maximum value of nonlinearity limited by parameter listed above.

For example,when $n = 8$ an almost optimal nonlinear bijective transformation Φ should satisfy the following

$$\mathsf{Set\ of\ cryptographic\ criteria\ for\ 8-bit\ permutations:}$$

- $\deg(\Phi) = 7$;
- $\delta_\Phi \leq 8$;
- $\mathcal{AI}_{gr}(\Phi) = 3$ with $r_\Phi^{(3)} = 441$;
- $\mathcal{NL}(\Phi) \geq 100$.

Our design criteria are basically the same as those included in the target set of criteria for the Gradient descent method [22]. However, we concentrate on generating 8-bit S-Boxes with almost optimal cryptographic parameters having good resistive properties both towards classical cryptanalysis as well side-channel attacks with some given level of masking.

4 Construction of permutations, involutions and orthomorphisms

Now, we present an special algorithmic-algebraic scheme based on the well-known Lai-Massey structure which can be used not only for constructing permutations, but also involutions and orthomorphisms having almost optimal cryptographic properties.

Let be $n = 2k$ a natural number, where $k \geq 2$. Choosing:

– Finite field inversion function $\mathcal{I} = x^{-1}$ over $\mathrm{GF}(2^k)$;

– Non-bijective k-bit function ψ which has no preimage for 0;

– Arbitrary permutation $h \in S(V_k)$;

– Arbitrary binary matrices $\mathcal{L}_i \in \mathsf{GL}_{2k}(\mathrm{GF}(2)), i = 1, 2$.

We construct the following $2k$-bit class of permutations π from V_{2k} to V_{2k} as follows

Construction of π

For the input value $(l\|r) \in V_{2k}$ we define the corresponding output value $\pi(l\|r) = (l_1\|r_1)$ as a result of the following computations:

$$(l_1\|r_1) := \mathcal{L}_1(l\|r);$$
$$(l_1\|r_1) := (\mathcal{I}(l_1) \otimes \psi(l_1 \otimes r_1))\big\|h(r_1 \otimes \psi(l_1 \otimes r_1));$$
$$(l_1\|r_1) := \mathcal{L}_2(l_1\|r_1).$$

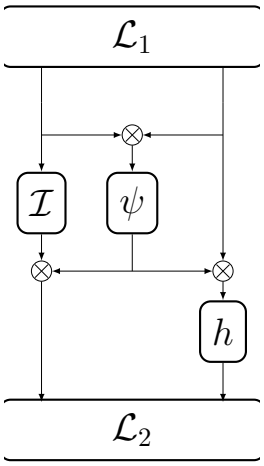


Fig. 1: High level structure of the S-Box π.

Notice, that the finite field multiplication $\otimes$ in the above construction correspond to multiplication operation in $\mathrm{GF}(2^k)$.The binary matrices $\mathcal{L}_1$ and $\mathcal{L}_2$ were inserted in such a way as to break the cycle structure of π and also to eliminate existence of fixed points. If define $\hat{\pi}$ as $\mathcal{L}_2^{-1} \circ \pi \circ \mathcal{L}_1^{-1}$ we can see that $\hat{\pi}$ share similarities with 1-round of Lai-Massey structure replacing in the latter the XORs by finite field multiplications. The non-bijective k-bit function ψ (which has no preimage for 0) was chosen in such a way to make all the whole structure invertible. Moreover, from the next construction:

- $\hat{\pi}^{-1}(l_1\|r_1) = l\|r$ where $l = h^{-1}(l_1) \otimes \mathcal{I}\big(\psi(h^{-1}(l_1) \otimes \mathcal{I}(r_1))\big), r = \mathcal{I}\big(r_1 \otimes \mathcal{I}\big(\psi(h^{-1}(l_1) \otimes \mathcal{I}(r_1))\big)\big)$;

we can easy derive the bijectivity of the π which is a necessary design criteria for SPN ciphers and quite useful for Feistel and Lai-Massey ciphers.

In more detail, the nonlinear bijective transformation $\hat{\pi}$ can be written as follows

$$\hat{\pi}(l\|r) = \begin{cases} 0, & \text{if} \quad l = r = 0; \\ 0\big\|h(r \otimes \psi(0)), & \text{if} \quad l = 0 \text{ and } r \neq 0; \\ \big(\mathcal{I}(l) \otimes \psi(0)\big)\big\|0, & \text{if} \quad l \neq 0 \text{ and } r = 0; \\ (\mathcal{I}(l) \otimes \psi(l \otimes r))\big\|h(r \otimes \psi(l \otimes r)), & \text{if} \quad l \neq 0 \text{ and } r \neq 0. \end{cases} \tag{10}$$

In what follows, for simplicity we restricted ourselves to the case when $h = \mathcal{I}$. The next well-known result is useful when studying some properties of the suggested class of permutations.

Lemma 1. *[3, 28] For any $b \in V_k \setminus \{0\}, a \in V_k$, the following inequality holds*

$$\left|\sum_{x\in V_k}(-1)^{\langle b,\mathcal{I}(x)\rangle\oplus\langle a,x\rangle}\right| \leq \lfloor 2^{\frac{k}{2}+1}\rfloor. \tag{11}$$

Proposition 1. *Let $h = \mathcal{I}$ and ψ— non-bijective k-bit function ψ which has no preimage for 0. Then*

$$\mathcal{NL}(\pi) \geq 2^k - \lfloor 2^{\frac{k}{2}+1}\rfloor - 1. \tag{12}$$

Proof. It is not difficult to see that permutations $\pi, \hat{\pi}$ are linear/affine equivalent, hence $\mathcal{NL}(\pi) = \mathcal{NL}(\hat{\pi})$. Let us calculate the Walsh transform of the nonlinear bijective transformation $\hat{\pi}$

$$\begin{aligned}\mathcal{W}_{\hat{\pi}}(a_1\|a_2, b_1\|b_2) &= \sum_{l\|r\in V_{2k}}(-1)^{\langle b_1\|b_2,\hat{\pi}(l\|r)\rangle\oplus\langle a_1\|a_2,l\|r\rangle}\\ &= -1+\sum_{r\in V_k}(-1)^{\langle b_2,\mathcal{I}(r\otimes\psi(0))\rangle\oplus\langle a_2,r\rangle}+\sum_{l\in V_k}(-1)^{\langle b_1,\mathcal{I}(l)\otimes\psi(0)\rangle\oplus\langle a_1,l\rangle}\\ &\quad+\sum_{l\in V_k^*}\sum_{r\in V_k^*}(-1)^{\langle b_1,\mathcal{I}(l)\otimes\psi(l\otimes r)\rangle\oplus\langle b_2,\mathcal{I}(r\otimes\psi(l\otimes r))\rangle\oplus\langle a_1,l\rangle\oplus\langle a_2,r\rangle}.\end{aligned}$$

Let us now estimate the Walsh transform $|\mathcal{W}_{\hat{\pi}}(a_1\|a_2, b_1\|b_2)|$. Directly from lemma 1 we can derive the following inequalities

- $\left|\sum_{r\in V_k}(-1)^{\langle b_2,\mathcal{I}(r\otimes\psi(0))\rangle\oplus\langle a_2,r\rangle}\right| \leq \lfloor 2^{\frac{k}{2}+1}\rfloor$;
- $\left|\sum_{l\in V_k}(-1)^{\langle b_1,\mathcal{I}(l)\otimes\psi(0)\rangle\oplus\langle a_1,l\rangle}\right| \leq \lfloor 2^{\frac{k}{2}+1}\rfloor$.

The module of sum $\sum_{l\in V_k^*}\sum_{r\in V_k^*}(-1)^{\langle b_1,\mathcal{I}(l)\otimes\psi(l\otimes r)\rangle\oplus\langle b_2,\mathcal{I}(r\otimes\psi(l\otimes r))\rangle\oplus\langle a_1,l\rangle\oplus\langle a_2,r\rangle}$ is upper bounded by $(2^k-1)\cdot(2^k-1)$, i.e.,

$$\left|\sum_{l\in V_k^*}\sum_{r\in V_k^*}(-1)^{\langle b_1,\mathcal{I}(l)\otimes\psi(l\otimes r)\rangle\oplus\langle b_2,\mathcal{I}(r\otimes\psi(l\otimes r))\rangle\oplus\langle a_1,l\rangle\oplus\langle a_2,r\rangle}\right| \leq (2^k-1)\cdot(2^k-1).$$

Hence,

$$|\mathcal{W}_{\hat{\pi}}(a_1\|a_2, b_1\|b_2)| \leq 2^{2k} - 2^{k+1} + 2\cdot\lfloor 2^{\frac{k}{2}+1}\rfloor + 2. \tag{13}$$

Thus, from (13) we obtain

$$\mathcal{NL}(\pi) = 2^{2k-1} - \frac{1}{2} \cdot \max_{b \neq 0, a \in V_{2k}} |\mathcal{W}_\pi(a_1 \| a_2, b_1 \| b_2)| \geq 2^k - \lfloor 2^{\frac{k}{2}+1} \rfloor - 1.$$

□

Proposition 2. *Let $\psi : V_k \to V_k$ be an arbitrary non-bijective function which has no preimage for 0. Then for the permutation π, when $h = \mathcal{I}$ the following inequalities holds*

$$k - 1 \leq \deg(\pi)_{(\mathsf{max})} \leq 2k - 1. \tag{14}$$

Proof. It is not difficult to see that permutations $\pi, \hat{\pi}$ are linear/affine equivalent, hence, $\deg(\pi)_{(\mathsf{max})} = \deg(\hat{\pi})_{(\mathsf{max})}$. From definition 4 we have

$$\deg(\hat{\pi})_{(\mathsf{max})} = \max_{(a_1 \| a_2) \neq 0 \in V_{2k}} \deg\big(\langle a_1 \| a_2, \hat{\pi}(l \| r) \rangle\big).$$

According to the equality (10), the function $\langle a_1 \| a_2, \hat{\pi}(l \| r) \rangle$ can be decomposed as

$$\begin{aligned} \langle a_1 \| a_2, \hat{\pi}(l \| r) \rangle \;=\; & \langle a_2, \mathcal{I}(r \otimes \psi(0)) \rangle \cdot \mathsf{Ind}(l, 0) \cdot \overline{\mathsf{Ind}}(r, 0) \\ & + \langle a_1, \mathcal{I}(l) \otimes \psi(0) \rangle \cdot \mathsf{Ind}(r, 0) \cdot \overline{\mathsf{Ind}}(l, 0) \\ & + \langle a_2, \mathcal{I}(r \otimes \psi(l \otimes r)) \rangle \cdot \overline{\mathsf{Ind}}(l, 0) \cdot \overline{\mathsf{Ind}}(r, 0) \\ & + \langle a_1, \mathcal{I}(l) \otimes \psi(l \otimes r) \rangle \cdot \overline{\mathsf{Ind}}(l, 0) \cdot \overline{\mathsf{Ind}}(r, 0). \end{aligned}$$

If $\deg \psi_{(\mathsf{max})} = 0$, then without loss of generality we can assume that for any $z \in V_k$, $\psi(z) = c$, where c is some nonzero element of $\mathrm{GF}(2^k)$. Now taking into account that $\deg \mathcal{I}_{(\mathsf{max})} = k - 1$ we obtain $\deg(\hat{\pi})_{(\mathsf{max})} = k - 1$. If $0 < \deg \psi_{(\mathsf{max})} \leq k$, then due to the fact that $\hat{\pi}$ is a permutation on V_{2k} we conclude that its maximum algebraic degree is upper bounded by $2k - 1$.

□

4.1 The Hamming distance between two instances of $\hat{\pi}$

In this section we are interested by the hamming distance between two instance of $\hat{\pi}$ having the following the non-bijective function ψ, $\hat{\psi}$ which are related as follows, $\psi(i) \neq \hat{\psi}(i)$, for some $i \in \{0, 1, \ldots, 2^k - 1\}$, $\psi(j) = \hat{\psi}(j)$, when $j \in \{0, 1, \ldots, 2^k - 1\} \setminus \{i\}$. In other words, the lookup-tables of ψ and $\hat{\psi}$ differs only in one position.

Proposition 3. *If the lookup-tables of non-bijective k-bit functions $\psi = \begin{pmatrix} \dots & i & \dots \\ \dots & \psi(i) & \dots \end{pmatrix}$ and $\hat{\psi} = \begin{pmatrix} \dots & i & \dots \\ \dots & \hat{\psi}(i) & \dots \end{pmatrix}$ differs from each other exactly in one output value, then for permutations $\hat{\pi}_\psi, \hat{\pi}_{\hat{\psi}}$ the following relation holds:*

$$\chi(\hat{\pi}_\psi, \hat{\pi}_{\hat{\psi}}) = \begin{cases} 2 \cdot (2^k - 1), & \text{if} \quad i = 0; \\ \\ 2^k - 1, & \text{if} \quad i \neq 0. \end{cases} \tag{15}$$

Proof. Consider the following possible cases:

1. $i = 0$. In this case we have the following relations $l \otimes r = 0$, which holds when $l = 0$ and $r \neq 0$ and when $r = 0$ and $l \neq 0$. It is easy to check that for $l = 0$ the next inequality, $\hat{\pi}_\psi(0\|r) \neq \hat{\pi}_{\hat{\psi}}(0\|r)$ holds for all $r \in V_k \setminus \{0\}$. Analogously, for $r = 0, l \neq 0 \in V_k$ the output $\hat{\pi}_\psi(l\|0) \neq \hat{\pi}_{\hat{\psi}}(l\|0)$. So we have exaclty $2 \cdot (2^k - 1)$ values in which the look-up-tables of $\hat{\pi}_\psi$ and $\hat{\pi}_{\hat{\psi}}$ will be differs;

2. $i \neq 0$. In this case for each fixed $l \in \mathrm{GF}(2^k)\backslash\{0\}$ there exist a unique $r \in \mathrm{GF}(2^k)\backslash\{0\}$ such that $l \otimes r = i$, therefore there are exactly $2^k - 1$ values of the form $(l\|r) \in V_{2k}$ such that $\hat{\pi}_\psi(l\|r) \neq \hat{\pi}_{\hat{\psi}}(l\|r)$.

Notice that we have exclude the case when $l = r = 0$ because in this situation we always have $\hat{\pi}_\psi(0) = \hat{\pi}_{\hat{\psi}}(0)$. So, we can conclude that $\chi(\hat{\pi}_\psi, \hat{\pi}_{\hat{\psi}}) \in \{2^k - 1, 2 \cdot (2^k - 1)\}$. □

4.2 Bounds on nonlinearity and δ-uniformity between two instances of $\hat{\pi}$

In this section, we study the nonlinearity and δ-uniformity parameter between two instances of $\hat{\pi}$ described in the previous section.

Proposition 4. *If the lookup-tables of non-bijective k-bit functions $\psi = \begin{pmatrix} \dots & i & \dots \\ \dots & \psi(i) & \dots \end{pmatrix}$ and $\hat{\psi} = \begin{pmatrix} \dots & i & \dots \\ \dots & \hat{\psi}(i) & \dots \end{pmatrix}$ differs from each other exactly in one output value, then for permutations $\hat{\pi}_\psi, \hat{\pi}_{\hat{\psi}}$ the following inequalities holds:*

1. $\mathcal{NL}(\hat{\pi}_\psi) - 2 \cdot \lfloor 2^{\frac{k}{2}+1} \rfloor \leq \mathcal{NL}(\hat{\pi}_{\hat{\psi}}) \leq \mathcal{NL}(\hat{\pi}_\psi) + 2 \cdot \lfloor 2^{\frac{k}{2}+1} \rfloor$, *when* $i = 0$;

2. $\mathcal{NL}(\hat{\pi}_\psi) - (2^k - 1) \leq \mathcal{NL}(\hat{\pi}_{\hat{\psi}}) \leq \mathcal{NL}(\hat{\pi}_\psi) + (2^k - 1)$, *when* $i \neq 0$.

Proof. Denote by $\rho(\mathcal{W}_{\hat{\pi}_\psi}, \mathcal{W}_{\hat{\pi}_{\hat{\psi}}}) = \mathcal{W}_{\hat{\pi}_\psi}(a_1\|a_2, b_1\|b_2) - \mathcal{W}_{\hat{\pi}_{\hat{\psi}}}(a_1\|a_2, b_1\|b_2)$ and $\rho(\mathcal{W}_{\hat{\pi}_{\hat{\psi}}}, \mathcal{W}_{\hat{\pi}_\psi}) = \mathcal{W}_{\hat{\pi}_{\hat{\psi}}}(a_1\|a_2, b_1\|b_2) - \mathcal{W}_{\hat{\pi}_\psi}(a_1\|a_2, b_1\|b_2)$ the difference of the Walsh transforms between functions $\hat{\pi}_\psi$ and $\hat{\pi}_{\hat{\psi}}$. To prove the proposition is sufficient to upper bound the following values

1. $\left|\rho(\mathcal{W}_{\hat{\pi}_\psi}, \mathcal{W}_{\pi_{\hat{\psi}}})\right|$;
2. $\left|\rho(\mathcal{W}_{\hat{\pi}_{\hat{\psi}}}, \mathcal{W}_{\pi_\psi})\right|$.

We shall prove the first item of the proposition. Let us calculate the Walsh transform of permutations $\hat{\pi}_\psi$ and $\hat{\pi}_{\hat{\psi}}$ respectively

$$\begin{aligned}
\mathcal{W}_{\hat{\pi}_\psi}(a_1\|a_2, b_1\|b_2) &= \sum_{l\|r\in V_{2k}} (-1)^{\langle b_1\|b_2, \hat{\pi}_\psi(l\|r)\rangle \oplus \langle a_1\|a_2, l\|r\rangle} \\
&= -1 + \sum_{r\in V_k} (-1)^{\langle b_2, \mathcal{I}(r\otimes\psi(0))\rangle \oplus \langle a_2, r\rangle} + \sum_{l\in V_k} (-1)^{\langle b_1, \mathcal{I}(l)\otimes\psi(0)\rangle \oplus \langle a_1, l\rangle} \\
&\quad + \sum_{l\in V_k^*}\sum_{r\in V_k^*} (-1)^{\langle b_1, \mathcal{I}(l)\otimes\psi(l\otimes r)\rangle \oplus \langle b_2, \mathcal{I}(r\otimes\psi(l\otimes r))\rangle \oplus \langle a_1, l\rangle \oplus \langle a_2, r\rangle},
\end{aligned}$$

From relations $\psi(0) \neq \hat{\psi}(0)$, and $\psi(j) = \hat{\psi}(j)$ for $j \in \{1, \ldots, 2^k - 1\}$ we obtain

$$\begin{aligned}
\mathcal{W}_{\hat{\pi}_{\hat{\psi}}}(a_1\|a_2, b_1\|b_2) &= \sum_{l\|r\in V_{2k}} (-1)^{\left\langle b_1\|b_2, \hat{\pi}_{\hat{\psi}}(l\|r)\right\rangle \oplus \langle a_1\|a_2, l\|r\rangle} \\
&= -1 + \sum_{r\in V_k} (-1)^{\left\langle b_2, \mathcal{I}(r\otimes\hat{\psi}(0))\right\rangle \oplus \langle a_2, r\rangle} + \sum_{l\in V_k} (-1)^{\left\langle b_1, \mathcal{I}(l)\otimes\hat{\psi}(0)\right\rangle \oplus \langle a_1, l\rangle} \\
&\quad + \sum_{l\in V_k^*}\sum_{r\in V_k^*} (-1)^{\langle b_1, \mathcal{I}(l)\otimes\psi(l\otimes r)\rangle \oplus \langle b_2, \mathcal{I}(r\otimes\psi(l\otimes r))\rangle \oplus \langle a_1, l\rangle \oplus \langle a_2, r\rangle}.
\end{aligned}$$

Then,

$$\begin{aligned}
\rho(\mathcal{W}_{\hat{\pi}_\psi}, \mathcal{W}_{\hat{\pi}_{\hat{\psi}}}) &= \Big(\sum_{r\in V_k} (-1)^{\langle b_2, \mathcal{I}(r\otimes\psi(0))\rangle \oplus \langle a_2, r\rangle} + \sum_{l\in V_k} (-1)^{\langle b_1, \mathcal{I}(l)\otimes\psi(0)\rangle \oplus \langle a_1, l\rangle}\Big) \\
&\quad - \Big(\sum_{r\in V_k} (-1)^{\left\langle b_2, \mathcal{I}(r\otimes\hat{\psi}(0))\right\rangle \oplus \langle a_2, r\rangle} + \sum_{l\in V_k} (-1)^{\left\langle b_1, \mathcal{I}(l)\otimes\hat{\psi}(0)\right\rangle \oplus \langle a_1, l\rangle}\Big),
\end{aligned}$$

$$\begin{aligned}
\rho(\mathcal{W}_{\hat{\pi}_{\hat{\psi}}}, \mathcal{W}_{\hat{\pi}_\psi}) &= \Big(\sum_{r\in V_k} (-1)^{\left\langle b_2, \mathcal{I}(r\otimes\hat{\psi}(0))\right\rangle \oplus \langle a_2, r\rangle} + \sum_{l\in V_k} (-1)^{\left\langle b_1, \mathcal{I}(l)\otimes\hat{\psi}(0)\right\rangle \oplus \langle a_1, l\rangle}\Big) \\
&\quad - \Big(\sum_{r\in V_k} (-1)^{\langle b_2, \mathcal{I}(r\otimes\psi(0))\rangle \oplus \langle a_2, r\rangle} + \sum_{l\in V_k} (-1)^{\langle b_1, \mathcal{I}(l)\otimes\psi(0)\rangle \oplus \langle a_1, l\rangle}\Big).
\end{aligned}$$

Using lemma 1 we derive the following bounds

$$\left|\rho(\mathcal{W}_{\hat{\pi}_\psi}, \mathcal{W}_{\hat{\pi}_{\hat{\psi}}})\right| \leq 4 \cdot \lfloor 2^{\frac{k}{2}+1} \rfloor, \tag{16}$$

$$\left|\rho(\mathcal{W}_{\hat{\pi}_{\hat{\psi}}}, \mathcal{W}_{\hat{\pi}_\psi})\right| \leq 4 \cdot \lfloor 2^{\frac{k}{2}+1} \rfloor. \tag{17}$$

From which we deduce that

$$\mathcal{NL}(\hat{\pi}_\psi) - 2 \cdot \lfloor 2^{\frac{k}{2}+1} \rfloor \leq \mathcal{NL}(\hat{\pi}_{\hat{\psi}}) \leq \mathcal{NL}(\hat{\pi}_\psi) + 2 \cdot \lfloor 2^{\frac{k}{2}+1} \rfloor.$$

Now, we prove the second item of the proposition. For each element $l \in \{1, 2, \ldots, 2^k - 1\}$ there exist a unique element $r \in \{1, 2, \ldots, 2^k - 1\}$ such that $l \otimes r = i$, where $i \in \{1, 2, \ldots, 2^k - 1\}$. Then, the Walsh transforms for permutations $\hat{\pi}_\psi$ and $\hat{\pi}_{\hat{\psi}}$ can be written as follows

$$\begin{aligned}
\mathcal{W}_{\hat{\pi}_\psi}(a_1\|a_2, b_1\|b_2) &= \sum_{l\|r \in V_{2k}} (-1)^{\langle b_1\|b_2, \hat{\pi}_\psi(l\|r)\rangle \oplus \langle a_1\|a_2, l\|r\rangle} \\
&= 1 + \sum_{r \in V_k^*} (-1)^{\langle b_2, \mathcal{I}(r \otimes \psi(0))\rangle \oplus \langle a_2, r\rangle} + \sum_{l \in V_k^*} (-1)^{\langle b_1, \mathcal{I}(l) \otimes \psi(0)\rangle \oplus \langle a_1, l\rangle} \\
&\quad + \sum_{l \in V_k^*} \sum_{r \in V_k^*} (-1)^{\langle b_1, \mathcal{I}(l) \otimes \psi(l \otimes r)\rangle \oplus \langle b_2, \mathcal{I}(r \otimes \psi(l \otimes r))\rangle \oplus \langle a_1, l\rangle \oplus \langle a_2, r\rangle}.
\end{aligned}$$

Denote $\hat{\mathcal{S}}(l, r) = \sum_{l \in V_k^*} \sum_{r \in V_k^*} (-1)^{\langle b_1, \mathcal{I}(l) \otimes \psi(l \otimes r)\rangle \oplus \langle b_2, \mathcal{I}(r \otimes \psi(l \otimes r))\rangle \oplus \langle a_1, l\rangle \oplus \langle a_2, r\rangle}$.

Then

$$\hat{\mathcal{S}}(l, r) = \sum_{l \in V_k^*} \mathcal{S}(l, r), \tag{18}$$

where $\mathcal{S}(l, r) = \sum_{r \in V_k^*} (-1)^{\langle b_1, \mathcal{I}(l) \otimes \psi(l \otimes r)\rangle \oplus \langle b_2, \mathcal{I}(r \otimes \psi(l \otimes r))\rangle \oplus \langle a_1, l\rangle \oplus \langle a_2, r\rangle}$.

For each fixed $l \in \{1, 2, \ldots, 2^k - 1\}$, the term $\mathcal{S}(l, r)$ can be rewritten as

$$\begin{aligned}
\mathcal{S}(l, r) &= \sum_{r \in V_k^* \setminus \{i \otimes l^{-1}\}} (-1)^{\langle b_1, \mathcal{I}(l) \otimes \psi(l \otimes r)\rangle \oplus \langle b_2, \mathcal{I}(r \otimes \psi(l \otimes r))\rangle \oplus \langle a_1, l\rangle \oplus \langle a_2, r\rangle} \\
&\quad + (-1)^{\langle b_1, \mathcal{I}(l) \otimes \psi(i)\rangle \oplus \left\langle b_2, \mathcal{I}((i \otimes l^{-1}) \otimes \psi(i))\right\rangle \oplus \langle a_1, l\rangle \oplus \left\langle a_2, i \otimes l^{-1}\right\rangle}.
\end{aligned}$$

Substituting $\mathcal{S}(l, r)$ in (18) we obtain

$$\begin{aligned}
\hat{\mathcal{S}}(l, r) &= \sum_{l \in V_k^*} \sum_{r \in V_k^* \setminus \{i \otimes l^{-1}\}} (-1)^{\langle b_1, \mathcal{I}(l) \otimes \psi(l \otimes r)\rangle \oplus \langle b_2, \mathcal{I}(r \otimes \psi(l \otimes r))\rangle \oplus \langle a_1, l\rangle \oplus \langle a_2, r\rangle} \\
&\quad + \sum_{l \in V_k^*} (-1)^{\langle b_1, \mathcal{I}(l) \otimes \psi(i)\rangle \oplus \left\langle b_2, \mathcal{I}((i \otimes l^{-1}) \otimes \psi(i))\right\rangle \oplus \langle a_1, l\rangle \oplus \left\langle a_2, (i \otimes l^{-1})\right\rangle}.
\end{aligned}$$

Hence,

$$\begin{aligned}\mathcal{W}_{\hat{\pi}_\psi}(a_1\|a_2,b_1\|b_2) &= \sum_{l\|r\in V_{2k}}(-1)^{\langle b_1\|b_2,\hat{\pi}_\psi(l\|r)\rangle\oplus\langle a_1\|a_2,l\|r\rangle}\\ &= 1+\sum_{r\in V_k^*}(-1)^{\langle b_2,\mathcal{I}(r\otimes\psi(0))\rangle\oplus\langle a_2,r\rangle}+\sum_{l\in V_k^*}(-1)^{\langle b_1,\mathcal{I}(l)\otimes\psi(0)\rangle\oplus\langle a_1,l\rangle}\\ &\quad+\sum_{l\in V_k^*}\sum_{r\in V_k^*\setminus\{i\otimes l^{-1}\}}(-1)^{\langle b_1,\mathcal{I}(l)\otimes\psi(l\otimes r)\rangle\oplus\langle b_2,\mathcal{I}(r\otimes\psi(l\otimes r))\rangle\oplus\langle a_1,l\rangle\oplus\langle a_2,r\rangle}\\ &\quad+\sum_{l\in V_k^*}(-1)^{\langle b_1,\mathcal{I}(l)\otimes\psi(i)\rangle\oplus\left\langle b_2,\mathcal{I}((i\otimes l^{-1})\otimes\psi(i))\right\rangle\oplus\langle a_1,l\rangle\oplus\left\langle a_2,(i\otimes l^{-1})\right\rangle}.\end{aligned}$$

Now, taking into account that $\psi(i)\neq\hat{\psi}(i)$, for some $i\in\{1,\ldots,2^k-1\}$, and $\psi(j)=\hat{\psi}(j)$ for any $j\in\{0,1,\ldots,2^k-1\}\setminus\{i\}$ we obtain

$$\begin{aligned}\mathcal{W}_{\hat{\pi}_{\hat{\psi}}}(a_1\|a_2,b_1\|b_2) &= \sum_{l\|r\in V_{2k}}(-1)^{\langle b_1\|b_2,\hat{\pi}_\psi(l\|r)\rangle\oplus\langle a_1\|a_2,l\|r\rangle}\\ &= 1+\sum_{r\in V_k^*}(-1)^{\langle b_2,\mathcal{I}(r\otimes\psi(0))\rangle\oplus\langle a_2,r\rangle}+\sum_{l\in V_k^*}(-1)^{\langle b_1,\mathcal{I}(l)\otimes\psi(0)\rangle\oplus\langle a_1,l\rangle}\\ &\quad+\sum_{l\in V_k^*}\sum_{r\in V_k^*\setminus\{i\otimes l^{-1}\}}(-1)^{\langle b_1,\mathcal{I}(l)\otimes\psi(l\otimes r)\rangle\oplus\langle b_2,\mathcal{I}(r\otimes\psi(l\otimes r))\rangle\oplus\langle a_1,l\rangle\oplus\langle a_2,r\rangle}\\ &\quad+\sum_{l\in V_k^*}(-1)^{\left\langle b_1,\mathcal{I}(l)\otimes\hat{\psi}(i)\right\rangle\oplus\left\langle b_2,\mathcal{I}((i\otimes l^{-1})\otimes\hat{\psi}(i))\right\rangle\oplus\langle a_1,l\rangle\oplus\left\langle a_2,(i\otimes l^{-1})\right\rangle}.\end{aligned}$$

Then,

$$\begin{aligned}\rho(\mathcal{W}_{\hat{\pi}_\psi},\mathcal{W}_{\hat{\pi}_{\hat{\psi}}}) &= \sum_{l\in V_k^*}(-1)^{\langle b_1,\mathcal{I}(l)\otimes\psi(i)\rangle\oplus\left\langle b_2,\mathcal{I}((i\otimes l^{-1})\otimes\psi(i))\right\rangle\oplus\langle a_1,l\rangle\oplus\left\langle a_2,(i\otimes l^{-1})\right\rangle}\\ &\quad-\sum_{l\in V_k^*}(-1)^{\left\langle b_1,\mathcal{I}(l)\otimes\hat{\psi}(i)\right\rangle\oplus\left\langle b_2,\mathcal{I}((i\otimes l^{-1})\otimes\hat{\psi}(i))\right\rangle\oplus\langle a_1,l\rangle\oplus\left\langle a_2,(i\otimes l^{-1})\right\rangle},\end{aligned}$$

$$\begin{aligned}\rho(\mathcal{W}_{\hat{\pi}_{\hat{\psi}}},\mathcal{W}_{\hat{\pi}_\psi}) &= \sum_{l\in V_k^*}(-1)^{\left\langle b_1,\mathcal{I}(l)\otimes\hat{\psi}(i)\right\rangle\oplus\left\langle b_2,\mathcal{I}((i\otimes l^{-1})\otimes\hat{\psi}(i))\right\rangle\oplus\langle a_1,l\rangle\oplus\left\langle a_2,(i\otimes l^{-1})\right\rangle}\\ &\quad-\sum_{l\in V_k^*}(-1)^{\langle b_1,\mathcal{I}(l)\otimes\psi(i)\rangle\oplus\left\langle b_2,\mathcal{I}((i\otimes l^{-1})\otimes\psi(i))\right\rangle\oplus\langle a_1,l\rangle\oplus\left\langle a_2,(i\otimes l^{-1})\right\rangle}.\end{aligned}$$

From here we obtain

$$\left|\rho(\mathcal{W}_{\hat{\pi}_\psi},\mathcal{W}_{\hat{\pi}_{\hat{\psi}}})\right|\leq 2\cdot(2^k-1), \tag{19}$$

$$\left|\rho(\mathcal{W}_{\hat{\pi}_{\hat{\psi}}}, \mathcal{W}_{\hat{\pi}_{\psi}})\right| \leq 2 \cdot (2^k - 1). \tag{20}$$

So, we conclude

$$\mathcal{NL}(\hat{\pi}_{\psi}) - 2^k + 1 \leq \mathcal{NL}(\hat{\pi}_{\hat{\psi}}) \leq \mathcal{NL}(\hat{\pi}_{\psi}) + 2^k - 1.$$

□

Proposition 4 can be used to increase the nonlinearity of permutation π when $h = \mathcal{I}$, which is very useful for searching nonlinear bijective transformations having good values of its basic cryptographic parameters using the construction of π.

The following proposition shows the behavior of the δ-uniformity parameter of permutations $\hat{\pi}_{\psi}, \hat{\pi}_{\hat{\psi}}$ when the lookup-tables of ψ and $\hat{\psi}$ differs only in one output value.

Proposition 5. *If the lookup-tables of non-bijective k-bit functions $\psi = \begin{pmatrix} \dots & i & \dots \\ \dots & \psi(i) & \dots \end{pmatrix}$ and $\hat{\psi} = \begin{pmatrix} \dots & i & \dots \\ \dots & \hat{\psi}(i) & \dots \end{pmatrix}$ differs from each other exactly in one output value, then for permutations $\hat{\pi}_{\psi}, \hat{\pi}_{\hat{\psi}}$ the following relations holds:*

1. $\delta_{\hat{\pi}_{\psi}} - 4 \cdot (2^k - 1) \leq \delta_{\hat{\pi}_{\hat{\psi}}} \leq \delta_{\pi_{\psi}} + 4 \cdot (2^k - 1)$, *when* $i = 0$;
2. $\delta_{\hat{\pi}_{\psi}} - 2 \cdot (2^k - 1) \leq \delta_{\hat{\pi}_{\hat{\psi}}} \leq \delta_{\hat{\pi}_{\psi}} + 2 \cdot (2^k - 1)$, *when* $i \neq 0$.

Proof. To prove the proposition is sufficient to bound the following sums

$$\Delta_{\hat{\pi}_{\psi}}(a, b) = \sum_{x \in V_n} \mathsf{Ind}(\hat{\pi}_{\psi}(x \oplus a) \oplus \hat{\pi}_{\psi}(x), b),$$

$$\Delta_{\hat{\pi}_{\hat{\psi}}}(a, b) = \sum_{x \in V_n} \mathsf{Ind}(\hat{\pi}_{\hat{\psi}}(x \oplus a) \oplus \hat{\pi}_{\hat{\psi}}(x), b).$$

1. Denote by $\omega_t, t = 1 \dots, 2 \cdot (2^k - 1)$ those points of V_{2k} for which $\hat{\pi}_{\psi}(\omega_t) \neq \hat{\pi}_{\hat{\psi}}(\omega_t)$. In other points, the output values of permutations $\hat{\pi}_{\psi}, \hat{\pi}_{\hat{\psi}}$ are equals, i.e., for any $x \in V_{2k} \setminus \{\omega_t | t = 1, \dots, 2 \cdot (2^k - 1)\}$ the following relation holds $\hat{\pi}_{\psi}(x) = \hat{\pi}_{\hat{\psi}}(x)$.

It is well-known that if $x = \omega_t, \omega_t \oplus a$, $t = 1, \dots, 2 \cdot (2^k - 1)$ are solutions of the equation $\hat{\pi}_{\psi}(x \oplus a) \oplus \hat{\pi}_{\psi}(x) = b$, then

$$\begin{cases} \hat{\pi}_{\psi}(\omega_t \oplus a) = \hat{\pi}_{\psi}(\omega_t) \oplus b; \\ \\ \quad t = 1, \dots, 2 \cdot (2^k - 1). \end{cases} \tag{21}$$

It is not difficult to see that for $a \neq 0$ the following relations holds $\omega_t \neq \omega_t \oplus a$, where $t = 1, \ldots, 2 \cdot (2^k - 1)$. Then, the next equalities are true

$$\hat{\pi}_\psi(\omega_t \oplus a) = \hat{\pi}_{\hat{\psi}}(\omega_t \oplus a), t = 1, \ldots, 2 \cdot (2^k - 1). \tag{22}$$

When $t = 1, \ldots, 2 \cdot (2^k - 1)$, the conditions $\hat{\pi}_\psi(\omega_t) \neq \hat{\pi}_{\hat{\psi}}(\omega_t)$ are equivalents to the following

$$\hat{\pi}_\psi(\omega_t) \oplus b \neq \hat{\pi}_{\hat{\psi}}(\omega_t) \oplus b, \tag{23}$$

where $b \in V_{2k}$. Then from (21),(22) and (23) under the condition that $x = \omega_t, \omega_t \oplus a$ are solutions of the equation $\hat{\pi}_\psi(x \oplus a) \oplus \hat{\pi}_\psi(x) = b$ we obtain

$$\hat{\pi}_{\hat{\psi}}(\omega_t \oplus a) \neq \hat{\pi}_{\hat{\psi}}(\omega_t) \oplus b, t = 1, \ldots, 2 \cdot (2^k - 1). \tag{24}$$

This means that if $x = \omega_t, \omega_t \oplus a$ where $t = 1, \ldots, 2 \cdot (2^k - 1)$, are solutions of the equation $\hat{\pi}_\psi(x \oplus a) \oplus \hat{\pi}_\psi(x) = b$, then they can not be solutions of the equation $\hat{\pi}_{\hat{\psi}}(x \oplus a) = \hat{\pi}_{\hat{\psi}}(x) \oplus b$.

Now, we shall prove that $\{\omega_t, \omega_t \oplus a\} \cap \{\omega_s, \omega_s \oplus a\} = \emptyset$ for any $t, s \in \{1, \ldots, 2 \cdot (2^k - 1)\}$, where $t \neq s$. It is sufficient to consider the following cases:

$\boldsymbol{\omega_t = \omega_s}$. According to proposition 3 when $i = 0$, we have $\chi(\hat{\pi}_\psi, \hat{\pi}_{\hat{\psi}}) = \#\{x \in V_{2k} | \hat{\pi}_\psi(x) \neq \hat{\pi}_{\hat{\psi}}(x)\} = 2 \cdot (2^k - 1)$. In this case, if $\omega_t = \omega_s$, where $t, s \in \{1, \ldots, 2 \cdot (2^k - 1)\}$, and $t \neq s$, then $\chi(\hat{\pi}_\psi, \hat{\pi}_{\hat{\psi}}) = \#\{x \in V_{2k} | \hat{\pi}_\psi(x) \neq \hat{\pi}_{\hat{\psi}}(x)\} < 2 \cdot (2^k - 1)$, which yields a contradiction;

$\boldsymbol{\omega_t = \omega_s \oplus a}$. In this case, by using relations (22) we obtain $\hat{\pi}_\psi(\omega_t) = \hat{\pi}_\psi(\omega_s \oplus a) = \hat{\pi}_{\hat{\psi}}(\omega_s \oplus a) = \hat{\pi}_{\hat{\psi}}(\omega_t)$ which yields a contradiction because $\hat{\pi}_\psi(\omega_t) \neq \hat{\pi}_{\hat{\psi}}(\omega_t)$ for $t = 1, \ldots, 2 \cdot (2^k - 1)$.

So, the sum $\Delta_{\hat{\pi}_\psi}(a, b)$ can be decomposed as follows

$$\begin{aligned} \Delta_{\hat{\pi}_\psi}(a, b) \;=\; & \sum_{x \in \bigsqcup_{t=1}^{2\cdot(2^k-1)} \{\omega_t, \omega_t \oplus a\}} \mathsf{Ind}(\hat{\pi}_\psi(x \oplus a) \oplus \hat{\pi}_\psi(x), b) \\ & + \sum_{x \in V_{2k} \setminus \bigsqcup_{t=1}^{2\cdot(2^k-1)} \{\omega_t, \omega_t \oplus a\}} \mathsf{Ind}(\hat{\pi}_{\hat{\psi}}(x \oplus a) \oplus \hat{\pi}_{\hat{\psi}}(x), b). \end{aligned}$$

As for any $t = 1, 2, \ldots, 2 \cdot (2^k - 1)$, the following relations holds

$$\sum_{x \in V_n \setminus \bigsqcup_{t=1}^{2\cdot(2^k-1)} \{\omega_t, \omega_t \oplus a\}} \mathsf{Ind}(\hat{\pi}_{\hat{\psi}}(x \oplus a) \oplus \hat{\pi}_{\hat{\psi}}(x), b) \leq \sum_{x \in V_n} \mathsf{Ind}(\hat{\pi}_{\hat{\psi}}(x \oplus a) \oplus \hat{\pi}_{\hat{\psi}}(x), b), \tag{25}$$

and

$$\sum_{x \in \bigsqcup_{t=1}^{2\cdot(2^k-1)} \{\omega_t, \omega_t \oplus a\}} \mathsf{Ind}(\hat{\pi}_\psi(x \oplus a) \oplus \hat{\pi}_\psi(x), b) \leq 2 \cdot (2 \cdot (2^k - 1)), \tag{26}$$

then from (25), (26) we obtain

$$\delta_{\hat{\pi}_\psi}(a, b) \leq \delta_{\hat{\pi}_{\hat{\psi}}}(a, b) + 4 \cdot (2^k - 1) \tag{27}$$

Analogously, we can derive that for $\delta_{\hat{\pi}_{\hat{\psi}}}(a, b)$ the following inequality holds

$$\delta_{\hat{\pi}_{\hat{\psi}}}(a, b) \leq \delta_{\hat{\pi}_\psi}(a, b) + 4 \cdot (2^k - 1) \tag{28}$$

Hence, from relations (27), (28) we conclude

$$\delta_{\hat{\pi}_\psi} - 4 \cdot (2^k - 1) \leq \delta_{\hat{\pi}_{\hat{\psi}}} \leq \delta_{\hat{\pi}_\psi} + 4 \cdot (2^k - 1).$$

2. The proof is quite similar to the proof of the item *1.*

□

Proposition 5, tell us that changing only one output value of the non-bijective k-bit function ψ (which has no preimage for 0) the δ-uniformity of π may decrease, which is quite useful when searching nonlinear bijective transformations having good values of its basic cryptographic parameters using the construction of π.

When $n = 8$, in correspondence with the suggested constructions of π, we need to construct; the 4-bit non-bijective function ψ , the 4-bit permutation $\mathcal{I}(x) = x^{14}$ over GF(2^4). In what follows, we shall work over the finite field GF(2^4)=GF(2)[ξ]/$g(\xi)$, constructing the latter with the irreducible polynomial $g(\xi) = \xi^4 + \xi + 1$.

By using propositions 4 and 5 we have conducted two search algorithms (implemented in SAGE [44]) for finding ordinary 8-bit S-Boxes having the following cryptographic parameters

- minimum algebraic degree equal to 7;
- graph algebraic immunity equal to 3 (with 441 equations);
- δ-uniformity equal to 6 or 8;
- nonlinearity in range of 100 up to a value of 104.

The algorithms are modified versions of algorithms for implementing the spectral-linear and spectral-differential methods presented in [31] and both of them operates with the following objects: $(a, b, c) \in \Xi_0(V_k) \times S(V_{2k}) \times \mathbb{Z}$, where by $\Xi_0(V_k)$ is denoted the set of all non-bijective functions $\psi : V_k \to V_k$

which have preimage for 0 . We define on the set of these objects the next order relation as follows

$$(\tilde{a}, \tilde{b}, \tilde{c}) \leq (a, b, c), \text{ if } \tilde{b} < b \text{ or } \tilde{b} = b. \tag{29}$$

Let $\ell \in \mathbb{N}$ be the size of some list L, which should be chosen according to available computing resources, then the algorithm for improving the differential proprieties in correspondence with our design criteria is presented bellow.

Algorithm 1: Optimizing differential properties of $\hat{\pi}$

Input: Permutation $\mathcal{I} = x^{-1}$ over $\mathrm{GF}(2^k)$, parameter $\ell \in \mathbb{N}$.

1 Generate randomly a non-bijective k-bit function ψ and construct $\hat{\pi}_\psi = (\mathcal{I}(l) \otimes \psi(l \otimes r)) \| \mathcal{I}(r \otimes \psi(l \otimes r)) \in S(V_{2k})$.

2 For permutation $\hat{\pi}_\psi \in S(V_{2k})$ calculate the value $\delta_{\hat{\pi}_\psi}$.

3 Initialize the list L:

$$\mathsf{L} = \{(\psi, \hat{\pi}_\psi, \delta_{\hat{\pi}_\psi})\}, \text{where } \#\mathsf{L} = 1.$$

4 Using the list $\mathsf{L} = \{(\psi^{(i)}, \hat{\pi}_{\psi^{(i)}}, \delta_{\hat{\pi}_{\psi^{(i)}}}) | i = 0, \ldots, \#\mathsf{L} - 1\}$ construct the new list

$$\widetilde{\mathsf{L}} = \{(\hat{\psi}^{(i)}_{j,t}, \hat{\pi}_{\hat{\psi}^{(i)}_{j,t}}, \delta_{\hat{\pi}_{\hat{\psi}^{(i)}_{j,t}}})\}, \text{where } \#\widetilde{\mathsf{L}} \leq \#\mathsf{L} \cdot 2^k \cdot (2^k - 2),$$

and for each $i = 0, \ldots, \#\mathsf{L} - 1$, $\delta_{\hat{\pi}_{\hat{\psi}^{(i)}_{j,t}}} \leq \delta_{\hat{\pi}_{\psi^{(i)}}}$, the non-bijective k-bit functions $\hat{\psi}^{(i)}_{j,t}, j = 0, \ldots, 2^k - 1, t = 0, \ldots 2^k - 3$, differs from $\psi^{(i)}$ exactly in one output value.

5 For the list $\widetilde{\mathsf{L}}$ do the following:

(I) Calculate the size $\#\widetilde{\mathsf{L}}$.
(II) Sort the elements of $\widetilde{\mathsf{L}}$ in the ascending order according to relation (29).
(III) Numerate the sorted list element by indexes $i = 0, \ldots, \#\widetilde{\mathsf{L}} - 1$.
(IV) Calculate values $m_1 = \min\{\#\mathsf{L} - 1, \#\widetilde{\mathsf{L}} - 1\}, m_2 = \min\{\ell - 1, \#\widetilde{\mathsf{L}} - 1\}$.

6 Compare the first elements of list L and $\widetilde{\mathsf{L}}$:

– If $\sum_{i=0}^{m_1} \delta_{\hat{\pi}_{\hat{\psi}^{(i)}}} < \sum_{i=0}^{m_1} \delta_{\hat{\pi}_{\psi^{(i)}}}$, then

(I) Clean the list L.
(II) Copy the elements from the list $\widetilde{\mathsf{L}}$ with indexes $i = 0, \ldots, m_2$ to L.
(III) Assign $\#\mathsf{L} = m_2 + 1$.
(IV) Go to step 4.

– Otherwise, the algorithm stops.

Output: The list $\widetilde{\mathsf{L}} = \{(\psi^{(i)}, \hat{\pi}_{\psi^{(i)}}, \delta_{\hat{\pi}_{\psi^{(i)}}}) | i = 0, \ldots, \#\widetilde{\mathsf{L}} - 1\}$, $\#\widetilde{\mathsf{L}} \leq \ell$.

Replacing in algorithm 1 the δ-uniformity parameter by the nonlinearity and making the appropriate changes in steps 4,6 we can easy obtain the algorithm for optimizing the (non)linear properties of $\hat{\pi}$, which is omitted

due to space limitations. The best results of this work have been achieved by using both algorithms.

4.3 Invariant subspaces with respect to the action of $\hat{\pi}$

In this section, we study the question about the existence of subspaces W of the vector space V_n such that $\hat{\pi}(\mathsf{W} \oplus a) = \mathsf{W} \oplus b$ for some fixed elements $a, b \in V_n$. Such subspaces are called invariant subspaces and are used in recent cryptanalytic approaches when mounting structural attacks on block ciphers (for example, in the so-called invariant subspaces attacks [29]). The existence of such structures can significantly decrease the cryptographic security of block ciphers. In [2, 43] were described some approaches for designing cryptographic primitives having a structure, knowledge of which allows to find the encryption key with a time complexity, significantly lower than the brute force method. Such structure is called a backdoor, and the whole encryption algorithm —backdoored encryption algorithm.

Another fundamental cryptanalytic method for block ciphers is the homomorphism attack. The effectiveness of this approach highly dependent on how close the encryption function can be approximated by permutations having the partition-preserving property. The authors of [41] studied the possibility to approximate permutations by permutations from the wreath product of symmetric groups in an imprimitive action, where the so-called W-intersection matrix was proposed as a parameter characterizing the approximability of permutations by permutations from the wreath group. The W-intersection matrix for a permutation Φ of $S(V_n)$ is defined as follows

$$\mathcal{M}_{\mathsf{W}}(\Phi) = \left\| c^{\mathsf{W}}_{\alpha,\beta}(\Phi) \right\|_{\alpha,\beta \in \mathcal{R}_{\mathsf{W}}},$$

where $c^{\mathsf{W}}_{\alpha,\beta}(\Phi) = \#\left\{ x \in \mathsf{W} \oplus \alpha \middle| \Phi(x) \in \mathsf{W} \oplus \beta \right\}$, $\mathsf{W} < V_n$, $\dim \mathsf{W} = d \in \{1, 2, \ldots, n-1\}$ and $\mathcal{R}_{\mathsf{W}}$ is the set of coset representatives for the subspace $\mathsf{W} < V_n$.

The W-intersection matrix is a very useful tool to automatically verify the invariance of a fixed subspace W with respect to the action of some given nonlinear bijective transformation.

Proposition 6. *Let* $\mathsf{W}_1 = \{(l\|0) | l \in V_k\}, \mathsf{W}_2 = \{(0\|r) | r \in V_k\}$ *be two* k*-dimensional subspaces of the vector space* V_{2k} *. Then*

$$c^{\mathsf{W}_1}_{0,0}(\hat{\pi}) = c^{\mathsf{W}_2}_{0,0}(\hat{\pi}) = 2^k. \tag{30}$$

Proof. The relations written in (30) are a direct consequence of the equality (10), for $h = \mathcal{I}$. □

Corollary 1. *The following k-dimensional subspaces $\mathsf{W}_1 \oplus (\alpha_1 \| 0)$ and $\mathsf{W}_2 \oplus (0 \| \alpha_2)$ are invariant with respect to the action of $\hat{\pi}$ for any $\alpha_1, \alpha_2 \in V_k$.*

Example 1. *Let $n = 2k = 2 \cdot 4$ and $\mathrm{GF}(2^4) = \mathrm{GF}[2][\xi] / \xi^4 \oplus \xi \oplus 1$, the 4-bit components ψ[1], $\mathcal{I}$ are given as follows*

$$\psi = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ 7 & 12 & 3 & 12 & 12 & 9 & 13 & 13 & 8 & 2 & 2 & 11 & 9 & 15 & 2 & 3 \end{pmatrix},$$

$$\mathcal{I} = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ 0 & 1 & 9 & 14 & 13 & 11 & 7 & 6 & 15 & 2 & 12 & 5 & 10 & 4 & 3 & 8 \end{pmatrix}.$$

Then, the resulting permutation $\hat{\pi}(l \| r) = (\mathcal{I}(l) \otimes \psi(l \otimes r)) \| \mathcal{I}(r \otimes \psi(l \otimes r)) \in S(V_8)$ and its cryptographic parameters are compiled in the following table.

S-Box $\hat{\pi}$															
$\mathcal{NL}(\hat{\pi}) = 104, \delta_{\hat{\pi}} = 6, \deg(\hat{\pi})_{(\mathsf{min})} = 7, \mathcal{AI}_{gr}(\hat{\pi}) = 3, r^{(3)}_{\hat{\pi}} = 441.$															
0x0	0x6	0x3	0x2	0x8	0xf	0x1	0x7	0x4	0xc	0xe	0xd	0x9	0xb	0xa	0x5
0x70	0xca	0x37	0xc6	0xcb	0x95	0xdf	0xdb	0x8a	0x21	0x26	0xb2	0x97	0xf6	0x28	0x39
0xa0	0x8e	0x65	0xfd	0x47	0x1c	0xde	0x13	0x6c	0x67	0xf5	0xda	0xc4	0x12	0x81	0xec
0xc0	0x4a	0xa2	0x7f	0x79	0x18	0xfa	0xf3	0x86	0x9d	0x5a	0xfb	0xae	0x4e	0x4d	0x19
0x50	0x3a	0x2e	0xff	0x3b	0xea	0x68	0x42	0xe9	0x4f	0x96	0x9b	0xf7	0x3e	0x7b	0x94
0x40	0xc2	0x5d	0xeb	0x61	0xe8	0x3d	0x74	0x5e	0x9a	0xd1	0xd4	0x55	0xc8	0xdd	0x66
0x60	0x54	0xa1	0xe7	0x4c	0xb7	0x5f	0x29	0xad	0x27	0xe6	0x93	0xe5	0xd9	0x91	0x2f
0x10	0x84	0xcd	0xc7	0xaa	0x53	0xe3	0x8b	0x41	0xc1	0xe1	0xe4	0xa6	0x38	0x36	0xfe
0xb0	0x1f	0x85	0x33	0x71	0xdc	0xee	0xa5	0xed	0x87	0x24	0x77	0xd5	0x2d	0xd8	0x8f
0xe0	0x49	0xb5	0x35	0x6a	0x51	0xb3	0x43	0xbc	0xd3	0x1b	0x1a	0x9e	0x6d	0x9c	0x44
0x20	0xb9	0x32	0x89	0xbf	0xf2	0xba	0xf9	0x75	0x64	0xa8	0x73	0xf8	0xd7	0x3c	0x63
0x80	0x15	0xb1	0xa7	0xaf	0x92	0xfc	0x99	0xc9	0xb4	0xf4	0xab	0x6f	0xc3	0xe2	0x9f
0x30	0x52	0x2b	0xbd	0x59	0x7c	0x7a	0xd2	0x7e	0xb8	0x11	0xce	0xd6	0x1e	0x1d	0xf1
0xf0	0x98	0x8d	0x56	0x5b	0x25	0x6b	0x2c	0xc5	0xcf	0xa9	0x17	0x58	0x82	0x88	0x16
0x90	0x69	0x57	0x76	0x22	0x72	0x5c	0x8c	0x6e	0x48	0x45	0xb6	0x78	0x62	0xef	0x83
0xd0	0xbe	0x14	0xbb	0x3f	0x2a	0xa3	0x7d	0xac	0x31	0x4b	0xa4	0xcc	0x23	0x46	0x34

Table 1: The constructed permutation $\hat{\pi} \in S(V_8)$.

From Table 1, we can see that the nonlinear bijective transformation $\hat{\pi} \in S(V_8)$ exhibit high values of its basic cryptographic parameters and it does not have polynomial relations of low degree.

Let us now verify the existence of some invariant subspaces with respect to the action of the constructed permutation $\hat{\pi} \in S(V_8)$. The W-intersection matrices $\mathcal{M}_{\mathsf{W}_i}(\hat{\pi}) = \left\| c^{\mathsf{W}_i}_{\alpha,\beta}(\hat{\pi}) \right\|_{\alpha,\beta \in \mathcal{R}_{\mathsf{W}_i}}$ given by (31), for subspaces $\mathsf{W}_1 = \{(l \| 0) | l \in V_4\}$, $\mathsf{W}_2 = \{(0 \| r) | r \in V_4\}$ of the vector space V_8 were found by computer calculations using SAGE [44].

[1]This component has been found using the algoritmhs described in Section 3.2.

From matrices $\mathcal{M}_{\mathsf{W}_i}(\hat{\pi}), i = 1, 2$, we can see that $c_{0,0}^{\mathsf{W}_1}(\hat{\pi}) = c_{0,0}^{\mathsf{W}_2}(\hat{\pi}) = 16$, which means that $\hat{\pi}(\mathsf{W}_i) = \mathsf{W}_i$. Hence the subspaces $\mathsf{W_1}$ and $\mathsf{W_2}$ are invariant under the action of the constructed permutation $\hat{\pi} \in S(V_8)$.

$$\mathcal{M}_{\mathsf{W}_1}(\hat{\pi}) = \begin{pmatrix}
\boxed{16} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 2 & 0 & 2 & 1 & 1 & 0 & 1 & 3 & 3 & 0 & 0 & 0 & 2 & 1 \\
0 & 2 & 2 & 1 & 1 & 3 & 0 & 2 & 0 & 0 & 0 & 1 & 0 & 3 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 1 & 3 & 3 & 0 & 1 & 0 & 2 & 0 & 2 & 0 & 2 \\
0 & 2 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 2 & 2 & 3 & 1 & 0 & 0 & 3 \\
0 & 1 & 3 & 1 & 0 & 2 & 0 & 1 & 2 & 0 & 2 & 0 & 3 & 0 & 0 & 1 \\
0 & 1 & 0 & 3 & 0 & 0 & 0 & 0 & 1 & 0 & 3 & 1 & 2 & 1 & 2 & 2 \\
0 & 0 & 2 & 3 & 1 & 1 & 0 & 1 & 0 & 3 & 0 & 2 & 2 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 2 & 1 & 0 & 0 & 2 & 1 & 0 & 3 & 2 & 3 & 0 \\
0 & 3 & 0 & 1 & 2 & 0 & 0 & 3 & 2 & 0 & 1 & 0 & 1 & 1 & 0 & 2 \\
0 & 3 & 0 & 0 & 2 & 2 & 3 & 0 & 1 & 1 & 1 & 2 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 2 & 3 & 0 & 1 & 2 & 0 & 0 & 2 & 3 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 3 & 2 & 2 & 3 & 1 & 0 & 0 & 1 & 0 & 2 & 1 \\
0 & 0 & 3 & 2 & 0 & 0 & 1 & 1 & 2 & 1 & 0 & 1 & 0 & 2 & 3 & 0 \\
0 & 2 & 1 & 0 & 0 & 0 & 2 & 0 & 3 & 0 & 1 & 1 & 2 & 3 & 0 & 1 \\
0 & 1 & 0 & 2 & 3 & 1 & 2 & 0 & 0 & 2 & 0 & 0 & 1 & 0 & 1 & 3
\end{pmatrix}, \mathcal{M}_{\mathsf{W}_2}(\hat{\pi}) = \begin{pmatrix}
\boxed{16} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 3 & 2 & 0 & 0 & 0 & 1 & 1 & 2 & 0 & 1 & 3 & 2 & 0 & 1 \\
0 & 3 & 0 & 0 & 1 & 0 & 3 & 0 & 2 & 0 & 1 & 0 & 1 & 2 & 1 & 2 \\
0 & 2 & 0 & 0 & 3 & 1 & 0 & 2 & 1 & 1 & 2 & 0 & 1 & 0 & 0 & 3 \\
0 & 0 & 1 & 3 & 2 & 1 & 1 & 1 & 0 & 3 & 0 & 0 & 0 & 0 & 2 & 2 \\
0 & 0 & 0 & 1 & 1 & 3 & 2 & 1 & 0 & 1 & 0 & 0 & 2 & 3 & 2 & 0 \\
0 & 0 & 3 & 0 & 1 & 2 & 1 & 0 & 0 & 2 & 2 & 1 & 0 & 1 & 3 & 0 \\
0 & 1 & 0 & 2 & 1 & 1 & 0 & 0 & 2 & 0 & 2 & 0 & 3 & 0 & 3 & 1 \\
0 & 1 & 2 & 1 & 0 & 0 & 0 & 2 & 3 & 0 & 1 & 1 & 0 & 3 & 2 & 0 \\
0 & 2 & 0 & 1 & 3 & 1 & 2 & 0 & 0 & 2 & 0 & 3 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 2 & 0 & 0 & 2 & 2 & 1 & 0 & 1 & 3 & 0 & 1 & 0 & 3 \\
0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 3 & 3 & 2 & 2 & 0 & 1 & 2 \\
0 & 3 & 1 & 1 & 0 & 2 & 0 & 3 & 0 & 0 & 0 & 2 & 1 & 2 & 0 & 1 \\
0 & 2 & 2 & 0 & 0 & 3 & 1 & 0 & 3 & 1 & 1 & 0 & 2 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 2 & 2 & 3 & 3 & 2 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 1 & 2 & 3 & 2 & 0 & 0 & 1 & 0 & 0 & 3 & 2 & 1 & 1 & 0 & 0
\end{pmatrix}. \tag{31}$$

So, despite the fact that permutation $\hat{\pi} \in S(V_8)$ exhibit a low value of δ-uniformity, high nonlinearity and can be described by a system of 441 polynomials equations of degree 3, it has a weakness: the existence of some structures (subspaces $\mathsf{W_1}$ and $\mathsf{W_2}$) which are invariant with respect to the action of this nonlinear bijective transformation. If using this permutation as nonlinear layer in a XSL-network, then these structures should be taken into account when designing the linear layer and the key-expansion algorithm, to avoid the existence of a large number of weak-keys for the encryption function. However, this weakness can be eliminated by choosing appropriate linear (resp. affine) layers $\mathcal{L}_1$ and $\mathcal{L}_2$ from $\mathsf{GL}_8(\mathrm{GF}(2))$.

When looking at the TU-decomposition (see e.g., [4]) of the 8-bit S-Box used in the block cipher Kuznyechik [13], denoted here by $\pi_{\mathsf{Kuz}} = \alpha \circ \hat{\pi}_{\mathsf{Kuz}} \circ \omega$, where $\alpha, \omega \in \mathsf{GL}_8(\mathrm{GF}(2))$ and $\hat{\pi}_{\mathsf{Kuz}}$ is a permutation based on Feistel-like structure, we have found by using the W-intersection matrix that the subspace $\mathsf{W_1} = \{(l \| 0) | l \in V_4\}$ is invariant with respect to the action of the nonlinear bijective transformation $\hat{\pi}_{\mathsf{Kuz}} = \omega^{-1} \circ \pi_{\mathsf{Kuz}} \circ \alpha^{-1}$, i.e., $\hat{\pi}_{\mathsf{Kuz}}(\mathsf{W_1} \oplus 0\mathrm{x}c) = \mathsf{W_1}$. However, by computing $\mathcal{M}_{\mathsf{W}_i}(\pi_{\mathsf{Kuz}}), i = 1, 2$, we have checked the absence of invariant subspaces such as $\mathsf{W_1}$ and $\mathsf{W_2}$ in the permutation π_{Kuz}.

In the above cases, we have seen the important role played by the linear layers used in those constructions, which also explain why we have inserted these matrices in the original construction of π. Its purposes are not only to break the cycle structure and eliminate the existence of fixed points but also circumvent the presence of invariant subspaces such as $\mathsf{W_1}$ and $\mathsf{W_2}$.

4.4 Constructing highly-nonlinear involutions

In this section we will study how to build involutive S-Boxes with strong cryptographic properties using constructions presented in the previous section. Involutions, have an particular interest in Cryptography because in the case of lightweight block ciphers, these components are used to decrease the cost of the implementation of decryption process.

Definition 11. *Let ε be the identity permutation of $S(V_n)$. A permutation $\Phi \in S(V_n)$ is called an involution if $\Phi \circ \Phi = \varepsilon$.*

Even when the function $\mathcal{I}$ is an involution on $S(V_k)$ and the permuttaion $h \in S(V_k)$ can be chosen to be involution too, the permuations generated by $\hat{\pi}$ are not always involutions. Taking $h = \mathcal{I}$, in order to achieve the property $\hat{\pi} \circ \hat{\pi} = \varepsilon$ we have performed a search algorithm . The algorithm take as input a randomly generated non-bijective 4-bit function ψ and for this ψ the resulting permutation $\hat{\pi}$ is constructed. Then, the Hamming distance $\chi(\varepsilon, \hat{\pi} \circ \hat{\pi})$ is calculated. If $\chi(\varepsilon, \hat{\pi} \circ \hat{\pi}) = 0$ and $\hat{\pi}$ satisfy the set of cryptographic criteria (listed in Section 2), the algorithm stops and as output we get an nonlinear involution. Otherwise, in an iterative process ψ is changed randomly (in a arbitrary number of positions) until $\chi(\varepsilon, \hat{\pi} \circ \hat{\pi})$ is equal to zero, which means that a involution is founded. We repeat the above procedure until an involution $\hat{\pi}$ with the properties listed in the set of cryptographic criteria has been founded.

We have implemented this algorithm in SAGE [44] obtaining some involutions having few fixed points with the following cryptografic properties

- minimum algebraic degree — 7;
- graph algebraic immunity — 3 (with 441 equations);
- δ-uniformity equal to 6 or 8;
- nonlinearity in range of 100 up to a value of 104.

Also, we have tried to design directly involutions using ours scheme as building block. To achieve the fulfillment of condition $\Phi \circ \Phi(x) = x$, our strategy was to combine our constructions into two or more rounds. Choosing two arbitrary k-bit involutions h_1, h_2, the following constructions is able to produce $2k$-bit involutions.

Construction of $\hat{\pi}^{(invol)}$

For the input value $(l\|r) \in V_{2k}$ we define the corresponding output value as follows

$$\hat{\pi}^{(invol)}(l\|r) = \big(\hat{\pi}_3 \circ \hat{\pi}_2 \circ \hat{\pi}_1\big)(l\|r) = l_1\|r_1 \text{ where,}$$

$$\hat{\pi}_1(l\|r) = (l \otimes \mathcal{I}(\psi(l \otimes r)))\Big\|(r \otimes \psi(l \otimes r));$$

$$\hat{\pi}_2(l\|r) = h_1(l)\Big\|h_2(r);$$

$$\hat{\pi}_3(l\|r) = \Big(l \otimes \psi(l \otimes r)\Big)\Big\|\Big(r \otimes \mathcal{I}(\psi(l \otimes r))\Big).$$

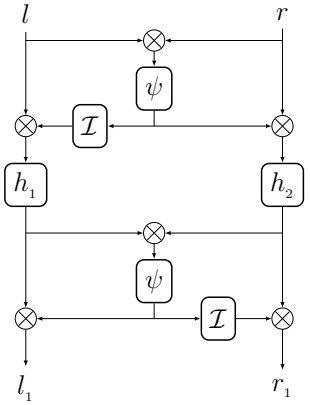


Fig 2: Structure of $\hat{\pi}^{(invol)}$.

As we can see from Figure 2, the construction of $\hat{\pi}^{(invol)}$ represent a composition of three functions $\hat{\pi}_3, \hat{\pi}_2$ and $\hat{\pi}_1$, where $\hat{\pi}_3$ and $\hat{\pi}_1$ have similarities with 1-round of Lai-Massey scheme. The involution property of the whole construction can be derived from the well-known fact, that if M is an involution over V_n, then for any permutation $G \in V_n$, the resulting transformation $F = G^{-1} \circ M \circ G$ is an involution over V_n. Here $F(l\|r) = \hat{\pi}_{invol}, G(l\|r) = \Big(l \otimes \mathcal{I}(\psi(l \otimes r)))\Big\|(l \otimes \psi(l \otimes r))\Big), M(l\|r) = h_1(l)\|h_2(r)$ and $G^{-1}(l\|r) = \Big((l \otimes \psi(l \otimes r))\|(l \otimes \mathcal{I}(\psi(l \otimes r))\Big)$.

Using the previous construction we have performed a search based on random generation of 4-bit involutions and non-bijective 4-bit function ψ (which has no preimage for 0) for finding almost optimal involutions without fixed points (in contrast to those generated by the construction of $\hat{\pi}$) with the following parameters

- minimum algebraic degree equal to 7;
- δ-uniformity equal to 8;
- graph algebraic immunity equal to 3 (with 441 equations);
- nonlinearity in range of 100 up to a value of 102.

The possibility of having no fixed points in those involutions constructed under the $\hat{\pi}^{(invol)}$. scheme has some significances. In fact, the involutions produced by this construction have more finite field multiplications which have an impact on the masking complexity of these kind of permutations in comparison with those involutions generated by $\hat{\pi}$ (Section 4). Moreover, the cryptographic properties related to linear and differential cryptanalysis of involutions based on $\hat{\pi}^{(invol)}$ -construction slightly decrease in comparison with those generated by $\hat{\pi}$.

4.5 Searching of highly-nonlinear orthomorphisms

In this section we will study the possibility of using our algorithmic-algebraic scheme to find a special kind of the so-called complete mappings. Complete mapping were first introduced by Mann [30] and the term orthomorphisms was first used by Johnson, Dulmage and Mendelsohn [21] and were also studied in the following articles [10, 11, 37, 38, 31, 32, 33, 34, 48]. Orthomorphisms are pertinent to the construction of mutually orthogonal Latin squares and can be used to design check digit systems.

In Cryptography, applications of orthomorphisms of the group $(V_n, \oplus)$ are found in the construction of block ciphers, stream ciphers and hash functions (in the Lai-Massey scheme most famously in well-known FOX [46] family of block ciphers, chinese stream cipher LOISS [19] and hash function EDON-R [17]). More recently, orthomorphisms have been used to strengthen the Even-Mansour block cipher against some cryptographic attacks [16].

Definition 12. *A permutation $\Phi \in S(V_n)$ is called ortomorphism on V_n, if the mapping $\widehat{\Phi} : V_n \to V_n$, defined as $\widehat{\Phi}(x) = x \oplus \Phi(x)$ is a permutation of $S(V_n)$.*

The set of all ortomorphisms of the additive group V_n is denoted by $\mathsf{Orth}(V_n)$. For any permutation $\Phi \in S(V_n)$ we define the following sets as follows

$$\mathcal{D}_\Phi = \left\{ \hat{\Phi}(x) \middle| x \in V_n \right\} = \left\{ \Phi(x) \oplus x \middle| x \in V_n \right\}, \widetilde{\mathcal{D}}_\Phi = V_n \setminus \mathcal{D}_\Phi. \tag{32}$$

Definition 13. *For any $\Phi \in S(V_n)$ the deficit of Φ, denoted by d_Φ, is defined as*

$$\mathsf{d}_\Phi = \#\widetilde{\mathcal{D}}_\Phi = 2^n - \#\mathcal{D}_\Phi. \tag{33}$$

From the above definition we have that $\Phi \in \mathsf{Orth}(V_n)$ if and only if $\mathsf{d}_\Phi = 0$, i.e., when $\#\mathcal{D}_\Phi = 2^n$.

Proposition 7. *For any $\Phi \in \mathsf{Orth}(V_n)$ the following relations holds $\mathcal{W}_\Phi(a, b) = \mathcal{W}_{\widehat{\Phi}}(a \oplus b, b)$ and $\Delta_\Phi(a, b) = \Delta_{\hat{\Phi}}(a, a \oplus b)$.*

Proof. If the permutation $\Phi \in S(V_n)$ is an ortomorphism on V_n, then for all $a, b \in V_n$, $\mathcal{W}_\Phi(a, b) = \sum_{x \in V_n} (-1)^{\langle b, \Phi(x) \rangle \oplus \langle a, x \rangle} = \sum_{x \in V_n} (-1)^{\left\langle b, \widehat{\Phi}(x) \right\rangle \oplus \langle a \oplus b, x \rangle} = \mathcal{W}_{\widehat{\Phi}}(a \oplus b, b)$. Analogously, we can easy obtain that $\Delta_\Phi(a, b) = \Delta_{\hat{\Phi}}(a, a \oplus b)$ holds for all $a, b \in V_n$. □

The next proposition shows that regardless the choice of the function ψ we can not construct orthomorphisms over V_n using the construction of $\hat{\pi}_\psi$.

Proposition 8. *Let $\psi : V_k \to V_k$ be an arbitrary non-bijective function which has no preimage for 0. Then, for the permutation $\hat{\pi}_\psi : V_{2k} \to V_{2k}$, defined as $\hat{\pi}_\psi = (\mathcal{I}(l) \otimes \psi(l \otimes r))\|\mathcal{I}(r \otimes \psi(l \otimes r))$, the following inequality holds*

$$\#\mathcal{D}_{\hat{\pi}_\psi} < 2^{2k} \tag{34}$$

Proof. Let us fix an arbitrary non-bijective k-bit function ψ which has no preimage for 0 and construct the permutation $\hat{\pi}_\psi = (\mathcal{I}(l) \otimes \psi(l \otimes r))\|\mathcal{I}(r \otimes \psi(l \otimes r))$. As for any $a, b \in \mathrm{GF}(2^k) \setminus \{0\}$ the equation $a \otimes x = b$ has a unique solution, then for some primitive element $c \in \mathrm{GF}(2^k)$, from relations $c^{-i} = c^i \otimes c^{-2i}$, where $i = 0, \ldots, 2^k - 2$, we have that for $l = 0, r = c^i, \psi(0) = c^{-2i}$ the following equalities holds

$$\hat{\pi}_\psi(0\|r) \oplus (0\|r) = (0\|\mathcal{I}(r \otimes \psi(0))) \oplus (0\|r) = (0\|c^i) \oplus (0\|c^i) = 0\|0 = 0.$$

Now, taking into account that $\mathsf{ord}(c^{-2}) = 2^k - 1$, we obtain $\{c^{-2i}|i = 0, \ldots, 2^k - 2\} = \mathrm{GF}(2^k) \setminus \{0\}$. So, independently of the choice of the output value $\psi(0)$, the following relations are always true

$$0 = \hat{\pi}_\psi(0\|0) \oplus (0\|0) = \hat{\pi}_\psi(0\|c^i) \oplus (0\|c^i), i = 0, \ldots, 2^k - 2. \tag{35}$$

This means, that for any non-bijective k-bit function ψ which has no preimage for 0, $\#\mathcal{D}_{\hat{\pi}_\psi} = \#\Big\{\hat{\pi}_\psi(l\|r) \oplus (l\|r)\Big| (l\|r) \in V_{2k}\Big\} < 2^{2k}$. □

So, in order to construct orthomorphisms over V_{2k} it is therefore appropriate to consider the following class of permutations $\dot{\pi}_\psi = \mathcal{I}(r \otimes \psi(l \otimes r))\|(\mathcal{I}(l) \otimes \psi(l \otimes r))$. In this case

$$\begin{aligned}\mathcal{D}_{\dot{\pi}_\psi} &= \Big\{0\Big\} \bigcup \Big\{\mathcal{I}(r \otimes \psi(0))\|r)\Big| r \in V_k^*\Big\} \bigcup \Big\{(l\|(\mathcal{I}(l) \otimes \psi(0))\Big| l \in V_k^*\Big\} \\ &\quad \bigcup \Big\{(\mathcal{I}(r \otimes \psi(l \otimes r))\|(\mathcal{I}(l) \otimes \psi(l \otimes r)) \oplus (l\|r)\Big| l, r \in V_k^*\Big\} \\ &= \mathcal{U}_0 \bigcup \mathcal{U}_1 \bigcup \mathcal{U}_2 \bigcup \mathcal{U}_3,\end{aligned}$$

where $\mathcal{U}_0 = \Big\{0\Big\}, \mathcal{U}_1 = \Big\{\mathcal{I}(r \otimes \psi(0))\|r)\Big| r \in V_k^*\Big\}, \mathcal{U}_2 = \Big\{(l\|(\mathcal{I}(l) \otimes \psi(0))\Big| l \in V_k^*\Big\}, \mathcal{U}_3 = \Big\{(\mathcal{I}(r \otimes \psi(l \otimes r))\|(\mathcal{I}(l) \otimes \psi(l \otimes r)) \oplus (l\|r)\Big| l, r \in V_k^*\Big\}$.

According to the principle of inclusion/exlusion we have

$$\begin{aligned}\#\mathcal{D}_{\dot{\pi}_\psi} &= \sum_{t=0}^{3} \#\mathcal{U}_t - \sum_{0 \le t_1 < t_2 \le 3} \#(\mathcal{U}_{t_1} \cap \mathcal{U}_{t_2}) + \sum_{0 \le t_1 < t_2 < t_3 \le 3} \#(\mathcal{U}_{t_1} \cap \mathcal{U}_{t_2} \cap \mathcal{U}_{t_3}) \\ &\quad - \#(\mathcal{U}_0 \cap \mathcal{U}_1 \cap \mathcal{U}_2 \cap \mathcal{U}_3).\end{aligned}$$

Hence, in order to achieve $\#\mathcal{D}_{\dot{\pi}_\psi} = 2^{2k}$ it is necessary and sufficient to have the fulfillment of the next conditions

1. $\#\mathcal{U}_1 = \#\mathcal{U}_2 = 2^k - 1$, $\#\mathcal{U}_3 = (2^k - 1) \cdot (2^k - 1)$;
2. $\#(\mathcal{U}_{t_1} \cap \mathcal{U}_{t_2}) = \emptyset$, where $0 \leq t_1 < t_2 \leq 3$.

However, the problem of describing all k-bit non-bijective functions ψ which have no preimage for 0 that satisfy simultaneously conditions 1,2 is a difficult task.

In the following proposition the deficit between two instances of $\dot{\pi}$ is bounded.

Proposition 9. *If the lookup-tables of non-bijective k-bit functions $\psi = \begin{pmatrix} \dots & i & \dots \\ \dots & \psi(i) & \dots \end{pmatrix}$, $\hat{\psi} = \begin{pmatrix} \dots & i & \dots \\ \dots & \hat{\psi}(i) & \dots \end{pmatrix}$, differs from each other exactly in one output value, then for permutations $\dot{\pi}_\psi, \dot{\pi}_{\hat{\psi}}$ the following relations holds:*

1. $\mathsf{d}_{\dot{\pi}_\psi} - 2 \cdot (2^k - 1) \leq \mathsf{d}_{\dot{\pi}_{\hat{\psi}}} \leq \mathsf{d}_{\dot{\pi}_\psi} + 2 \cdot (2^k - 1)$, *when $i = 0$;*
2. $\mathsf{d}_{\dot{\pi}_\psi} - 2^k + 1 \leq \mathsf{d}_{\dot{\pi}_{\hat{\psi}}} \leq \mathsf{d}_{\dot{\pi}_\psi} + 2^k - 1$, *when $i \neq 0$.*

Proof. Let prove the first item of the proposition. The set $\mathcal{D}_{\dot{\pi}_{\hat{\psi}}}$ can be written as

$$\begin{aligned} \mathcal{D}_{\dot{\pi}_{\hat{\psi}}} \;=\; & \Big\{0\Big\} \bigcup \Big\{ \mathcal{I}(r \otimes \hat{\psi}(0)) \| r) \Big| r \in V_k^* \Big\} \bigcup \Big\{ (l \| (\mathcal{I}(l) \otimes \hat{\psi}(0)) \Big| l \in V_k^* \Big\} \\ & \bigcup \Big\{ (\mathcal{I}(r \otimes \hat{\psi}(l \otimes r)) \| (\mathcal{I}(l) \otimes \hat{\psi}(l \otimes r)) \oplus (l \| r) \Big| l, r \in V_k^* \Big\}, \end{aligned}$$

From conditions of the proposition, when $i = 0$ we have that $\psi(0) \neq \hat{\psi}(0)$, and $\psi(j) = \hat{\psi}(j)$ for any $j \in \{1, \dots, 2^k - 1\}$. Then

$$\begin{aligned} \mathcal{D}_{\dot{\pi}_{\hat{\psi}}} \;=\; & \Big\{0\Big\} \bigcup \Big\{ \mathcal{I}(r \otimes \hat{\psi}(0)) \| r) \Big| r \in V_k^* \Big\} \bigcup \Big\{ (l \| (\mathcal{I}(l) \otimes \hat{\psi}(0)) \Big| l \in V_k^* \Big\} \\ & \bigcup \Big\{ (\mathcal{I}(r \otimes \psi(l \otimes r)) \| (\mathcal{I}(l) \otimes \psi(l \otimes r)) \oplus (l \| r) \Big| l, r \in V_k^* \Big\}, \end{aligned}$$

where obviously $\#\Big\{ \mathcal{I}(r \otimes \hat{\psi}(0)) \| r) \Big| r \in V_k^* \Big\} = \#\Big\{ (l \| (\mathcal{I}(l) \otimes \hat{\psi}(0)) \Big| l \in V_k^* \Big\} = 2^k - 1$.

As for the set $\mathcal{D}_{\dot{\pi}_\psi}$ the next inclusion is true,

$$\mathcal{D}_{\dot{\pi}_\psi} \;\supseteq\; \Big\{0\Big\} \bigcup \Big\{ (\mathcal{I}(r \otimes \psi(l \otimes r)) \| (\mathcal{I}(l) \otimes \psi(l \otimes r)) \oplus (l \| r) \Big| l \in V_k^*, r \in V_k^* \Big\},$$

then

$$\mathcal{D}_{\dot{\pi}_{\hat{\psi}}} \;\subseteq\; \mathcal{D}_{\dot{\pi}_\psi} \bigcup \Big\{ \mathcal{I}(r \otimes \hat{\psi}(0)) \| r) \Big| r \in V_k^* \Big\} \bigcup \Big\{ (l \| (\mathcal{I}(l) \otimes \hat{\psi}(0)) \Big| l \in V_k^* \Big\}.$$

Hence

$$\#\mathcal{D}_{\dot{\pi}_{\hat{\psi}}} \leq \#\mathcal{D}_{\dot{\pi}_{\psi}} + 2 \cdot (2^k - 1). \tag{36}$$

Analogously for $\mathcal{D}_{\dot{\pi}_{\psi}}$ the following inequality holds

$$\#\mathcal{D}_{\dot{\pi}_{\psi}} \leq \#\mathcal{D}_{\dot{\pi}_{\hat{\psi}}} + 2 \cdot (2^k - 1). \tag{37}$$

So, from (36),(37) we deduce that

$$\mathsf{d}_{\dot{\pi}_{\psi}} - 2 \cdot (2^k - 1) \leq \mathsf{d}_{\dot{\pi}_{\hat{\psi}}} \leq \mathsf{d}_{\dot{\pi}_{\psi}} + 2 \cdot (2^k - 1).$$

Let now prove the second item of the proposition. The set $\mathcal{D}_{\dot{\pi}_{\hat{\psi}}}$ can be decomposed into subsets as follows

$$\begin{aligned}\mathcal{D}_{\dot{\pi}_{\hat{\psi}}} \;=\; & \Big\{0\Big\} \bigcup \Big\{\mathcal{I}(r \otimes \hat{\psi}(0)) \| r) \Big| r \in V_k^*\Big\} \bigcup \Big\{(l \| (\mathcal{I}(l) \otimes \hat{\psi}(0)) \Big| l \in V_k^*\Big\} \\ & \bigcup \Big\{(\mathcal{I}(r \otimes \hat{\psi}(l \otimes r)) \| (\mathcal{I}(l) \otimes \hat{\psi}(l \otimes r)) \oplus (l \| r) \Big| l, r \in V_k^*\Big\}\end{aligned}$$

From conditions of the proposition, when $i \neq 0$ we have $\psi(i) \neq \hat{\psi}(i)$, for some $i \in \{1, \ldots, 2^k - 1\}$ and $\psi(j) = \hat{\psi}(j)$ for any $j \in \{0, 1, \ldots, 2^k - 1\} \setminus \{i\}$. Then

$$\begin{aligned}\mathcal{D}_{\dot{\pi}_{\hat{\psi}}} \;=\; & \Big\{0\Big\} \bigcup \Big\{\mathcal{I}(r \otimes \psi(0)) \| r) \Big| r \in V_k^*\Big\} \bigcup \Big\{(l \| (\mathcal{I}(l) \otimes \psi(0)) \Big| l \in V_k^*\Big\} \\ & \bigcup \Big\{(\mathcal{I}(r \otimes \psi(l \otimes r)) \| (\mathcal{I}(l) \otimes \psi(l \otimes r)) \oplus (l \| r) \Big| l \in V_k^*, r \neq i \otimes l^{-1} \in V_k^*\Big\} \\ & \bigcup \Big\{(\mathcal{I}(r \otimes \hat{\psi}(l \otimes r)) \| (\mathcal{I}(l) \otimes \hat{\psi}(l \otimes r)) \oplus (l \| r) \Big| l \in V_k^*, r = i \otimes l^{-1}\Big\},\end{aligned}$$

and it is not difficult to see that $\#\Big\{(\mathcal{I}(r \otimes \hat{\psi}(l \otimes r)) \| (\mathcal{I}(l) \otimes \hat{\psi}(l \otimes r)) \oplus (l \| r) \Big| l \in V_k^*, r = i \otimes l^{-1}\Big\} \leq 2^k - 1$.

Taking into account that for $\mathcal{D}_{\dot{\pi}_{\psi}}$ the following relation is true,

$$\begin{aligned}\mathcal{D}_{\dot{\pi}_{\psi}} \;\supseteq\; & \Big\{0\Big\} \bigcup \Big\{\mathcal{I}(r \otimes \psi(0)) \| r) \Big| r \in V_k^*\Big\} \bigcup \Big\{(l \| (\mathcal{I}(l) \otimes \psi(0)) \Big| l \in V_k^*\Big\} \\ & \bigcup \Big\{(\mathcal{I}(r \otimes \psi(l \otimes r)) \| (\mathcal{I}(l) \otimes \psi(l \otimes r)) \oplus (l \| r) \Big| l \in V_k^*, r \neq i \otimes l^{-1} \in V_k^*\Big\},\end{aligned}$$

then

$$\mathcal{D}_{\dot{\pi}_{\hat{\psi}}} \;\subseteq\; \mathcal{D}_{\dot{\pi}_{\psi}} \bigcup \Big\{(\mathcal{I}(r \otimes \hat{\psi}(l \otimes r)) \| (\mathcal{I}(l) \otimes \hat{\psi}(l \otimes r)) \oplus (l \| r) \Big| l \in V_k^*, r = i \otimes l^{-1}\Big\},$$

which means

$$\#\mathcal{D}_{\dot{\pi}_{\hat{\psi}}} \leq \#\mathcal{D}_{\dot{\pi}_{\psi}} + 2^k - 1. \tag{38}$$

Analogously for $\mathcal{D}_{\dot{\pi}_{\psi}}$ the following inequality holds

$$\#\mathcal{D}_{\dot{\pi}_{\psi}} \leq \#\mathcal{D}_{\dot{\pi}_{\hat{\psi}}} + 2^k - 1, \tag{39}$$

and thus from relations (38),(39) we obtain

$$\mathsf{d}_{\dot{\pi}_\psi} - 2^k + 1 \leq \mathsf{d}_{\dot{\pi}_{\hat{\psi}}} \leq \mathsf{d}_{\dot{\pi}_\psi} + 2^k - 1.$$

□

Proposition 9 can be used for searching highly-nonlinear orthomorphisms. In order to achieve the property $\mathcal{D}_{\dot{\pi}_\psi} = 2^{2k}$ we have performed a search algorithm similar to algorithm 1. The main task of this algorithm is to decrease the values of the deficit of $\dot{\pi}$ up to zero, which means that an nonlinear orthomorphism is founded. At the same time, according to propositions 4 and 5, it is not difficult to see that the algorithm for searching highly-nonlinear orthomorphisms can also optimize the differential and (non)linear properties of the initial permutation $\dot{\pi}_\psi$. So, we have we have implemented this algorithm (which is omitted due to space limitations) in SAGE [44] obtaining some affine nonequivalent 8-bit orthomorphisms having the following cryptographic parameters

- minimum algebraic degree equal to 7;
- graph algebraic immunity equal to 3 (with 441 equations);
- δ-uniformity equal to 8;
- nonlinearity in range of 100 up to a value of 104.

5 Some concrete S-Boxes, its Pollock representations, column frequency tables and W-intersection matrices

We include in Table 1 some permuations generated by our method, one ordinary permutation with the best founded cryptographic parameters, two involutions and one of the best founded orthomophism.

Recall, that the Linear Approximation Table of an n-bit S-Box Φ, denoted by LAT_Φ, is an matrix over $\mathbb{Z}$ which coefficients are defined as $\mathsf{LAT}_\Phi(a,b) = \frac{1}{2}\mathcal{W}_\Phi(a,b)$.

In [4] the authors suggested looking at the visual representation of the LAT of an S-Box with the goal of finding some unexpected patterns, which can be used in some sense to distinguish it from a random one. The suggested representation is a heatmap of the LAT matrix and was named "a Jackson Pollock representation" of the LAT.

Similarly to [4], in [45] the author illustrate the usefulness of the "Jackson Pollock representation" of the LAT of an S-Box, defining the so-called column frequency table, a tool which can be used to strengthen the effect of some unexpected patterns of a given S-Box.

Definition 14. *Let $\mathcal{A}$ be an $n \times m$ matrix over $\mathbb{Z}$. The column frequency table of $\mathcal{A}$, denoted by $\mathsf{CF}(\mathcal{A})$, is defined as*

$$\mathcal{A}[y,x] = \#\Big\{\hat{y} \in \mathbb{Z} \Big| \mathcal{A}[\hat{y},x] = \mathcal{A}[y,x]\Big\}. \tag{40}$$

S-Box π_1

$\mathcal{NL}(\pi_1) = 104, \delta(\pi_1) = 6, \deg(\pi_1) = 7, \mathcal{AI}_{gr}(\pi_1) = 3, r_{\pi_1^{(3)}} = 441.$

0x6e	0xe8	0x5f	0xa8	0x32	0x24	0xa7	0xe	0x1d	0x64	0x87	0x14	0xc3	0x6f	0x95	0x92
0xfb	0x4c	0x82	0x99	0x3d	0x19	0xac	0x45	0x9f	0xfe	0xde	0x15	0xb9	0xf9	0xe2	0x8a
0xec	0xf5	0xd	0xea	0x3a	0x77	0x47	0x12	0x11	0x1	0x97	0xc5	0x13	0x10	0x81	0x9d
0xed	0x75	0x88	0x68	0xfa	0xa4	0xc0	0xca	0xba	0xb2	0x3b	0x61	0xae	0xa	0x6c	0x65
0xd5	0x42	0x5d	0xdc	0xf2	0x85	0x9b	0xa6	0x67	0x50	0x63	0x91	0xc7	0x34	0x80	0xd7
0x96	0x1b	0x8e	0x5e	0x94	0x2f	0xb1	0xad	0xa0	0x93	0x2c	0x52	0xd0	0x29	0x7	0xc8
0x8d	0x7f	0x49	0x6b	0x36	0x2e	0xd9	0xe0	0x37	0xcd	0x83	0xaf	0x6d	0x57	0xce	0xb3
0x5c	0xc6	0x60	0xd8	0x3f	0xe4	0x4f	0xab	0x56	0xa1	0x72	0xe7	0x69	0xf1	0xdd	0x9c
0x84	0x90	0x25	0x4b	0x76	0x5a	0x6a	0xda	0xf0	0xe5	0x53	0x5b	0x7e	0x2a	0x2b	0xd3
0x35	0xa3	0x1c	0xa2	0x28	0x9e	0x30	0xa9	0xb4	0x6	0xb	0xef	0xaa	0x43	0xe9	0x7d
0xe1	0x3e	0x31	0x44	0x54	0xdb	0x79	0xc9	0x41	0xfc	0xf7	0x66	0x7a	0xb7	0x51	0x38
0xdf	0x62	0x40	0xbb	0x26	0x9	0xf3	0xcf	0xd2	0x1a	0x20	0xc	0x4	0x16	0x33	0x22
0x4e	0xa5	0x58	0x9a	0xd6	0x2	0xe6	0xcb	0xbe	0xeb	0x86	0x7b	0xbd	0xd1	0x3	0xf6
0xee	0x8f	0xf	0x55	0x8b	0x4a	0x7c	0x23	0x2d	0xb6	0x1f	0xc2	0x17	0xbf	0x73	0x8
0xcc	0x70	0x1e	0x59	0x46	0xe3	0x27	0xff	0x78	0xb8	0x18	0x21	0xd4	0xbc	0x98	0xf4
0xc1	0xc4	0x74	0x39	0x89	0xf8	0xfd	0x48	0x71	0x4d	0xb0	0x3c	0x0	0x8c	0xb5	0x5

Involution π_2

$\mathcal{NL}(\pi_2) = 104, \delta(\pi_2) = 6, \deg(\pi_2) = 7, \mathcal{AI}_{gr}(\pi_2) = 3, r_{\pi_2^{(3)}} = 441.$

0x0	0x10	0x90	0xe0	0xd0	0xb0	0x70	0x60	0xf0	0x20	0xc0	0x50	0xa0	0x40	0x30	0x80
0x1	0x11	0x19	0x85	0x2f	0x2c	0x8b	0xf5	0x2e	0x12	0xfa	0x9a	0x8c	0x98	0xfb	0x93
0x9	0x2d	0x4e	0x3c	0x47	0xd5	0x36	0xdc	0x3b	0x29	0xdb	0x46	0x15	0x21	0x18	0x14
0xe	0xb3	0xb8	0x64	0xb4	0x81	0x26	0x3f	0x86	0x6b	0x89	0x28	0x23	0x65	0x3e	0x37
0xd	0x87	0x8a	0x63	0x6c	0x9c	0x2b	0x24	0x66	0x4f	0x96	0x9b	0x83	0x4d	0x22	0x49
0xb	0xad	0x62	0xbe	0x61	0x5c	0xb7	0xa8	0x69	0xb2	0xa3	0x5b	0x55	0xed	0xe1	0xe9
0x7	0x54	0x52	0x43	0x33	0x3d	0x48	0x67	0xc2	0x58	0x6e	0x39	0x44	0xcc	0x6a	0xcb
0x6	0xbd	0xbf	0x7c	0xaa	0xe2	0x76	0xdf	0xa5	0xb9	0xdd	0xe4	0x73	0xee	0xde	0xa9
0xf	0x35	0xc6	0x4c	0xc3	0x13	0x38	0x41	0xc8	0x3a	0x42	0x16	0x1c	0x8e	0x8d	0x8f
0x2	0x94	0x92	0x1f	0x91	0xd7	0x4a	0xe7	0x1d	0xd3	0x1b	0x4b	0x45	0xd6	0xea	0xe5
0xc	0xc1	0xe9	0x5a	0xcd	0x78	0xab	0xf9	0x57	0x7f	0x74	0xa6	0xac	0x51	0xfd	0xff
0x5	0xc4	0x59	0x31	0x34	0xb5	0xcf	0x56	0x32	0x79	0xbb	0xba	0xce	0x71	0x53	0x72
0xa	0xa1	0x68	0x84	0xb1	0xc7	0x82	0xc5	0x88	0xa2	0xca	0x6f	0x6d	0xa4	0xbc	0xb6
0x4	0xf6	0xd8	0x99	0xd4	0x25	0x9d	0x95	0xd2	0xfc	0xfe	0x2a	0x27	0x7a	0x7e	0x77
0x3	0x5e	0x75	0xe3	0x7b	0x9f	0xe8	0x97	0xe6	0x5f	0x9e	0xf1	0xf2	0x5d	0x7d	0xf7
0x8	0xeb	0xec	0xf4	0xf3	0x17	0xd1	0xef	0xf8	0xa7	0x1a	0x1e	0xd9	0xae	0xda	0xaf

Involution $\pi_3^{(invol)}$

$\mathcal{NL}(\pi_3^{(invol)}) = 100, \delta(\pi_3^{(invol)}) = 8, \deg(\pi_3^{(invol)}) = 7, \mathcal{AI}_{gr}(\pi_3^{(invol)}) = 3, r_{\pi_3^{(invol)}}^{(3)} = 441.$

0xa8	0x5b	0x13	0x39	0x56	0x1b	0x66	0x86	0xff	0x1a	0x74	0x3f	0x96	0x2c	0x9f	0x8e
0x17	0x19	0xe6	0x2	0xab	0xbf	0xfb	0x10	0x4c	0x11	0x9	0x5	0x71	0x64	0x59	0xc7
0x67	0x25	0x49	0xbc	0xfc	0x21	0x31	0x9d	0x58	0x45	0xa0	0x8d	0xd	0x9e	0x6e	0xf8
0x93	0x26	0x69	0x5d	0xbb	0x6d	0xa1	0xf9	0x62	0x3	0x3b	0x3a	0xac	0x7b	0x82	0xb
0x54	0x98	0xea	0xaf	0x88	0x29	0x60	0xb3	0xb1	0x22	0xd9	0xcf	0x18	0xa9	0xd8	0x53
0x81	0x76	0xc9	0x4f	0x40	0x8a	0x4	0x89	0x28	0x1e	0x92	0x1	0xb6	0x33	0x7a	0xd7
0x46	0xca	0x38	0xe5	0x1d	0xcd	0x6	0x20	0x90	0x32	0xc0	0xfd	0xb0	0x35	0x2e	0xa4
0xd1	0x1c	0x83	0xf7	0xa	0x91	0x51	0x85	0xd2	0xec	0x5e	0x3d	0x99	0xa7	0xc1	0xa3
0xc5	0x50	0x3e	0x72	0xce	0x77	0x7	0xc3	0x44	0x57	0x55	0xe8	0xb7	0x2b	0xf	0xdd
0x68	0x75	0x5a	0x30	0xe9	0xc8	0xc	0xeb	0x41	0x7c	0xe1	0xc4	0xdf	0x27	0x2d	0xe
0x2a	0x36	0xed	0x7f	0x6f	0xd0	0xd5	0x7d	0x0	0x4d	0xde	0x14	0x3c	0xef	0xb8	0x43
0x6c	0x48	0xf3	0x47	0xfe	0xda	0x5c	0x8c	0xae	0xf2	0xdb	0x34	0x23	0xbe	0xbd	0x15
0x6a	0x7e	0xe0	0x87	0x9b	0x80	0xd3	0x1f	0x95	0x52	0x61	0xd4	0xe4	0x65	0x84	0x4b
0xa5	0x70	0x78	0xc6	0xcb	0xa6	0xfa	0x5f	0x4e	0x4a	0xb5	0xba	0xe7	0x8f	0xaa	0x9c
0xc2	0x9a	0xf6	0xf5	0xcc	0x63	0x12	0xdc	0x8b	0x94	0x42	0x97	0x79	0xa2	0xf0	0xad
0xee	0xf4	0xb9	0xb2	0xf1	0xe3	0xe2	0x73	0x2f	0x37	0xd6	0x16	0x24	0x6b	0xb4	0x8

Orthomorphism π_4

$\mathcal{NL}(\pi_4) = 104, \delta(\pi_4) = 8, \deg(\pi_4) = 7, \mathcal{AI}_{gr}(\pi_4) = 3, r_{\pi_4^{(3)}} = 441.$

0xe1	0x3d	0x2d	0x17	0x51	0x71	0x9b	0x1a	0x96	0xfa	0x64	0x46	0x2f	0x1b	0xe3	0x40
0x1f	0xea	0x12	0xd1	0xa2	0x11	0x5d	0x44	0xb	0xa0	0xaa	0xc9	0x5f	0x58	0xf	0x15
0x5b	0xce	0x49	0x5c	0x7d	0x8a	0xb1	0x2	0x8c	0xcc	0xc8	0xaf	0x56	0xf7	0x4b	0x95
0xa3	0xab	0xcf	0x6f	0xeb	0xd9	0x37	0xdf	0xa8	0x3c	0xbd	0xa4	0x10	0xd7	0xed	0x24
0x29	0x7b	0xe9	0x27	0x22	0x57	0xb6	0xf6	0x79	0x45	0x55	0x82	0xb4	0xc5	0x97	0x69
0x48	0xda	0x2a	0x8f	0x6	0xe2	0x80	0xfe	0xc1	0xf5	0xff	0x3b	0x8d	0x6b	0x85	0xc3
0xdc	0x23	0xca	0x1e	0x5a	0xd	0xf3	0x0	0x81	0xe5	0xc4	0x52	0x32	0x3a	0x1d	0x6d
0x6a	0x77	0x75	0xbe	0xbb	0xb0	0xfc	0xc2	0xf8	0xb2	0x4a	0x9d	0x86	0x4	0x39	0x20
0xd4	0x8	0x4c	0x42	0x94	0xad	0x70	0xa5	0xd5	0x90	0xa7	0xe4	0x3f	0x53	0xde	0x9a
0x54	0xfd	0x2c	0x7	0x5	0x76	0xb8	0xdd	0x7f	0x87	0xa1	0x16	0xd2	0x3e	0xec	0x6e
0x63	0xc	0x50	0x3	0xe8	0x4e	0x73	0x26	0x13	0x4d	0x60	0xa6	0x2e	0xdb	0x8e	0x34
0x5e	0xe7	0xb9	0x1	0xf0	0x72	0xbf	0x25	0x93	0xae	0x62	0x83	0xf4	0xe6	0x47	0x19
0x89	0x7c	0x35	0xa9	0xb5	0x7a	0xf1	0x38	0xef	0x14	0xee	0x1c	0x74	0x31	0xe	0xfb
0xf9	0x91	0x78	0x33	0x18	0xd6	0xd8	0x41	0x9e	0x7e	0xa	0xd3	0x28	0xe0	0xb3	0x21
0x66	0x9f	0xba	0x2b	0x30	0x92	0xbc	0xc6	0x8b	0x6c	0x65	0x68	0x9c	0x4f	0xf2	0x61
0x36	0x84	0xcd	0xc0	0x88	0xc7	0xb7	0x43	0x59	0x98	0xd0	0x99	0x9	0xac	0xcb	0x67

Table 2: Some constructed 8-bit S-Boxes

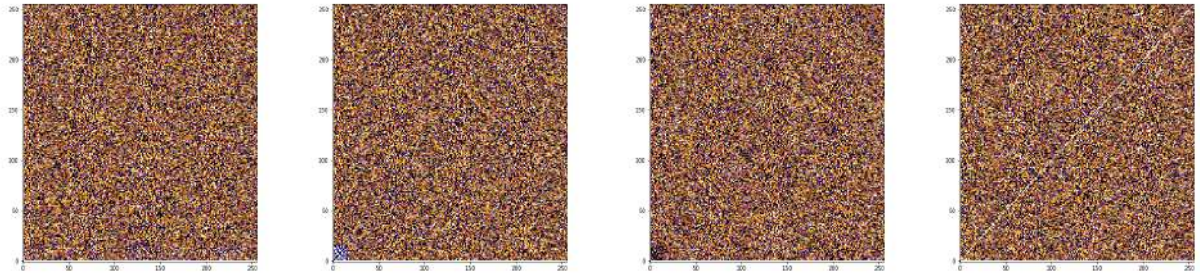


Fig. 3: Pollock representation of the LAT of S-Boxes $\pi_1, \pi_2, \hat{\pi}_3^{(invol)}$ and π_4.

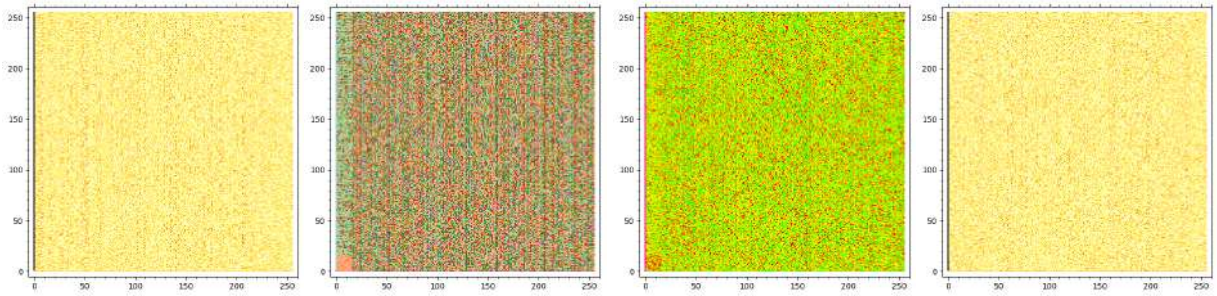


Fig. 4: Column Frequency Tables of the LAT of S-Boxes $\pi_1, \pi_2, \hat{\pi}_3^{(invol)}$ and π_4.

The Pollock representation and column frequency tables of the LAT of S-Boxes π_1,π_2,$\hat{\pi}_3^{(invol)}$ and π_4 listed in Table 1 are shown in Fig. 3 and 4 respectively.

As we can from Fig. 3 and 4 the existence of some visual patterns can not be detected for the S-Box π_1, which is due to the use of some binary linear layers in construction of π_1. If we remove these binary matrices, the S-Box π_1 has some patterns similar to those detected for involutions π_2 and $\hat{\pi}_3^{(invol)}$ (second and third images displayed in Fig. 3 and 4 respectively). The diagonal lines observed in Fig. 3 and 4 respectively for the orthomorphism π_4 is due to the fact that for any orthomorphism $\Phi \in \mathsf{Orth}(V_n)$ the following relation $\mathcal{W}_{\Phi}(a,a) = \mathcal{W}_{\hat{\Phi}}(0,a) = 0$ holds for all $a \in V_n$.

The W-intersection matrices (see, Section 3.3) of nonlinear bijective transformations π_1,π_2,$\hat{\pi}_3^{(invol)}$ and π_4 for subspaces $\mathsf{W}_1 = \{(l\|0)|l \in V_4\}$, $\mathsf{W}_2 = \{(0\|r)|r \in V_4\}$ of the vector space V_8 have the following form

$$\mathcal{M}_{\mathsf{W}_1}(\pi_1) = \begin{pmatrix} 1&2&1&1&0&1&3&0&1&2&2&0&1&0&1&0\\ 0&2&0&1&2&0&0&0&2&2&1&1&0&1&1&3\\ 2&4&0&1&1&0&0&1&1&2&0&0&1&0&2&1\\ 1&0&0&1&0&0&4&1&1&0&2&2&2&0&1&1\\ 0&0&0&1&1&2&2&0&2&2&1&0&1&3&0&1\\ 1&1&3&0&0&2&0&0&1&3&2&1&1&1&0&0\\ 0&0&1&2&1&1&2&1&2&0&1&1&2&1&1&0\\ 0&0&0&1&1&2&2&1&0&1&2&0&1&2&2&1\\ 0&0&3&0&1&3&1&2&1&1&0&0&0&2&1&1\\ 2&1&1&2&1&0&0&1&0&1&4&1&0&0&2&0\\ 0&0&0&3&2&2&1&2&0&0&0&1&1&1&1&2\\ 3&2&3&1&1&0&1&0&0&0&0&1&1&2&0&1\\ 2&0&0&0&1&1&0&1&1&1&1&2&1&2&2&1\\ 2&2&2&0&1&1&0&2&2&0&0&2&1&0&1&0\\ 0&2&2&0&1&1&0&2&0&1&0&2&1&1&1&2\\ 2&0&0&2&2&0&0&2&2&0&0&2&2&0&0&2 \end{pmatrix}, \mathcal{M}_{\mathsf{W}_2}(\pi_1) = \begin{pmatrix} 0&2&0&0&1&2&1&0&0&0&0&1&3&2&3&1\\ 2&0&2&1&1&3&1&0&1&0&0&1&1&0&1&2\\ 2&1&1&0&1&1&0&0&2&1&0&0&1&2&2&2\\ 0&0&1&0&1&1&0&0&3&3&2&3&1&0&1&0\\ 0&0&2&0&2&0&5&0&1&1&2&1&0&1&0&1\\ 0&0&1&1&3&1&0&1&1&2&2&1&0&0&2&1\\ 2&1&0&1&0&0&1&3&0&2&1&1&2&1&0&1\\ 1&0&1&1&0&1&1&0&1&2&2&2&0&1&1&2\\ 2&3&1&0&1&0&1&2&1&0&1&0&0&2&1&1\\ 1&2&1&1&1&1&2&0&1&0&1&1&2&1&0&1\\ 2&0&1&3&0&0&1&3&1&0&0&2&1&0&2&1\\ 0&3&2&0&1&2&1&1&0&0&0&2&2&0&0&2\\ 2&0&0&2&2&0&0&2&0&2&2&0&0&2&2&0\\ 1&2&0&1&1&0&1&2&0&2&0&2&2&0&0&2\\ 1&2&1&3&0&2&0&1&1&1&0&1&1&1&1&0\\ 0&0&2&2&1&2&1&1&3&0&1&0&1&2&0&0 \end{pmatrix}, \mathcal{M}_{\mathsf{W}_1}(\pi_2) = \begin{pmatrix} 1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\ 1&3&3&0&0&0&0&0&3&3&0&0&0&0&0&3\\ 1&3&3&3&0&0&0&0&0&0&0&0&3&0&0&1\\ 1&0&3&3&0&0&3&0&3&0&0&3&0&0&0&0\\ 1&0&3&0&3&0&3&0&3&3&3&0&0&0&0&0\\ 1&0&0&0&0&3&3&0&0&0&0&3&3&0&0&1\\ 1&0&0&3&3&3&3&0&0&0&0&0&0&3&0&0\\ 1&0&0&0&0&0&0&3&0&0&3&3&3&0&1&0\\ 1&3&0&3&3&0&0&0&3&0&0&0&3&0&0&0\\ 1&3&0&0&3&0&0&0&0&3&0&0&0&3&3&0\\ 1&0&0&0&0&3&0&3&0&3&0&3&0&3&0&3\\ 1&0&0&3&0&3&0&3&0&0&0&3&3&0&0&0\\ 1&0&0&0&0&0&3&0&3&0&3&3&3&0&0&0\\ 1&0&3&0&0&0&3&0&3&0&3&0&0&0&3&1\\ 1&0&0&0&0&3&0&3&0&3&0&3&0&0&3&3\\ 1&3&0&0&0&0&0&0&0&0&3&0&0&3&3&3 \end{pmatrix}, \mathcal{M}_{\mathsf{W}_2}(\pi_2) = \begin{pmatrix} 1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\ 1&3&0&1&3&1&1&1&0&0&0&1&0&3&1&0\\ 1&0&3&0&0&1&1&0&3&3&1&0&1&0&1&1\\ 1&1&0&3&3&1&0&0&0&1&1&0&3&0&1&1\\ 1&3&0&3&3&0&0&0&1&0&0&1&1&1&1&1\\ 1&1&1&1&0&3&0&3&1&0&0&0&3&1&0&1\\ 1&1&1&0&0&0&3&1&3&0&1&3&0&1&0&1\\ 1&1&0&0&1&3&1&3&1&1&0&0&1&0&0&3\\ 1&0&3&0&0&1&3&1&3&1&0&1&0&1&1&0\\ 1&0&3&1&0&0&0&1&1&3&1&1&1&0&0&3\\ 1&0&1&1&1&0&1&0&0&1&3&3&0&1&3&0\\ 1&1&0&0&1&0&3&0&1&1&3&3&0&0&1&1\\ 1&0&1&3&1&3&0&1&0&1&0&0&3&1&1&0\\ 1&3&0&0&1&1&1&0&1&0&1&0&1&3&3&0\\ 1&1&1&1&0&0&0&0&1&0&3&1&1&3&3&0\\ 1&0&1&1&1&1&1&3&0&3&0&1&0&0&0&3 \end{pmatrix}$$

$$\mathcal{M}_{\mathsf{W}_1}(\hat{\pi}_3^{(invol)}) = \begin{pmatrix} 0&3&1&2&0&2&1&1&2&2&1&0&0&0&0&1\\ 3&4&0&0&1&1&1&1&0&0&1&1&1&0&1&1\\ 1&0&2&1&2&1&2&0&1&2&1&1&0&0&0&2\\ 2&0&1&2&0&1&3&1&1&1&2&1&0&0&0&1\\ 0&1&2&0&0&2&1&0&1&1&2&2&1&2&1&0\\ 2&1&1&1&2&0&0&0&2&3&1&0&1&1&2&1\\ 1&1&2&3&1&0&1&0&0&0&1&1&2&0&1&1\\ 1&1&0&1&0&2&0&0&2&2&2&0&1&2&1&1\\ 2&0&1&1&1&3&0&2&0&0&0&1&3&1&1&0\\ 2&0&2&1&1&1&1&2&0&0&0&0&2&1&3&0\\ 1&1&1&2&2&0&1&2&0&0&0&1&0&3&2&0\\ 0&1&1&1&2&1&1&0&1&1&0&2&0&2&2&3\\ 0&1&0&0&1&1&3&1&3&2&0&0&0&2&2&0\\ 0&0&0&0&2&1&0&2&1&1&3&2&2&0&1&1\\ 0&1&0&0&1&0&1&1&1&3&2&0&2&1&0&3\\ 1&1&2&1&0&0&1&1&0&0&0&3&0&1&3&2 \end{pmatrix}, \mathcal{M}_{\mathsf{W}_2}(\hat{\pi}_3^{(invol)}) = \begin{pmatrix} 0&2&1&1&1&2&1&2&2&0&2&0&1&0&1&0\\ 2&0&0&0&1&2&3&0&2&1&2&1&1&0&1&0\\ 1&0&0&3&0&0&2&0&2&4&2&0&0&1&1&0\\ 1&0&3&0&0&2&1&3&0&1&0&0&1&1&0&3\\ 1&1&0&0&0&0&1&0&1&1&1&4&2&1&2&1\\ 2&2&0&2&0&0&1&1&1&2&1&0&2&0&1&1\\ 2&0&0&3&0&1&1&0&0&2&0&1&2&2&0&2\\ 2&2&2&0&1&1&0&0&2&0&0&1&1&0&2&2\\ 0&1&4&1&1&1&0&2&0&0&2&0&1&1&1&0\\ 2&2&2&0&1&2&1&0&0&2&0&2&0&0&2&0\\ 0&1&0&0&4&1&1&1&1&0&2&0&0&3&0&2\\ 1&1&0&1&2&0&2&2&1&2&0&0&2&1&0&1\\ 0&0&1&1&1&2&0&2&0&1&0&3&1&0&2&2\\ 0&0&0&3&1&1&0&2&2&0&0&2&1&2&2&0 \end{pmatrix}, \mathcal{M}_{\mathsf{W}_1}(\pi_4) = \begin{pmatrix} 0&3&2&1&2&1&1&1&0&2&0&0&0&0&2&1\\ 2&4&0&0&1&3&0&0&0&0&3&0&1&1&1&0\\ 1&0&0&0&2&3&0&1&2&1&1&1&3&0&0&1\\ 0&1&1&2&0&0&1&0&0&0&4&1&1&3&2&0\\ 0&0&3&0&1&2&1&2&1&1&0&2&1&1&0&1\\ 1&0&1&1&1&0&1&0&4&0&0&0&2&1&1&3\\ 2&1&1&1&0&1&1&2&1&0&1&1&0&2&1&1\\ 1&0&1&2&1&2&1&2&1&1&0&1&0&0&1&1\\ 2&1&2&1&2&1&2&1&1&0&1&0&0&1&1&0\\ 1&1&1&0&1&1&1&1&1&1&2&0&0&2&2&2\\ 1&2&0&3&0&0&0&3&1&0&1&1&0&0&2&2\\ 1&1&2&1&1&0&0&2&0&2&0&2&0&1&3&1\\ 0&0&1&1&1&0&5&0&1&3&0&2&1&0&0&1\\ 1&0&0&1&1&1&1&0&2&2&1&1&4&1&0&0 \end{pmatrix}, \mathcal{M}_{\mathsf{W}_2}(\pi_4) = \begin{pmatrix} 0&1&0&2&2&0&2&0&1&3&1&1&1&0&1&1\\ 0&1&0&1&1&0&0&2&1&0&2&2&2&2&1&1\\ 1&0&1&0&0&2&0&0&1&3&3&0&2&2&0&1\\ 1&2&1&2&0&0&0&3&0&1&0&1&1&0&2&2\\ 2&1&2&0&1&2&2&0&0&1&2&2&0&0&1&2&1\\ 1&2&3&0&0&0&2&2&0&1&2&0&0&1&2&1&0\\ 2&2&0&2&0&0&1&2&2&0&0&1&2&0&1&0&1\\ 1&1&2&2&1&1&2&3&0&1&0&1&0&0&1&1&1\\ 0&2&0&2&0&1&1&0&2&2&0&2&1&0&1&2\\ 2&0&1&0&1&3&0&1&1&0&1&0&3&1&2&0\\ 2&1&1&0&2&2&0&1&1&0&3&0&0&1&1&1\\ 0&0&2&2&2&0&3&0&1&2&0&1&1&0&1&1\\ 1&0&2&0&3&0&2&0&1&1&0&0&1&1&1&3\\ 1&1&0&1&1&1&1&2&1&0&1&3&1&0&1&1\\ 0&0&1&2&0&1&0&2&1&0&2&1&2&3&1&1\\ 2&2&0&1&2&2&0&1&0&2&1&1&0&1&1&0 \end{pmatrix}.$$

As it can be seen, the matrices $\mathcal{M}_{\mathsf{W}_i}(s), i = 1,2$, where $s \in \{\pi_1, \pi_2, \hat{\pi}_3^{(invol)}, \pi_4\}$ does not have any element equal to 16, which confirms that subspaces $\mathsf{W}_1 = \{(l\|0)|l \in V_4\}$, $\mathsf{W}_2 = \{(0\|r)|r \in V_4\}$ of the vector space V_8 are not invariant with respect to the action of these nonlinear bijective transformations.

6 Masking complexity of 8-bit S-Boxes obtained by the scheme of $\hat{\pi}$

In this section we study the possibility of combine ours 8-bit S-Boxes with the classical masking countermeasure against SCAs in terms of its masking complexity. The polynomial representation of an S-Box defined by relation (7) is based on four kinds of operations over GF(2^n): additions, multiplications by constants (scalar multiplications), squares, and nonlinear multiplications (i.e. multiplications of two different variables). Except for the latter, all these operations are linear (resp. affine) over V_n. The processing of any S-Box can then be performed as a sequence of functions which are linear (resp. affine) over V_n (themselves composed of additions, squares and scalar multiplications) and of nonlinear multiplications. Masking an S-Box processing can hence be done by masking every operation mentioned above independently. We recall hereafter the concept of masking complexity introduced in [7] and defined as follows.

Definition 15. *The masking complexity of any n-bit S-Box Φ, denoted by $\mathcal{MC}(\Phi)$, is the minimal number of nonlinear multiplications required to evaluate its polynomial representation over* GF(2^n).

Denoting by $\mathcal{M}_k^n$ as the class of exponents α such that X^α has a masking complexity equal to k we summarizes in Table 3 the results (obtained in [7]) for the cyclotomic classes $C_\alpha = \{\alpha \cdot 2^j \mod (15) \,|\, j = 0, 1, 2, 3.\}$ in $\mathcal{M}_k^4$.

k	Cyclotomic classes in $\mathcal{M}_k^4$
0	$C_0 = \{0\}, C_1 = \{1, 2, 4, 8\}$
1	$C_3 = \{3, 6, 12, 9\}, C_5 = \{5, 10\}$
2	$C_7 = \{7, 11, 13, 14\}$

Table 3: Cyclotomic classes for $n = 4$ w.r.t. the masking complexity k.

Taking into account that the number of field multiplications for any 4-bit permutation and any 4-bit non-bijective function is lower bounded by 0 and upper bounded by 3,4 respectively (see, [7]), we obtain the following bounds for 8-bit S-Boxes produced by our construction:

$$5 \leq \#\text{ nonl. mult. of } \hat{\pi} \leq 12. \tag{41}$$

As we can see from (15), 8-bit S-Boxes with only 5 nonlinear multiplications over GF(2^4) can be constructed using the proposed scheme.

The number of field of multiplications for those involutions obtained by the $\hat{\pi}^{(invol)}$ scheme is given by the following bound $10 \leq$

nonl. mult. of $\hat{\pi}^{(invol)} \leq 24$. As we can see, masking these involutions is more expensive than ordinary S-Boxes produced by the construction of $\hat{\pi}$.

S-Box class	# nonl. multiplications
AES's S-Box [15]	4 (GF(2^8))
AES's S-Box [23]	5 (GF(2^4))
Clefia S-Box [15]	10 (GF(2^8))
Iceberg S-Box [15]	18 (GF(2^4))
Khazad S-Box [15]	18 (GF(2^4))
Picaro S-Box [40]	4 (GF(2^4))
Zorro S-Box [15]	4 (GF(2^4))
S-Boxes based on $\hat{\pi}$ scheme [this work]	$5 \leq$ # nonl. multiplications ≤ 10
S-Boxes based on $\hat{\pi}^{(invol)}$ scheme [this work]	$10 \leq$ # nonl. multiplications ≤ 24

Table 4: Comparison of 8-bit S-Boxes w.r.t. # nonl. multiplications.

Finally, in Table 3 we compare our results with some candidates having a given level of masking. As we can see our S-Boxes based on $\hat{\pi}$ scheme exhibits better values of fields multiplications than S-Boxes of Clefia, Iceberg and Khazad respectively, having at the same time stronger cryptographic properties but at the cost of a worse number of nonlinear multiplications compared with the AES [23], Picaro [40] and Zorro S-Boxes [15].

7 Conclusion and Future Work

In this work we have presented a new algorithmic-algebraic scheme based on the Lai-Massey structure for constructing permutations of dimension $n = 2k, k \geq 2$. Compared to the best nonlinearity (108, for $k = 4$) offered by the construction presented in [9] and latter generalized in [14], the nonlinearity for the permutations obtained by our scheme slightly decrease up to 104, but to the best of our knowledge the schemes presented in [9, 14] can not produce involutions and orthomorphisms with cryptographic properties close to the optimal ones, so we can conclude that the new structure presented in this work is more powerful and attractive due to the diversity of permutations that can be constructed. Interestingly, the involutions and orthomorphisms founded in this work have comparable classical cryptographic properties as those constructed by using spectral-linear and spectral-differential methods [31] and the limited deficit's method [33]. The main advantage of our 8-bit permutations is that they can be constructed using smaller 4-bit components which could be useful for the implementation of the S-Box in hardware or using a bit-sliced approach. The aim of this work was to present a new scheme that can help to find permutations, involutions and orthomophisms with rather good cryptographic properties. There are several questions (more

theoretical results, hardware and bit-sliced implementations, more efficient methods of masking) about the class of permutations suggested in this work which are left as future work.

Acknowledgements. The author is very grateful to Oleg V. Kamlovskiy and the anonymous reviewers of CTCrypt'2020 for their useful comments and valuable observations, which helped to improve the final version of this article.

References

[1] Avanzi R. A Salad of Block Ciphers.: The State of the Art in Block Ciphers and their Analysis. Cryptology ePrint Archive, Report 2017/1171, 2017. http://eprint.iacr.org/2017/1171.

[2] Bannier A., Bodin N„ and Filiol E.: Partition-based trapdoor ciphers. Cryptology ePrint Archive, Report 2016/493, 2016. http://eprint. iacr.org/2016/493.

[3] Bracken, C., Leander,G.: A highly nonlinear differentially 4 uniform power mapping that permutes fields of even degree, Finite Fields and Their Applications 16 (2010) 231–242.

[4] Biryukov, A., Perrin L., and Udovenko A.: Reverse engineering the S-Box of streebog, kuznyechik and STRIBOBr1. In Marc Fischlin and Jean-Sébastien Coron, editors, Advances in Cryptology - EUROCRYPT 2016, Part I, volume 9665 of LNCS, pages 372-402. Springer, Heidelberg, May 2016.

[5] Brier E., Clavier C., and Olivier F.: Correlation Power Analysis with a Leakage Model // Proceedings of Cryptographic Hardware and Embedded Systems 2004, LNCS 2004, p. 157-173, Springer-Verlag, 2004.

[6] Carlet C.: Vectorial Boolean functions for cryptography. Boolean models and methods in mathematics, computer science, and engineering, 134:398–469, 2010.

[7] Carlet C., Goubin L., Prouff E., Quisquater M., and Rivain M.: Higher-order masking schemes for s-boxes. In Anne Canteaut, editor, FSE, volume 7549 of LNCS, pages 366-384. Springer, 2012.

[8] Courtois, N. T., and Pieprzyk, J.: Cryptanalysis of Block Ciphers with Overdefined Systems of Equations, http://eprint.iacr.org/2002/044/,2002.

[9] 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. In: Lange T., Dunkelman O. (eds) Progress in Cryptology – LATINCRYPT 2017. LATINCRYPT 2017. Lecture Notes in Computer Science, vol 11368. Springer, Cham.

[10] Evans A.: Applications of complete mappings and orthomorphisms of finite groups. Quasigroups and relat. syst., 23, 2015, pp. 5–30.

[11] Evans A.: Orthomorphisms graphs and groups, Springer-Verlag, Berlin, 1992.

[12] Gierlichs B., Batina L., Tuyls P. , and Preneel B.: Mutual information analysis // International Workshop Cryptographic Hardware and Embedded System , p. 426-442., 2008.

[13] GOST R 34.12-2015 Information technology. Cryptographic protection of information. Block ciphers. Moscow, Standartinform, 2015.

[14] Fomin D.: New Classes of 8-bit Permutations Based on a Butterfly Structure. In: Preproceedings of CTCrypt'18-Suzdal, Russia, 2016. p.199-211.

[15] Gérard, B., Grosso, V., Naya-Plasencia, M., Standaert, F.X.: Block ciphers that are easier to mask: how far can we go? In: Cryptographic Hardware and Embedded Systems-CHES 2013. Springer (2013) 383-399.

[16] Gilboa Sh., and Gueron Sh.: Balanced permutations Even-Mansour ciphers. Cryptology ePrint Archive, Report 2014/642, 2014.

[17] Gligoroski D., Odegard R.S., Mihova M., et al. Cryptographic Hash Function Edon-R // Proc. IWSCN. Trondheim, 2009, pp. 1-9.

[18] Guilley S., Hoogvorst P. and Pacalet R.: Differential power analysis modeland some results. In CARDIS, pages 127-142, 2004.

[19] Feng D., Feng X., Zhang W., Fan X. and Wu C.: "Loiss: a byte oriented stream cipher," in Coding and Cryptology, vol. 6639 of LNCS, pp. 109–125, Springer, Heidelberg, Germany, 2011.

[20] Hodgers P., Regazzoni F., Gilmore R., Moore C. , Oder T.: Safe Crypto. Secure Architectures of Future State-of-the-Art in Physical Side-Channel Attacks and Resistant Technologies Emerging cryptography. Report by European Commission.

[21] Johnson D.M., Dulmage A.L. and Mendelsohn N.S., "Orthomorphisms of groups and orthogonal Latin squares", I.Canad. J. Math., 13, 1961, pp. 356–372.

[22] Kazymyrov O. V., Kazymyrova V. N., Oliynykov R. V.: A method for generation of high-nonlinear S-boxes based on gradient descent, Mat. Vopr. Kriptogr., 2014, Volume 5, Issue 2, 71–78.

[23] Kim H., S. Hong, and J. Lim.: A fast and provably secure higher-order masking of AES s-box. In B. Preneel and T. Takagi, editors. CHES 2011 - 13th International Workshop, Nara, Japan. Proceedings, volume 6917 of LNCS. Springer, 2011,pages 95-107.

[24] Kim J.: Combined Differential, Linear and Related-Key Attacks on Block Ciphers and MAC Algorithms. IACR Cryptology ePrint Archive (2006), Report 2006/451,http://eprint.iacr.org/2006/451.pdf.

[25] Kocher P., Jaffe J., and Jun B.: Introduction to Differential Power Analysis and Related Attacks Technical Report // Cryptography Research Inc., 1998. Available athttp://www.cryptography.com/resources/whitepapers/DPA-technical.html

[26] Kocher, P.: Timing attacks on implementations of Diffie-Hellman,RSA, DSS and other systems // In Neal Koblitz, editor, Advances in Cryptology - Proceedings of CRYPTO 1996, number 1109 in LNCS, 104-113. Springer-Verlag, 1996.

[27] Knudsen, L. R.: Truncated and Higher Order Differentials. In FSE, B. Preneel,Ed., vol. 1008 of of LNCS. Springer Berlin Heidelberg, 1995, pp. 196-211.

[28] Lachaud G, Wolfmann J. The weights of the orthogonals of the extended quadratic binary Goppa codes. IEEE Trans Inform Theory, 1990, 36: 686–692.

[29] Leander G., Abdelraheem M., Alkhzaimi H., Zenner E., "A cryptanalysis of PRINT cipher: The invariant subspace attack", *CRYPTO'11, Lect. Notes Comput. Sci.*, **6841** (2011), 206–221.

[30] Mann H.B. On orthogonal latin squares, Bulletin of the American Mathematical Society 50, 1944, pp. 249-257.

[31] Menyachikhin A.V.: Spectral-linear and spectral-differential methods for generating S-boxes having almost optimal cryptographic parameters, Mat. Vopr. Kriptogr., 8:2, 2017, pp. 97-116.

[32] Menyachikhin A.V. Method for generating S-boxes using the values of linear and differential spectra and device for its realization. RU Patent № 2633132, Bull. № 29, 2017.

[33] Menyachikhin A.V. Orthomorphisms of abelian groups with minimal pairwise distances. Diskr. Mat., 2018, Vol. 30,issue. 4, pp. 55-65.

[34] Menyachikhin A."The limited deficit's method and the construction problem of orthomorphisms and almost ortomorphisms of abelian group", Diskr. Mat. Mat., 2019, Vol. 31,issue. 3, pp. 58–77.

[35] Menyachikhin A.: "Device for generating orthomorphisms using paired differences". RU Patent 2632119, Bull 28 (2017).

[36] Millan W. L.: Low order approximation of cipher functions. In Cryptography: Policy and Algorithms Conference, Proceedings, volume 1029 of of LNCS, pp. 144-155. Springer Verlag, 1996.

[37] Niederreiter H.:, Robinson K. Bol loops of order pq, Math. Proc. Cambridge Philos. Soc., 1981, pp. 241-256.

[38] Niederreiter H.: Robinson K. Complete mappings of finite fields. J. Austral. Math. Soc. Ser., 1982, pp. 197-212.

[39] NIST. Advanced Encryption Standard. Federal Information Processing Standard (FIPS) 197,November 2001.

[40] Piret G., Roche T„ and Carlet C.: PICARO - a block cipher allowing efficient higher-order side-channel resistance. In Feng Bao, Pierangela Samarati, and Jianying Zhou, editors, ACNS, volume 7341 of Lecture Notes in Computer Science, pages 311-328. Springer, 2012.

[41] Pogorelov B. A., Pudovkina M. A., "On the distance from permutations to imprimitive groups for a fixed system of imprimitivity", Discrete Math. Appl., 24:2 (2014), 95–108.

[42] Prouff E.: DPA Attacks and S-boxes. In FSE, pages 424-441, 2005.

[43] Rijmen V., Preneel B.: A family of trapdoor ciphers // FSE'97. Lect. Notes Comp.Sci. - 1997. - V. 1267. - P. 139 -148.

[44] Sage Mathematics Software (Version 8.1). (2018) http://www.sagemath.org.

[45] Udovenko, A.: Design and Cryptanalysis of symmetric-key algorithms in black and white-box models. `http://hdl.handle.net/10993/39350`.

[46] Vaudenay S., Junod P.: Device and method for encrypting and decrypting a block of data. United States Patent (20040247117), see also "Fox, a New Family of Block Ciphers". `http://crypto.junod.info/sac04a.pdf`, 2004.

[47] Xuejia Lai and James L. Massey.: A proposal for a new block encryption standard. In Ivan Damgard, editor, Advances in Cryptology nïS EUROCRYPT' 90, volume 473 of LNCS, pages 389-404. Springer, Heidelberg, May 1991.

[48] Yan T., Huanguo Zh., Haiqing H.: Using Evolutionary Computation in Construction of Orthomorphism. Procedings of the International Conference on Multimedia Information Networking and Security, IEEE Computer Society, 2009(2):478-481

Metrical Properties of the Set of Bent Functions in View of Duality

Aleksandr Kutsenko[1,2] and Natalia Tokareva[1]

[1]Sobolev Institute of Mathematics, Novosibirsk, Russia
[2]Novosibirsk State University, Novosibirsk, Russia
alexandrkutsenko@bk.ru, tokareva@math.nsc.ru

Abstract

In this work[1] we give a review of metrical properties of the entire set of bent functions and its significant subclasses of self-dual and anti-self-dual bent functions. We give results for iterative contruction of bent functions in $n+2$ variables based on the concatention of four bent functions and consider related open problem proposed by the second author. Criterion of self-duality for bent iterative functions and corollaries on sign functions and constructions of self-dual bent functions are discussed. It is explored that the pair of sets of bent functions and affine functions as well as a pair of sets of self-dual and anti-self-dual bent functions in $n \geqslant 4$ variables is a pair of mutually maximally distant sets that implies metrical duality. The solution to the problems of preserving bentness and anti-self-duality within automorphisms of the set of all Boolean functions is considered.

Keywords: Boolean bent function, self-dual bent function, Hamming distance, metrical regularity, automorphism group, iterative construction

1 Introduction

How much do we know about some cryptographic objects? One way to measure it is to describe what we can do with them. Otherwise to characterize groups of automorphisms of these objects — separately for each object or together while they form some special class. The question about the group of automorphisms of a set in the Boolean cube necessarily leads us to metrical properties of this set. That is why we are very interested in *metrical properties* of distinct cryptographic Boolean functions.

The term "bent function" was introduced by Oscar Rothaus in the 1960s [28]. It is known [36], that at the same time Boolean functions with maximal nonlinearity were also studied in the Soviet Union. The term

[1]The work is supported by Mathematical Center in Akademgorodok under agreement No. 075-15-2019-1613 with the Ministry of Science and Higher Education of the Russian Federation and Laboratory of Cryptography JetBrains Research.

minimal function, which is actually a counterpart of a bent function, was proposed by the Soviet scientists Eliseev and Stepchenkov in 1962. Bent functions have connections with such combinatorial objects as Hadamard matrices and difference sets. Since bent functions have maximum Hamming distance to linear structures and affine functions they deserve attention for practical applications in symmetric cryptography, in particular, for block and stream ciphers. We refer to the survey [5] and monographies of Mesnager [25] and Tokareva [36] for more information concerning known results and open problems related to bent functions. Results regarding the study of metrical properties of the set of bent functions one can find in article [16].

In this paper we give a review on metrical properties of the entire class of bent function $\mathcal{B}_n$ and its important subclasses — self-dual bent functions $\mathrm{SB}^+(n)$ (i.e. functions such that $f = \widetilde{f}$) and anti-self-dual bent functions $\mathrm{SB}^-(n)$ (i.e. functions such that $f \oplus 1 = \widetilde{f}$), where $\widetilde{f}$ is the dual of f. We suppose that the *keys* to the nontrivial and important properties of the class of bent functions are in understanding how does the *duality mapping* $f \to \widetilde{f}$ operate with bent functions. Recall that $\widetilde{\widetilde{f}} = f$ for every bent function f. It is important to note that the duality mapping is the *unique* known isometric mapping of the bent functions into themselves that can not be extended to a typical isometry of the whole set of all Boolean functions that preserves bent functions.

On other hand, the essence of bent functions is expressed in their metrical properties, namely in maximizing distances between them and affine functions. Note that this very idea in more general form is realized in the concept of metrical complement and metrically regular sets. Recall that $\widehat{X}$ is the metrical complement of the set of functions X if it contains all Boolean functions that are on the maximal possible distance from X. The set is metrically regular, if $\widehat{\widehat{X}} = X$. There is a some similarity to the self-duality of bent functions, is not it?

Our attention is drawn to automorphism groups of the sets $\mathcal{B}_n$, $\mathcal{A}_n$, $\mathrm{SB}^+(n)$, $\mathrm{SB}^-(n)$ and their metrical properties. Previously, we established that the set of all bent functions $\mathcal{B}_n$ and the set of all affine functions $\mathcal{A}_n$ form a pair of metrically regular sets, i.e. $\widehat{\widehat{\mathcal{B}_n}} = \widehat{\mathcal{A}_n} = \mathcal{B}_n$. Now we prove the same fact for the classes of self-dual and anti-self-dual functions: they form another such pair of metrically complement functions, i.e. $\widehat{\widehat{\mathrm{SB}^+(n)}} = \widehat{\mathrm{SB}^-(n)} = \mathrm{SB}^+(n)$. In both cases for elements in a pair of metrically regular sets we prove the coincidence of automorphism groups. Thus, $\mathrm{Aut}\,(\mathcal{B}_n) = \mathrm{Aut}\,(\mathcal{A}_n)$ and $\mathrm{Aut}\left(\mathrm{SB}^+(n)\right) = \mathrm{Aut}\left(\mathrm{SB}^-(n)\right)$. Some other cu-

rious properties of bent functions related to their special constructions are discussed in the paper.

The work has the following structure: notation and definitions are in the Section 2. In Section 3 the duality of a bent function is described, including some its important properties and relevant hypothesis (Section 3.1). Some general and metrical properties of the set of bent functions which coincide with their duals, namely self-dual bent functions, are given in Section 3.2. In Section 4 we discuss the iterative construction of bent function in $n+2$ variables based on the concatenation of four bent functions in n variables. The lower bounds on its cardinality and open problem relevant for the set of bent function are in Section 4.1. Criterion of self-duality for bent iterative functions and its corollaries for sign functions together with constructions of self-dual bent functions are discussed in Sections 4.2 and 4.3. In Section 5 the metrical complement of the set of bent functions is studied (Section 5.2) and the results regarding metrical regularity of the set of bent functions and the set of affine functions are given. Metrical complement of the set of (anti-)self-dual bent functions is in Section 5.3. In Section 6 groups of automorphisms of considered sets are studied. The group of automorphisms of the set of bent functions is characterized in Section 6.3 while the (anti-)self-dual case is in Section 6.5. In Section 6.4 we discuss automorphisms of the set of all Boolean functions in n variables which define bijections between sets of self-dual and anti-self-dual bent functions. In Section 6.6 we state the relation between the results from Section 6.5 and preserving of the Rayleigh quotient of a Boolean function.

2 Notation

Let $\mathbb{F}_2^n$ be a space of binary vectors of length n. Denote, following [12], the orthogonal group of index n over the field $\mathbb{F}_2$ as

$$\mathcal{O}_n = \left\{ L \in GL\left(n, \mathbb{F}_2\right) | LL^T = I_n \right\},$$

where L^T denotes the transpose of L and I_n is an identical matrix of order n over the field $\mathbb{F}_2$.

A *Boolean function* f in n variables is a map from $\mathbb{F}_2^n$ to $\mathbb{F}_2$. Its *sign function* is $F(x) = (-1)^{f(x)}$, $x \in \mathbb{F}_2^n$. We will also refer to a sign function as to a vector from the set $\{\pm 1\}^{2^n}$:

$$F = (-1)^f = \left((-1)^{f_0}, (-1)^{f_1}, ..., (-1)^{f_{2^n-1}}\right) \in \{\pm 1\}^{2^n},$$

where $(f_0, f_1, ..., f_{2^n-1}) \in \mathbb{F}_2^{2^n}$ is a truth-table representation of f with arguments given in the lexicographic order. The set of Boolean functions in n variables is denoted by $\mathcal{F}_n$.

The *algebraic normal form* (ANF, Zhegalkin polynomial) of a Boolean function $f \in \mathcal{F}_n$ is defined to be

$$f(x_1, x_2, ..., x_n) = \bigoplus_{(i_1,i_2,...,i_n)\in\mathbb{F}_2^n} a_{i_1 i_2 ... i_n} x_1^{i_1} x_2^{i_2} ... x_n^{i_n},$$

where $a_z \in \mathbb{F}_2$ for any $z \in \mathbb{F}_2^n$ (with the convention $0^0 = 1$). The *algebraic degree* $\deg(f)$ of a Boolean function f is the maximal degree of monomials which occur in its algebraic normal form with nonzero coefficients.

The *Hamming weight* $\mathrm{wt}(x)$ of the vector $x \in \mathbb{F}_2^n$ is the number of nonzero coordinates of x. The *Hamming weight* $\mathrm{wt}(f)$ of the function $f \in \mathcal{F}_n$ is the Hamming weight of its vector of values. The sign $\oplus$ denotes a sum modulo 2. The *Hamming distance* $\mathrm{dist}(f, g)$ between Boolean functions f, g in n variables is a cardinality of the set $\{x \in \mathbb{F}_2^n : f(x) \oplus g(x) = 1\}$. For $x, y \in \mathbb{F}_2^n$ denote $\langle x, y\rangle = \bigoplus_{i=1}^{n} x_i y_i$. Boolean functions in n variables of the form $f(x) = \langle a, x\rangle \oplus a_0, x \in \mathbb{F}_2^n$, where $a_0 \in \mathbb{F}_2, a \in \mathbb{F}_2^n$, are called *affine* functions. The set of affine functions in n variables is denoted by $\mathcal{A}_n$.

A mapping φ of the set of all Boolean functions in n variables to itself is called *isometric* if it preserves the Hamming distance between functions, that is

$$\mathrm{dist}(\varphi(f), \varphi(g)) = \mathrm{dist}(f, g),$$

for any $f, g \in \mathcal{F}_n$.

The *Walsh–Hadamard transform* (WHT) of a Boolean function f in n variables is an integer valued function $W_f : \mathbb{F}_2^n \to \mathbb{Z}$, defined as

$$W_f(y) = \sum_{x\in\mathbb{F}_2^n} (-1)^{f(x)\oplus\langle x,y\rangle}, \quad y \in \mathbb{F}_2^n.$$

A Boolean function f in an even number n of variables is called *bent* if

$$|W_f(y)| = 2^{n/2},$$

for all $y \in \mathbb{F}_2^n$. The set of all bent functions in n variables is denoted by $\mathcal{B}_n$.

3 The dual of a bent function

From the definition of a bent function it follows that for any $y \in \mathbb{F}_2^n$ we have

$$W_f(y) = (-1)^{\widetilde{f}(y)} 2^{n/2},$$

for some $\widetilde{f} \in \mathcal{F}_n$. The Boolean function $\widetilde{f}$ defined above is called the *dual* function of the bent function f. Thus, for any bent function in n variables its dual Boolean function is uniquely defined. The duality of bent functions was introduced by Dillon [10].

3.1 Properties

Some basic known properties of dual functions are the following [5]:

– Every dual function is a bent function;

– If $\widetilde{f}$ is dual to f and $\widetilde{\widetilde{f}}$ is dual to $\widetilde{f}$, then $\widetilde{\widetilde{f}} = f$;

– The mapping $f \to \widetilde{f}$ which acts on the set of bent functions, preserves the Hamming distance.

There is the following connection between the algebraic degrees of a bent function and its dual [13]:

$$n/2 - \deg(f) \geqslant \frac{n/2 - \deg\left(\widetilde{f}\right)}{\deg\left(\widetilde{f}\right) - 1}.$$

Some results obtained for dual functions can be used in proving the results concerning bent functions, in particular, the connection between algebraic normal form (ANF) coefficients of a bent function and its dual, see [7]:

$$\sum_{x \preccurlyeq y} f(x) = 2^{\mathrm{wt}(y)} - 2^{n/2-1} + 2^{\mathrm{wt}(y)-n/2} \sum_{x \preccurlyeq y \oplus 1} \widetilde{f}(x).$$

One of the most important problem in bent functions is to find the number of them. A new approach to this problem was introduced in [32], see Section 4.1, and the following hypothesis was formulated.

Hypothesis (Tokareva, 2011): *any Boolean function in n variables of degree not more than $n/2$ can be represented as the sum of two bent functions in n variables, where $n \geqslant 2$ is an even number.*

The review of partial results regarding this problem and also in favour of the Hypothesis one can find in [34]. It was also proved in [35] that

Theorem 1. *A bent function in $n \geqslant 4$ variables can be represented as the sum of two bent functions in n variables if and only if its dual bent function does.*

So, it follows that the mentioned Hypothesis with the decomposition problem, see Section 4.1, can not be considered separately for a bent function and its dual.

It is worth noting that this hypothesis is a counterpart of the Goldbach's conjecture in number theory unsolved since 1742: any even number $n > 4$ can be represented as the sum of two prime numbers.

Isometric mappings of the set of all Boolean functions in n variables to itself which preserve bentness and the Hamming distance between every bent function and its dual were characterized in [19], namely it was proved that

Theorem 2. *An isometric mapping φ of the set of all Boolean functions in n variables into itself preserves bentness and the Hamming distance between every bent function and its dual if and only if φ has form*

$$f(x) \longrightarrow f\left(L\left(x \oplus c\right)\right) \oplus \langle c, x\rangle \oplus d, \quad x \in \mathbb{F}_2^n,$$

for some $L \in \mathcal{O}_n$, $c \in \mathbb{F}_2^n$, $\mathrm{wt}(c)$ is even, $d \in \mathbb{F}_2$.

3.2 Self-duality

If a bent function f coincides with its dual it is said to be *self-dual*, that is $f = \widetilde{f}$. A bent function which coincides with the negation of its dual is called an *anti-self-dual*, that is $f = \widetilde{f} \oplus 1$. The set of (anti-)self-dual bent functions in n variables, according to [14], is denoted by $\mathrm{SB}^+(n)$ $\left(\mathrm{SB}^-(n)\right)$.

Self-dual bent functions were explored in paper of Carlet et. al. [4] in 2010, where some important properties and constructions were given. All equivalence classes of self-dual bent functions in 2, 4, and 6 variables and all quadratic self-dual bent functions in 8 variables with respect to a restricted form of an affine transformation

$$f(x) \longrightarrow f\left(L\left(x \oplus c\right)\right) \oplus \langle c, x\rangle \oplus d, \quad x \in \mathbb{F}_2^n,$$

where $L \in \mathcal{O}_n$, $c \in \mathbb{F}_2^n$, $\mathrm{wt}(c)$ is even, $d \in \mathbb{F}_2$, which preserves self-duality were also presented. Further, equivalence classes of cubic self-dual bent functions in 8 variables with respect to the mentioned above restricted form of affine transformation one can find in [11]. In [14] a classification of quadratic self-dual bent functions was obtained. The upper bound for the cardinality of the set of self-dual bent functions was given in [15]. In [20, 24] one can find new constructions of self-dual bent functions. A connection of quaternary self-dual bent functions and self-dual bent Boolean functions was shown in [29]. In [18] it was proved that for any $d \in \{2, 3, ..., n/2\}$ there exists a self-dual bent function of algebraic degree d.

In papers [17, 18, 19] metrical properties of the sets of (anti-)self-dual bent functions in n variables were studied. Below we briefly discuss some of them.

Recall that bent functions in $2k$ variables which have a representation

$$f(x,y) = \langle x, \pi(y)\rangle \oplus g(y), \quad x,y \in \mathbb{F}_2^k,$$

where $\pi : \mathbb{F}_2^k \to \mathbb{F}_2^k$ is a permutation and g is a Boolean function in k variables, form the well known *Maiorana–McFarland class* of bent functions [23]. Let the denotion $\mathrm{SB}^+_{\mathcal{M}}(n)$ stands for the set of self-dual Maiorana–McFarland bent functions and $\mathrm{SB}^-_{\mathcal{M}}(n)$ for the set of anti-self-dual ones. Necessary and sufficient conditions of (anti-)self-duality of bent functions from Maiorana–McFarland class are known from [4]. Regarding the spectrum of Hamming distances in [17] the following result was proved.

Theorem 3. *Let* $f, g \in \mathrm{SB}^+_{\mathcal{M}}(n) \cup \mathrm{SB}^-_{\mathcal{M}}(n)$*, then*

$$\mathrm{dist}(f,g) \in \left\{ 2^{n-1}, 2^{n-1}\left(1 \pm \frac{1}{2^r}\right), r = 0,1,...,n/2-1 \right\},$$

Moreover, if either $f, g \in \mathrm{SB}^+_{\mathcal{M}}(n)$ *or* $f, g \in \mathrm{SB}^-_{\mathcal{M}}(n)$*, then all distances except* 2^{n-1} *are attainable, and for any pair* $f \in \mathrm{SB}^+_{\mathcal{M}}(n)$ *and* $g \in \mathrm{SB}^-_{\mathcal{M}}(n)$ *it holds* $\mathrm{dist}(f,g) = 2^{n-1}$*.*

By analysis of the set of distances from Theorem 3 the minimal Hamming distance between considered functions can be obtained:

Corollary 1. *Let* $n \geqslant 4$*, then the minimal Hamming distance between (anti-)self-dual Maiorana–McFarland bent functions is equal to* 2^{n-2}*.*

Moreover, since the minimal Hamming distance between quadratic Boolean functions in n variables (which correspond to codewords of the $\mathrm{RM}(2,n)$ code) is at least 2^{n-2} [21], the following fact holds

Corollary 2. *Let* $n \geqslant 4$*, then the minimal Hamming distance between quadratic bent functions can be attained on (anti-)self-dual Maiorana–McFarland bent functions.*

It is well known that the minimal Hamming distance between bent functions in n variables is equal to $2^{n/2}$, see [16] for instance. In [18] it was proved that this extremal value can be attained on (anti-)self-dual bent functions.

Theorem 4. *Let* $n \geqslant 4$*, then the minimal Hamming distance between distinct (anti-)self-dual bent functions in* n *variables is equal to* $2^{n/2}$*.*

4 Iterative construction $\mathcal{BI}$

Let f_0, f_1, f_2, f_3 be Boolean functions in n variables. Consider a Boolean function g in $n+2$ variables which is defined as

$$g(00, x) = f_0(x), \quad g(01, x) = f_1(x),$$
$$g(10, x) = f_2(x), \quad g(11, x) = f_3(x),$$

where $x \in \mathbb{F}_2^n$.

It is known (Preneel et. al., 1991; see also [1, 32]) that under condition $f_0, f_1, f_2, f_3 \in \mathcal{B}_n$ the mentioned function g is a bent function in $n+2$ variables if and only if

$$\widetilde{f}_0 \oplus \widetilde{f}_1 \oplus \widetilde{f}_2 \oplus \widetilde{f}_3 = 1,$$

that gives the construction of a bent function in $n+2$ variables through the concatenation of vectors of values of four bent functions in n variables [27].

Bent functions which are obtained by this construction, in accordance with [32], are called *bent iterative functions* ($\mathcal{BI}$) and the set of such bent functions in n variables is denoted by $\mathcal{BI}_n$.

In the article [6] the comparison of cardinalities of different known iterative constructions of bent functions in $n \leqslant 10$ variables was presented and the class $\mathcal{BI}$ had the biggest cardinality among them.

According to [1] there exist bent functions from Maiorana–McFarland class [23] and from the class $\mathcal{PS}$ (Partial Spreads) [10] that can not be represented as bent iterative functions. Also from paper [2] on nonnormal bent functions it follows that there exist bent functions in $\mathcal{BI}_n$ that are nonequivalent to Maiorana–McFarland bent functions.

4.1 Lower bounds on the cardinality and related open problem

In paper [32] some possible ways of how to calculate the number of bent iterative functions were shown.

Theorem 5. *For any even $n \geqslant 4$*

$$|\mathcal{BI}_n| = \sum_{f' \in \mathcal{B}_{n-2}} \sum_{f'' \in \mathcal{B}_{n-2}} |(\mathcal{B}_{n-2} \oplus f') \cap (\mathcal{B}_{n-2} \oplus f'')| \,.$$

Denote $X_n = \{f \oplus h | f, h \in \mathcal{B}_n\}$ and consider the system $\{C_f : f \in \mathcal{B}_n\}$ of its subsets defined as $C_f = \mathcal{B}_n \oplus f$. So,

$$X_n = \bigcup_{f \in \mathcal{B}_n} C_f.$$

Let ψ be an element of X_n. The number of subsets C_f that cover ψ, according to [32], is called *multiplicity* of ψ and is denoted by $m(\psi)$. One can notice that if ψ is covered by C_f then it is covered by any set $C_{f'}$, where f' is obtained from f by adding an affine function.

In [32] the exact number of bent iterative functions through the multiplicities was obtained.

Theorem 6. *For any even $n \geqslant 2$*

$$|\mathcal{BI}_{n+2}| = \sum_{\psi \in C_f} m^2(\psi).$$

So, in order to evaluate $|\mathcal{BI}_{n+2}|$ (and then $|\mathcal{B}_{n+2}|$) we have to study the set X_n and the distribution of multiplicities for its elements. Such analysis, as shown in [32], gives the following lower bound.

Theorem 7. *For any even $n \geqslant 2$*

$$\frac{|\mathcal{B}_{n+2}|^4}{|X_n|} \leqslant |\mathcal{BI}_{n+2}| \leqslant |\mathcal{B}_{n+2}|.$$

Thus for calculating the exact number of bent iterative functions one has to study the structure of the set X_n. So, we come to a new problem statement.

Open problem: bent sum decomposition (Tokareva, 2011). *What Boolean functions can be represented as the sum of two bent functions in n variables? How many such representations does a Boolean function admit?*

The related Hypothesis was previuosly mentioned in the Section 3.1.

4.2 Self-dual bent iterative functions

The set of (anti-)self-dual bent functions from $\mathcal{BI}_n$ is further denoted by $\mathrm{SB}^+_{\mathcal{BI}}(n)$ $\left(\mathrm{SB}^-_{\mathcal{BI}}(n)\right)$.

In paper [18] the necessary and sufficient conditions of self-duality of bent iterative functions were studied, namely the following result was obtained.

Theorem 8. *Let $g \in \mathcal{BI}_{n+2}$ then g is self-dual if and only if there exists such pair of functions $g_1, g_2 \in \mathcal{B}_n$ and a function $h \in \mathcal{F}_n$ that:*

$$\begin{aligned}
f_0 &= (g_1 \oplus g_2)\, h \oplus g_1 = \widetilde{g}_2,\\
f_1 &= (g_1 \oplus g_2)\, h \oplus g_2 = \widetilde{g_1 \oplus h},\\
f_2 &= (g_1 \oplus g_2)\, h \oplus g_2 \oplus h = \widetilde{g}_1,\\
f_3 &= (g_1 \oplus g_2)\, h \oplus g_1 \oplus h \oplus 1 = \widetilde{g_2 \oplus h} \oplus 1.
\end{aligned}$$

Remark 1. *It can be proved that the function h is uniquely defined by a pair of bent functions g_1, g_2, namely: $h = g_1 \oplus \widetilde{g}_1 \oplus g_2 \oplus \widetilde{g}_2$.*

By considering constant function h one can immediately obtain two constructions of self-dual bent iterative functions.

Corollary 3. *Functions*

$$f'(y_1, y_2, x) = (y_1 \oplus y_2)\left(f(x) \oplus \widetilde{f}(x)\right) \oplus f(x) \oplus y_1 y_2,$$

$$f''(y_1, y_2, x) = (y_1 \oplus y_2)(\varphi(x) \oplus \omega(x)) \oplus \varphi(x) \oplus \alpha_1 y_1 \oplus \alpha_2 y_2 \oplus y_1 y_2,$$

where

$$y_1, y_2, \alpha_1, \alpha_2 \in \mathbb{F}_2, \alpha_1 \oplus \alpha_2 = 1, x \in \mathbb{F}_2^n,$$
$$f \in \mathcal{B}_n, \varphi \in \mathrm{SB}^+(n), \omega \in \mathrm{SB}^-(n),$$

are self-dual bent functions in $n+2$ variables.

Remark 2. *The first construction from those listed above (for f') was presented in [4] as an example of the construction which uses the indirect sum of bent functions, see [3]. It is worth noting that the second construction (for f'') can also be obtained from indirect sum of bent functions.*

Since these constructions do not intersect, the sum of their cardinalities is a lower bound for the cardinality of the set of self-dual bent iterative functions.

Corollary 4. *It holds*

$$|\mathcal{B}_{n-2}| + \left|\mathrm{SB}^+(n-2)\right|^2 \leqslant \left|\mathrm{SB}^+_{\mathcal{BI}}(n)\right| \leqslant |\mathcal{B}_{n-2}|^2.$$

4.3 The dimension of linear span of sign functions of self-dual bent functions

Let I_n be an 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

$$H_n H_n^T = 2^n I_{2^n}.$$

Denote $\mathcal{H}_n = 2^{-n/2} H_n$. In terms of sign functions the function $f \in \mathcal{F}_n$ is bent if for its sign function F it holds $\mathcal{H}_n F \in \{\pm 1\}^{2^n}$.

Recall that a non-zero vector $v \in \mathbb{C}^n$ is called an *eigenvector* of a square $n \times n$ matrix A attached to the eigenvalue $\lambda \in \mathbb{C}$ if $Av = \lambda v$. A linear span of eigenvectors attached to the eigenvalue λ is called an *eigenspace* associated with λ. Consider a linear mapping $\psi : \mathbb{C}^n \to \mathbb{C}^n$ represented by a $n \times n$ complex matrix A. A *kernel* of ψ is the set

$$\mathrm{Ker}\,(\psi) = \{x \in \mathbb{C}^n | Ax = \mathbf{0} \in \mathbb{C}^n\},$$

where $\mathbf{0}$ is a zero element of the space $\mathbb{C}^n$.

From the definition of self-duality it follows that sign function of any self-dual bent function is the eigenvector of $\mathcal{H}_n$ attached to the eigenvalue 1, that is an element from the subspace $\mathrm{Ker}\,(\mathcal{H}_n - I_{2^n}) = \mathrm{Ker}\left(H_n - 2^{n/2} I_{2^n}\right)$. The same holds for a sign function of any anti-self-dual bent function, which obviously is an eigenvector of $\mathcal{H}_n$ attached to the eigenvalue (-1), that is an element from the subspace $\mathrm{Ker}\,(\mathcal{H}_n + I_{2^n}) = \mathrm{Ker}\left(H_n + 2^{n/2} I_{2^n}\right)$.

In [4] an orthogonal decomposition of $\mathbb{R}^{2^n}$ in eigenspaces of H_n was given:

$$\mathbb{R}^{2^n} = \mathrm{Ker}\left(H_n + 2^{n/2} I_{2^n}\right) \oplus \mathrm{Ker}\left(H_n - 2^{n/2} I_{2^n}\right), \tag{1}$$

where the symbol $\oplus$ denotes a direct sum of subspaces.

It is known that

$$\dim\left(\mathrm{Ker}\left(H_n + 2^{n/2} I_{2^n}\right)\right) = \dim\left(\mathrm{Ker}\left(H_n - 2^{n/2} I_{2^n}\right)\right) = 2^{n-1},$$

where $\dim(V)$ is the dimension of the subspace $V \subseteq \mathbb{R}^{2^n}$. Moreover, from symmetricity of $\mathcal{H}_n$ it follows that the subspaces $\mathrm{Ker}\left(H_n - 2^{n/2} I_{2^n}\right)$ and $\mathrm{Ker}\left(H_n + 2^{n/2} I_{2^n}\right)$ are mutually orthogonal.

In [18] it was proved that within the set of sign functions of self-dual and anti-self-dual bent functions in $n \geqslant 4$ variables there exist basises of the eigenspaces of the matrix $\mathcal{H}_n$ attached to the eigenvalues 1 and (-1) correspondingly.

Theorem 9. *The linear span of sign functions of (anti-)self-dual bent functions in $n \geqslant 4$ variables has dimension 2^{n-1}.*

It is worth notice that the desired basises consist of sign functions of (anti-)self-dual bent iterative functions provided by two constructions from Corollary 3.

5 Metrical complement and regularity

In this section we give results regarding notable metrical property of a subset of Boolean cube called metrical regularity. The sets of affine Boolean

functions and bent functions possess it. The sets of self-dual and anti-self-dual bent functions in $n \geqslant 4$ variables are also mutually maximaly distant. That implies metrical *duality*, in some sence, between the considered pairs of subsets of Boolean functions.

Regarding that some essential and intriguing questions arise: for instance, are there any pairs of metrically regular subsets inside the metrically regular set of bent functions in n variables? If additionally, in order to exclude some trivial cases we consider only the subsets which include functions together with their negations, the maximal Hamming distance from the considered sets is at most 2^{n-1}. Are there any pairs of metrically regular subsets with additional mentioned requirement such that the distance between them is exactly 2^{n-1}, that is to say they are extremal in a manner?

5.1 Definitions

Let $X \subseteq \mathbb{F}_2^n$ be an arbitrary set and let $y \in \mathbb{F}_2^n$ be an arbitrary vector. Define the *distance* 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

$$\mathrm{d}(X) = \max_{y \in \mathbb{F}_2^n} \mathrm{dist}(y, X).$$

In coding theory this number is also known as the *covering radius* of the set X. A vector $z \in \mathbb{F}_2^n$ is called *maximally distant* from a 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 $\widehat{\widehat{X}} = X$. Define, a subset of Boolean functions to be *metrically regular* if the set of corresponding vectors of values is metrically regular [36].

Sets of functions which have maximum distance from partition set functions were studied in [30], it was shown that partition set functions defined by some partition are mutually maximally distant sets. Lower bound on size of the largest metrically regular subset of the Boolean cube was studied in [26].

5.2 The set of bent functions

Further the symbol $\mathrm{GA}(n)$ stands for the affine group. It is well-known that

Proposition 1. *Any isometric mapping of the form*

$$f(x) \longrightarrow f\left(Ax \oplus b\right) \oplus \langle c, x \rangle \oplus d,$$

where $A \in \mathrm{GL}(n)$, $b, c \in \mathbb{F}_2^n$, $d \in \mathbb{F}_2$, *preserves bentness.*

In [33] the following theorem was proved:

Theorem 10. *For each non-affine Boolean function* $h \in \mathcal{F}_n$ *there exists a bent function* $f \in \mathcal{B}_n$ *such that* $f \oplus h$ *is not bent.*

From Proposition 1 and Theorem 10 it follows that the set of bent functions is closed under addition of affine Boolean functions only. This fact implies that the affine functions are precisely all Boolean functions which are at the maximum distance from the class of bent functions. Namely, in [33] it was shown that

Theorem 11. *A Boolean function in* n *variables is*

- *a bent function if and only if it has the maximal possible distance* $2^{n-1} - 2^{n/2-1}$ *to the set of all affine functions, that is it is an element of* $\widehat{\mathcal{A}_n}$*;*
- *an affine function if and only if it has the maximal possible distance* $2^{n-1} - 2^{n/2-1}$ *to the set of all bent functions, that is it is an element of* $\widehat{\mathcal{B}_n}$.

Thus, from the results given in [33] it follows that there exists a *duality*, in some sense, between the definitions of bent functions and affine functions. In particular, we obtain metrical regularity of the sets of affine functions and bent functions.

Corollary 5. *It holds:*

- *the set* $\mathcal{A}_n$ *of all affine Boolean functions in* n *variables is metrically regular;*
- *the set* $\mathcal{B}_n$ *of all bent functions in* n *variables is metrically regular.*

5.3 The set of (anti-)self-dual bent functions

Since for any self-dual Boolean function $f \in \mathrm{SB}^+(n)$ its negation $f \oplus 1$ is also self-dual, the maximal Hamming distance from the set $\mathrm{SB}^+(n)$ is at most 2^{n-1}. It was proved by Carlet et. al. in [4] that the Hamming distance between any pair of self-dual and anti-self-dual bent functions, both in n variables, is equal to 2^{n-1}. From that it follows that

$$\mathrm{d}\left(\mathrm{SB}^+(n)\right) = 2^{n-1},$$

and all anti-self-dual bent functions in n variables belong to the metrical complement of the set of self-dual bent functions in n variables.

In paper [18] the metrical complement of the set of (anti-)self-dual bent functions in $n \geqslant 4$ variables was completely characterized by using the orthogonal decomposition (1) and the existence of the basis provided by the Theorem 9, namely it was proven that

Theorem 12. *Let $n \geqslant 4$, then the following statements hold:*

- *The metrical complement of the set of self-dual bent functions coincides with the set of anti-self-dual bent functions;*
- *The metrical complement of the set of anti-self-dual bent functions coincides with the set of self-dual bent functions.*

As for the pair of the sets of bent functions and affine functions, it follows that there exists a *duality*, in some sense, between the sets of self-dual and anti-self-dual bent functions in $n \geqslant 4$ variables.

The case $n = 2$ was considered explicitely and it appeared that both $\mathrm{SB}^+(2)$ and $\mathrm{SB}^-(2)$ are metrically regular sets. From that and the Theorem 12 it follows

Theorem 13. *The sets $\mathrm{SB}^+(n)$, $\mathrm{SB}^-(n)$ are metrically regular sets, both with covering radius 2^{n-1}.*

6 The group of automorphisms

Study of automorphism groups of mathematical objects deserves attention since these groups are closely connected with the structure of the objects. There exists a natural question: how groups of automorphisms of two mathematical objects, one of which is embedded to another one, are related.

An example of such a problem statement is the set of bent functions in n variables and one of its significant subclasses which consisits of self-dual bent functions in n variables.

It is also worth mentioning that the complexity of classification of combinatorial objects depends on generality of the approach. Consequently, the question '*if the common approach to classify (self-dual) bent functions is the most general within automorphisms of the set of Boolean functions*', arises naturally.

6.1 Isometric mappings and automorphism groups

Recall that a mapping φ of the set of all Boolean functions in n variables to itself is called *isometric* if it preserves the Hamming distance between

functions, that is

$$\operatorname{dist}(\varphi(f), \varphi(g)) = \operatorname{dist}(f, g),$$

for any $f, g \in \mathcal{F}_n$. Following [19] denote the set of all isometric mappings of the set of all Boolean functions in n variables to itself by $\mathcal{I}_n$.

It is known (A. A. Markov, 1956) that every isometric mapping of all Boolean functions in n variables to itself has the unique representation of the form

$$f(x) \longrightarrow f(\pi(x)) \oplus g(x), \tag{2}$$

where π is a permutation on the set $\mathbb{F}_2^n$ and $g \in \mathcal{F}_n$ [22]. The mapping of this form is denoted by $\varphi_{\pi,g} \in \mathcal{I}_n$.

The *group of automorphisms* of a fixed subset $M \subseteq \mathcal{F}_n$ is the group of isometric mappings of the set of all Boolean functions in n variables to itself preserving the set M. It is denoted by $\operatorname{Aut}(M)$.

6.2 Matrix representation

For a number $k \in \{0, 1, ..., 2^n - 1\}$ denote by $\mathbf{v}_k \in \mathbb{F}_2^n$ its binary representation.

Recall that a square matrix is called *monomial* (or *generalized permutation matrix*) if it has exactly one nonzero entry in each row and each column.

There is an one-to-one correspondence between the set $\mathcal{I}_n$ and the set of monomial matrices of order $2^n \times 2^n$ with nonzero elements from the set $\{\pm 1\}$. Indeed, consider an arbitrary mapping $\varphi_{\pi,g} \in \mathcal{I}_n$. Then for any $f \in \mathcal{F}_n$ and its sign function

$$F = \left((-1)^{f(\mathbf{v}_0)}, (-1)^{f(\mathbf{v}_1)}, ..., (-1)^{f(\mathbf{v}_{2^n-1})}\right) \in \{\pm 1\}^{2^n},$$

the sign function

$$F' = \left((-1)^{f'(\mathbf{v}_0)}, (-1)^{f'(\mathbf{v}_1)}, ..., (-1)^{f'(\mathbf{v}_{2^n-1})}\right) \in \{\pm 1\}^{2^n},$$

of $f' = \varphi_{\pi,g}(f) \in \mathcal{F}_n$ can be expressed as $F' = AF$, where A is a $2^n \times 2^n$ monomial matrix, constructed by the permutation π and the function g:

$$i \begin{pmatrix} & & & j & & & \\ & & & \vdots & & & \\ & & & 0 & & & \\ & & & \vdots & & & \\ \dots & 0 & \dots & (-1)^{g(\mathbf{v}_{i-1})} & \dots & 0 & \dots \\ & & & \vdots & & & \\ & & & 0 & & & \\ & & & \vdots & & & \end{pmatrix},$$

in which in the i-th row a nonzero element $(-1)^{g(\mathbf{v}_{i-1})}$ is in the j-th column, where $(j-1)$ is a number with binary representation $\pi(\mathbf{v}_{i-1})$. So the i-th component of $F' = AF$ is equal to

$$(-1)^{f'(\mathbf{v}_{i-1})} = (-1)^{f(\pi(\mathbf{v}_{i-1}))} \cdot (-1)^{g(\mathbf{v}_{i-1})} = (-1)^{f(\pi(\mathbf{v}_{i-1}))\oplus g(\mathbf{v}_{i-1})},$$

for any $i \in \{1, 2, ..., 2^n\}$, that is equivalent to

$$f'(x) = f(\pi(x)) \oplus g(x), \quad x \in \mathbb{F}_2^n.$$

6.3 The group of automorphisms of the set of bent functions

Some attempts to determine the automorphism group of a given bent function were undertaken by Dempwolff [9] in 2006. Results were presented in terms of elementary Abelian Hadamard difference sets (equivalently, bent functions).

A natural question whether there exist isometric mappings of Boolean functions into itself, distinct from those mentioned in Proposition 1, which preserve the class of bent function was completely solved in paper [31], where it was proved that there were no other mappings possessing such a property. Namely by using the Theorem 11 in view of the duality the following coincidence was shown.

Theorem 14.

$$\mathrm{Aut}(\mathcal{B}_n) = \mathrm{Aut}(\mathcal{A}_n).$$

Note that the set of all affine functions in n variables forms a group isomorphic to $\mathbb{F}_2^{n+1}$. The group of automorphisms of the set of all affine functions in n variables consists, as it is well known, of mappings of the form (2) with affine permutation π and affine shift g, see, for example, [21]. So, the result is formulated as follows.

Theorem 15. *It holds*

$$\mathrm{Aut}(\mathcal{B}_n) = \mathrm{GA}(n) \ltimes \mathbb{F}_2^{n+1},$$

where the symbol $\ltimes$ *for semidirect product.*

These results imply the non-existence of a more general approach to equivalence of bent functions than that on the base of isometric mappings.

6.4 Isometric bijections between self-dual and anti-self-dual bent functions

It is known [4] that there exists a bijection between $\mathrm{SB}^+(n)$ and $\mathrm{SB}^-(n)$, based on the decomposition of sign functions of (anti-)self-dual bent functions. Also note that from the existence of such bijection it follows that $\left|\mathrm{SB}^+(n)\right| = \left|\mathrm{SB}^-(n)\right|$.

Namely, let $(Y, Z) \in \{\pm 1\}^{2^n}$, where $Y, Z \in \{\pm 1\}^{2^{n-1}}$, be a sign function for some $f \in \mathrm{SB}^+(n)$. Then a vector $(Z, -Y) \in \{\pm 1\}^{2^n}$ is a sign function for some function from $\mathrm{SB}^-(n)$. In terms of isometric mappings the mentioned transformation can be represented as

$$f(x) \longrightarrow f\left(x \oplus c\right) \oplus \langle c, x \rangle \,,$$

where $c = (1, 0, 0, ..., 0) \in \mathbb{F}_2^n$.

In paper [14] it was mentioned that the more general form of this mapping

$$f(x) \longrightarrow f\left(x \oplus c\right) \oplus \langle c, x \rangle \,,$$

where $c \in \mathbb{F}_2^n$, $\mathrm{wt}(c)$ is odd, is a bijection between $\mathrm{SB}^+(n)$ and $\mathrm{SB}^-(n)$. It is obvious that this mapping is an element from $\mathcal{I}_n$.

In paper [19] these results were generalized within isometric mappings from the set $\mathcal{I}_n$ for $n \geqslant 4$.

The criterion of bijectivity between self-dual and anti-self-dual bent functions was obtained in [19] with a use of the orthogonal decomposition (1) and the basis from the Theorem 9.

Theorem 16. *Let $n \geqslant 4$, then isometric mapping $\varphi_{\pi,g} \in \mathcal{I}_n$ with matrix A is a bijection between $\mathrm{SB}^+(n)$ and $\mathrm{SB}^-(n)$ if and only if $A\mathcal{H}_n = -\mathcal{H}_n A$.*

By using this criterion in [19] the general form of considered isometric bijections was found.

Theorem 17. *For $n \geqslant 4$ isometric mapping $\varphi_{\pi,g} \in \mathcal{I}_n$ is a bijection between $\mathrm{SB}^+(n)$ and $\mathrm{SB}^-(n)$ if and only if*

$$\pi(x) = L\left(x \oplus c\right), \ x \in \mathbb{F}_2^n,$$

and

$$g(x) = \langle c, x \rangle \oplus d, \ x \in \mathbb{F}_2^n,$$

where $L \in \mathcal{O}_n$, $c \in \mathbb{F}_2^n$, $\mathrm{wt}(c)$ is odd, $d \in \mathbb{F}_2$.

6.5 The group of automorphisms of the set of (anti-)self-dual bent functions

In [4] the followng problem was pointed:

Open question (Carlet, Danielson, Parker, Solé, 2010): *to find mappings preserving self-duality, distinct from the known ones, or give a proof that there are no more.*

In paper [19] this question was resolved within isometric mappings of the set of all Boolean functions in $n \geqslant 4$ variables into itself.

At first the problem of how the sets of isometric mapping preserving self-duality and anti-self-duality or, in other words, groups of automorphisms of the sets $\mathrm{SB}^+(n)$ and $\mathrm{SB}^-(n)$ are related. This problem was solved in [19], where with a use of the orthogonal decomposition (1) and the basis from the Theorem 9, the criterion of preserving self-duality was given.

Theorem 18. *Let $n \geqslant 4$, then for isometric mapping $\varphi_{\pi,g} \in \mathcal{I}_n$ with matrix A the following conditions are equivalent:*

1) $\varphi_{\pi,g}$ preserves self-duality;

2) $\varphi_{\pi,g}$ preserves anti-self-duality;

3) $A\mathcal{H}_n = \mathcal{H}_n A$.

From this result it follows that

Corollary 6. *For $n \geqslant 4$ it holds* $\mathrm{Aut}\left(\mathrm{SB}^+(n)\right) = \mathrm{Aut}\left(\mathrm{SB}^-(n)\right)$.

The problem of characterizing mappings which preserve self-duality was studied by Carlet et. al. in [4] and Feulner et. al. in [11], where it was shown that the mapping

$$f(x) \longrightarrow f\left(L\left(x \oplus c\right)\right) \oplus \langle c, x\rangle \oplus d,$$

where $L \in \mathcal{O}_n$, $c \in \mathbb{F}_2^n$, $\mathrm{wt}(c)$ is even, $d \in \mathbb{F}_2$, preserves self-duality of a bent function. It is obvious that this mapping is isometric and corresponds to $\varphi_{\pi,g} \in \mathcal{I}_n$ with

$$\pi(x) = L\left(x \oplus c\right),\ x \in \mathbb{F}_2^n,$$

and

$$g(x) = \langle c, x\rangle \oplus d,\ x \in \mathbb{F}_2^n,$$

where $L \in \mathcal{O}_n$, $c \in \mathbb{F}_2^n$, $\mathrm{wt}(c)$ is even, $d \in \mathbb{F}_2$. The group which consists of mappings of such form is called an *extended orthogonal group* and denoted by $\overline{\mathcal{O}}_n$ [8, 11]. It is known that this group is a subgroup of $\mathrm{GL}\left(n+2, \mathbb{F}_2\right)$ [11].

In paper [19] known results were generalized within isometric mappings from the set $\mathcal{I}_n$ for $n \geqslant 4$. Namely by using the criterion from Theorem 18 and the matrix representation of isometric mappings it was obtained that the desired group of automorphisms coincides with the extended orthogonal group.

Theorem 19. *For $n \geqslant 4$ it holds*

$$\mathrm{Aut}\left(\mathrm{SB}^+(n)\right) = \mathrm{Aut}\left(\mathrm{SB}^-(n)\right) = \overline{\mathcal{O}}_n.$$

In view of Theorems 17 and 19 it appeares that bijections and mappings which preserve self-duality are quite similar except the parity of the vector $c \in \mathbb{F}_2^n$, which 'switches' them in some sence.

It follows that the classification of self-dual bent functions in $n \geqslant 4$ variables based on the restricted form of affine equivalence proposed in articles [4, 11] is the most general within isometric mappings of the set of all Boolean functions in n variables into itself.

6.6 Isometric mappings and the Rayleigh quotient

In [4] the *Rayleigh quotient* S_f of a Boolean function $f \in \mathcal{F}_n$ was defined as

$$S_f = \sum_{x,y \in \mathbb{F}_2^n} (-1)^{f(x) \oplus f(y) \oplus \langle x,y \rangle} = \sum_{y \in \mathbb{F}_2^n} (-1)^{f(y)} W_f(y).$$

In a scope of bent functions the Rayleigh quotient characterizes the Hamming distance between a bent function and its dual. Indeed, let $f \in \mathcal{B}_n$, then

$$\mathrm{dist}(f, \widetilde{f}) = 2^{n-1} - \frac{1}{2^{n/2+1}} S_f = 2^{n-1} - \frac{1}{2} N_f.$$

In [4] it was proved that for any $f \in \mathcal{F}_n$ the absolute value of S_f is at most $2^{3n/2}$ with equality if and only if f is self-dual $\left(+2^{3n/2}\right)$ and anti-self-dual $\left(-2^{3n/2}\right)$ bent function. That is the maximum (minimum) value of the Rayleigh quotient of a Boolean function in an even number of variables is attainable on self-dual (anti-self-dual) bent functions and only them, thus providing a criterion for (anti-)self-duality in terms of the Rayleigh quotient values.

In article [8] the operations on Boolean functions that preserve bentness and the Rayleigh quotient were given. Namely, it was proved that for any $f \in \mathcal{B}_n, L \in \mathcal{O}_n, c \in \mathbb{F}_2^n, d \in \mathbb{F}_2$ the functions $g, h \in \mathcal{B}_n$ defined as $g(x) = f(Lx) \oplus d$ and $h(x) = f(x \oplus c) \oplus \langle c, x \rangle$ provide $N_g = N_f$ and $N_h = (-1)^{\langle c,c \rangle} N_f$.

The mentioned operations are isometric mappings from $\mathcal{I}_n$. The complete characterization of isometric mappings that preserve the Rayleigh quotient as well as change it was given in [19].

Theorem 20. *If $n \geqslant 4$ then isometric mapping $\varphi_{\pi,g} \in \mathcal{I}_n$ preserves the Rayleigh quotient if and only if it preserves self-duality.*

Theorem 21. *If $n \geqslant 4$ then isometric mapping $\varphi_{\pi,g} \in \mathcal{I}_n$ changes the sign of the Rayleigh quotient if and only if it is a bijection between $\mathrm{SB}^+(n)$ and $\mathrm{SB}^-(n)$.*

7 Conclusion

In this work, we have given a review of metrical properties of the set of bent functions and its subset of functions which coincide with their duals. The group of automorphisms and metrical complements of these sets are described. We also reviewed some general metrical properties of the set of self-dual bent functions and considered an iterative construction of bent functions. Some relevant open problems and hypothesis on bent functions weere discussed.

An interesting question is the characterization of isometric mappings preserving bentness and self-duality, that are beyond the automorphisms of the set of all Boolean functions.

The solution of the problems, that were considered in this review, with regard to different generalizations of bent functions that is study of metrical properties and the duality as well as self-duality in this scope is a goal worth pursuing.

References

[1] Canteaut A., Charpin P., "Decomposing bent functions", *IEEE Trans. Inform. Theory*, **49**:8 (2003), 2004–2019.

[2] Canteaut A., Daum M., Dobertin H., Leander G., "Finding nonnormal bent functions", *Discrete Appl. Math.*, **154**:2 (2006), 202–218.

[3] Carlet C., "Boolean functions for cryptography and error correcting code", *In: Crama Y., Hammer P.L. (eds.) Boolean Models and Methods in Mathematics, Computer Science, and Engineering. Cambridge University Press, Cambridge*, 2010, 257–397.

[4] Carlet C., Danielson L.E., Parker M.G., Solé. P., "Self-dual bent functions", *Int. J. Inform. Coding Theory*, **1** (2010), 384–399.

[5] Carlet C., Mesnager S., "Four decades of research on bent functions", *Des. Codes Cryptogr.*, **78**:1 (2016), 5–50.

[6] Climent J.-J., Garcia F.J., Requena V., "A construction of bent functions of $n+2$ variables from a bent function of n variables and its cyclic shifts", *Algebra*, **2014** (2014).

[7] Cusick T.W., Stănică P., *Cryptographic Boolean functions and applications*, Acad. Press, London, 2017, 288.

[8] Danielsen L.E., Parker M.G., Solé P., "The Rayleigh quotient of bent functions", *Springer Lect. Notes in Comp. Sci.*, **5921** (2009), 418–432.

[9] Dempwolff U., "Automorphisms and Equivalence of Bent Functions and of Difference Sets in Elementary Abelian 2-Groups", *Commun. Algebra*, **34**:3 (2006), 1077–1131.

[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] Janusz G.J., "Parametrization of self-dual codes by orthogonal matrices", *Finite Fields Appl.*, **13**:3 (2007), 450–491.

[13] Hou X.-D., "New Constructions of Bent Functions", *Journal of Combinatorics, Information and System Sciences*, International Conference on Combinatorics, Information Theory and Statistics, **25**, 2000, 173–189.

[14] Hou X.-D., "Classification of self dual quadratic bent functions", *Des. Codes Cryptogr.*, **63**:2 (2012), 183–198.

[15] 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.

[16] Kolomeec N.,, "The Graph of Minimal Distances of Bent Functions and Its Properties", *Des. Codes Cryptogr.*, **85**:3 (2017), 1–16.

[17] Kutsenko A.V., "The Hamming Distance Spectrum Between Self-Dual Maiorana-McFarland Bent Functions", *Journal of Applied and Industrial Mathematics*, **12**:1 (2018), 112–125.

[18] Kutsenko A., "Metrical properties of self-dual bent functions", *Des. Codes Cryptogr.*, **88**:1 (2020), 201–222.

[19] Kutsenko A., "The group of automorphisms of the set of self-dual bent functions", *Cryptogr. Commun.*, 2020, DOI: 10.1007/s12095-020-00438-y.

[20] Luo G., Cao X., Mesnager S., "Several new classes of self-dual bent functions derived from involutions", *Cryptogr. Commun.*, **11**:6 (2019).

[21] MacWilliams F. J., Sloane N. J. A., "The Theory of Error-Correcting Codes", *Amsterdam, New York, Oxford: North-Holland*, 1983, 782.

[22] Markov A. A., "On transformations without error propagation", *Selected Works, Vol. II: Theory of Algorithms and Constructive Mathematics. Mathematical Logic. Informatics and Related Topics, MTsNMO, Moscow*, 2003, 70–93, In Russian.

[23] McFarland R. L., "A family of difference sets in non-cyclic groups", *J. Combin. Theory Ser. A*, **15**:1 (1973), 1–10.

[24] Mesnager S., "Several New Infinite Families of Bent Functions and Their Duals", *IEEE Trans. Inf. Theory*, **60**:7 (2014), 4397–4407.

[25] Mesnager S., *Bent Functions: Fundamentals and Results,*, Springer, Berlin, 2016, 544 p.

[26] Oblaukhov A., "A lower bound on the size of the largest metrically regular subset of the Boolean cube", *Cryptogr. Commun.*, **11**:4 (2019), 777–791.

[27] Preneel B., Van Leekwijck W., Van Linden L., Govaerts R., Vandewalle J., "Propagation characteristics of Boolean functions", *Advances in Cryptology-EUROCRYPT, Lecture Notes in Computer Science*, **473**, Springer, Berlin, Heidelberg, 1990, 161–173.

[28] Rothaus O.S., "On bent functions", *J. Combin. Theory. Ser. A*, **20**:3 (1976), 300–305.

[29] Sok L., Shi M., Solé. P., "Classification and Construction of quaternary self-dual bent functions", *Cryptogr. Commun.*, **10**:2 (2018), 277–289.

[30] Stănică P., Sasao T., Butler J.T.,, "Distance duality on some classes of Boolean functions", *J. Combin. Math. and Combin. Computing*, **107** (2018), 181–198.

[31] Tokareva N.N.,, "The group of automorphisms of the set of bent functions", *Discrete Mathematics and Applications*, **20**:5 (2010), 655–664.

[32] Tokareva N.N.,, "On the number of bent functions from iterative constructions: lower bounds", *Adv. Math. Commun.*, **5**:4 (2011), 609–621.

[33] Tokareva N., "Duality between bent functions and affine functions", *Discrete Math.*, **312**:3 (2012), 666–670.

[34] Tokareva N.N.,, "On decomposition of a Boolean function into sum of bent functions", *Siberian Electronic Mathematical Reports*, **11** (2014), 745–751.

[35] Tokareva N.N.,, "On Decomposition of a Dual Bent Function into Sum of Two Bent Functions", *Prikl. Diskretn. Mat.*, **26**:4 (2014), 59–61, In Russian.

[36] Tokareva N., *Bent Functions, Results and Applications to Cryptography*, Acad. Press. Elsevier, 2015, 230 p.

Extending AES Improvements: A Proposal for Alpha-MAC in View of Collision Resistance

Adrián Alfonso Peñate and Pablo Freyre Arrozarena

Institute of Cryptography, University of Havana, Cuba
yamilkate@infomed.sld.cu, pfreyre@matcom.uh.cu

Abstract

Alred is a kind of construction for Message Authentication Codes based on a block cipher, and one specific instance of the same one, with AES as underlying primitive, result in the MAC function alpha-MAC. In this work we compute the successful probability of one of the applied attacks against alpha-MAC, when the transformation ShiftRows is replaced by a random diffusion optimal permutation.

Keywords: Alpha-MAC, dynamic AES, random ShiftRows.

1 Introduction

MAC functions are symmetric primitives, used to ensure authenticity of messages, taking as input a secret key and the message and producing as output a short tag. The MAC function alpha-MAC (an specific instance of the alred construction) is presented in [1] with AES as underlying primitive, because this block cipher is efficient in hardware and software, and also it has been proved strong since its publication as Rijndael [2]. A better description of alpha-MAC will be presented in section 2.

In this paper we will study the attack against alpha-MAC presented in [3] where the internal structure of this MAC function is analyzed, and based on the algebraic properties of the algorithm AES, internal collisions are obtained. The reason we choose this attack is its simplicity and effectiveness against alpha-MAC, once we know an internal state. In another attacks performed against alpha-MAC the recovery of the internal state has been successful [4, 5, 6, 7]. A better description of this attack will be presented in section 3.

On the other hand, in recent papers [8, 9, 10, 11, 12, 13, 14, 15, 16] new MAC functions are designed and existed MAC functions are improved, or new attacks for MAC functions are proposed [17, 18, 19, 20, 21], showing that the study of these symmetric primitives is an important subject today.

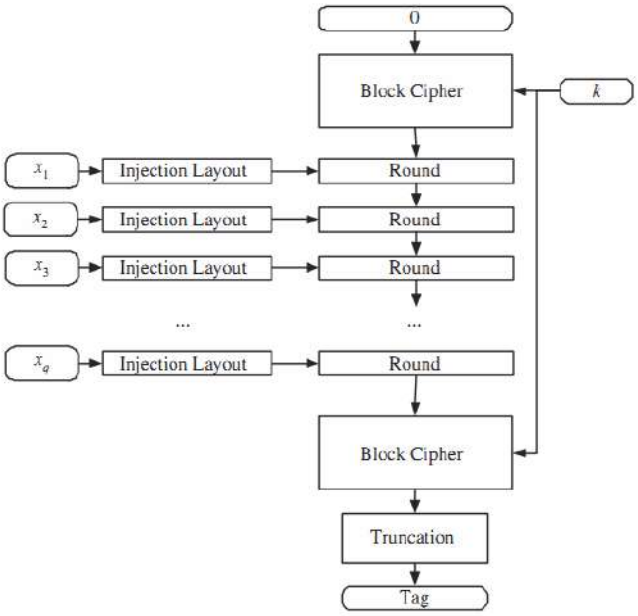
Our contribution is to show in section 4 how the function alpha-MAC can be improved replacing the transformation ShiftRows in every round for a

random diffusion optimal permutation. We evaluate the practical significance of the mentioned attack with the new transformations, and also give our considerations about the replace of the MDS matrix of the transformation MixColumns by another MDS matrix randomly generated.

2 Description of alpha-MAC

Alpha-MAC is an iterative function based on the round transformations of AES. The secret key is used only twice in AES like a block cipher, and the state is changed by consecutive injections of the message blocks. The main advantage of alpha-MAC over another MAC constructions based in a block cipher, such as CBC-MAC, is that per 128 bits of message this one only process four round functions of AES, reducing in 2.5 the runtime respect to CBC-MAC, which is significant for long messages.

The structure of alpha-MAC is presented in the next figure.



Here k is the secret key (128, 192 or 256 bits) at the input of the function and $x_1, x_2, \cdots, x_q$ are the message blocks, each one of 32 bits (assuming that $32q$ is the length of the message, in other case the message is padded). The injection layout consist in the matrix

x_i^1	0	x_i^3	0
0	0	0	0
x_i^2	0	x_i^4	0
0	0	0	0

used in the step AddRoundKey for the unkeyed round functions, where $x_i^1 x_i^2 x_i^3 x_i^4$ is a 32-bit message block for all $1 \leq i \leq q$. The truncation is not significant in the attack presented, and so we assume a 128-bits Tag.

The alpha-MAC algorithm is presented next.

Algorithm 2:

Input: message $x = x_1 x_2 \cdots x_q$ and secret key k

1 **begin**
2 $\quad z_0 = \mathrm{AES}_k(0)$
3 $\quad$ **for** i **from** 1 **to** q **do**
4 $\quad\quad z_i = f(z_{i-1}, I(x_i))$ $\quad f$ = AES round function, I = injection
5 $\quad \mathrm{Tag} = T(\mathrm{AES}_k(z_q))$ $\quad T$ = truncation

Output: message tag $\mathrm{MAC}(x, k) = \mathrm{Tag}$

Daemen and Rijmen also present another MAC function based in the Alred construction with Rijndael as underlying primitive [22]. This another MAC function, named Pelican, can be seen as a simplified and optimized version of alpha-MAC where the injection layout is omitted and the round function of Rijndael is applied four times on each occasion, instead of one. In 2014 they presented an update, Pelican 2.0, with only change a different initial value [23].

3 The proposed attack

Basically the attack consists in a method to find second preimages, based on the assumption that the key (or an intermediate value) is known, exploiting the algebraic properties of the AES round function. One internal collision is a fact if we know only five blocks of the message using this attack.

The second preimage search algorithm is performed in two stages: the Backwards-aNd-Forwards search and the Backwards-aNd-Backwards search, solving each one four groups of two linear equations.

Algorithm 3:

Input: five message blocks $(x_{y-3}, x_{y-2}, x_{y-1}, x_y, x_{y+1})$ and the intermediate value at the input of the round $y-3$

1 **begin**

2 generate a second preimage $(\overline{x}_{y-3}, \overline{x}_{y-2}, \overline{x}_{y-1}, \overline{x}_y, \overline{x}_{y+1})$ randomly, guaranteeing that $\overline{x}_{y-3} \neq x_{y-3}$

3 perform Backwards-aNd-Backwards search to generate a 32-bit collision, modifying the message block $\overline{x}_{y-2}$

4 perform Backwards-aNd-Forwards search to extend 32 bits to 128-bit collision, modifying the message blocks $\overline{x}_{y-1}$, $\overline{x}_y$ and $\overline{x}_{y+1}$

Output: second preimage $(\overline{x}_{y-3}, \overline{x}_{y-2}, \overline{x}_{y-1}, \overline{x}_y, \overline{x}_{y+1})$ of the five message blocks $(x_{y-3}, x_{y-2}, x_{y-1}, x_y, x_{y+1})$

The result of the previous algorithm is that the two five message blocks $(\overline{x}_{y-3}, \overline{x}_{y-2}, \overline{x}_{y-1}, \overline{x}_y, \overline{x}_{y+1})$ and $(x_{y-3}, x_{y-2}, x_{y-1}, x_y, x_{y+1})$ generate the same 128 bit value at the output of the round $y+1$ (an internal collision), knowing the intermediate value at the input of the round $y-3$.

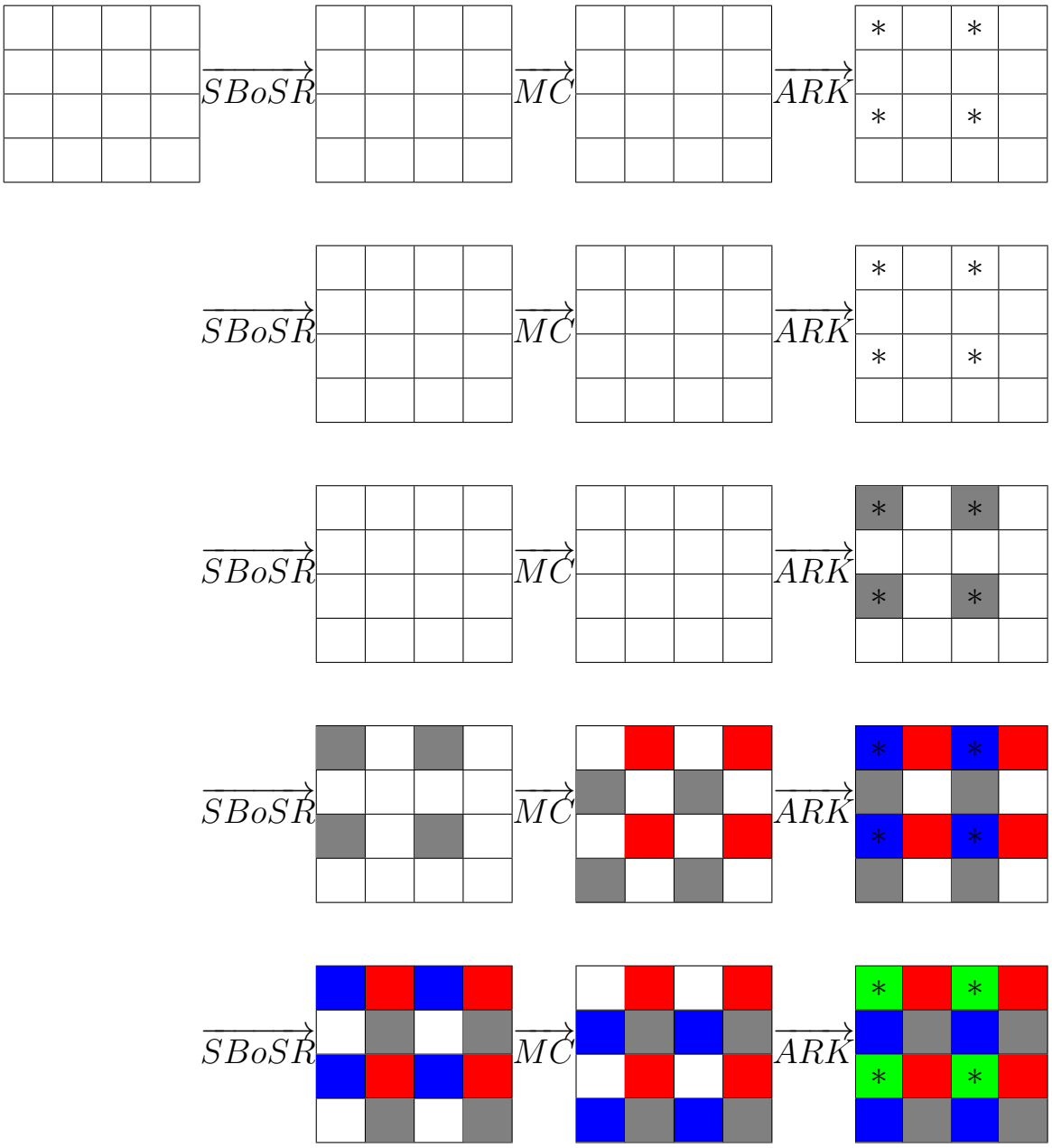
3.1 Backwards-aNd-Forwards

In the Backwards-aNd-Forwards search it is assumed that we were able to find two messages x and $\overline{x}$ which collide in the bytes of the positions [0,1], [0,3], [2,1] and [2,3] at the output of the transformation MixColumns in round y (*blocks in red*).

Step 1: Assume that the bytes of the positions [1,0], [1,2], [3,0] and [3,2] at the output of the transformation MixColumns in round y collide (*blocks in gray*), then solving two equation systems the bytes of the positions [0,0], [0,2], [2,0] and [2,2] at the input of the transformation MixColumns in round y turn collision-dependent. Performing the inverses of ShiftRows and SubBytes it is possible replace the message block $\overline{x}_{y-1}$ making the same one collision-dependent. At this point we has been extend a 32-bit collision to a 64-bit collision in round y after MixColumns.

Step 2: Repeat **Step 1** in the round $y+1$ (*blocks in blue*) to replace the message block $\overline{x}_y$ making the same one collision-dependent. At this point we has been extend a 64-bit collision to a 96-bit collision in round $y+1$ after MixColumns.

Step 3: Get the message block $\overline{x}_{y+1}$ directly canceling the differences between the byte of the positions [0,0], [0,2], [2,0] and [2,2] (*blocks in green*). At this point the five message blocks $(\overline{x}_{y-3}, \overline{x}_{y-2}, \overline{x}_{y-1}, \overline{x}_y, \overline{x}_{y+1})$ collide on 128 bits at the output of the round $y+1$.



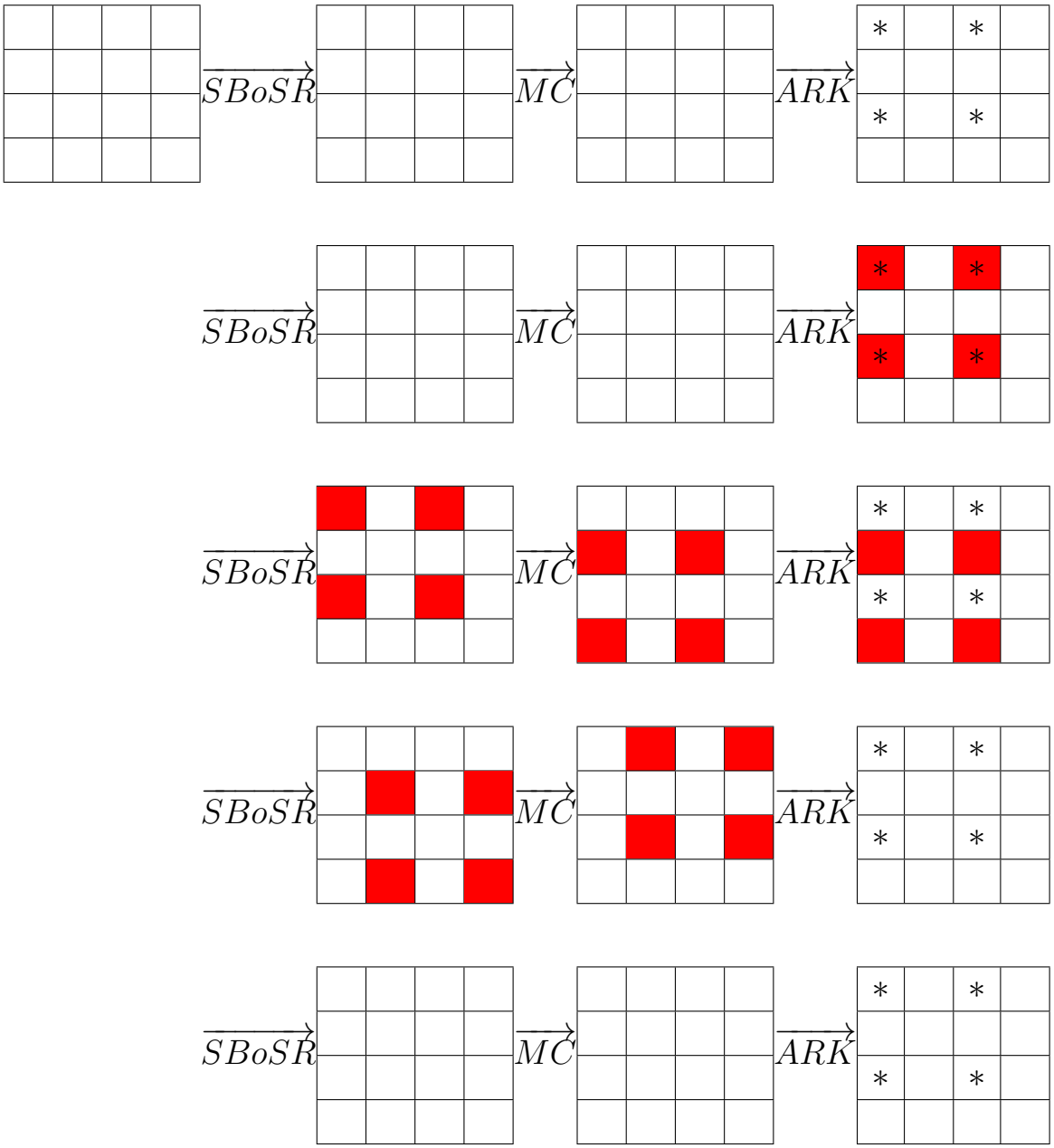
3.2 Backwards-aNd-Backwards

The Backwards-aNd-Backwards search is performed to find a message $\overline{x}$ such that the 32-bit collision requested for the Backwards-aNd-Forwards search is done. The five message blocks $\overline{x}_{y-3}$, $\overline{x}_{y-2}$, $\overline{x}_{y-1}$, $\overline{x}_y$ and $\overline{x}_{y+1}$ are randomly generated.

Step 1: Assume that the bytes of the positions [0,1], [0,3], [2,1] and [2,3] at the output of the transformation MixColumns in round y collide (*blocks in red*), then solving two equation systems the bytes in the positions [1,1], [1,3], [3,1] and [3,3] at the input of the transformation MixColumns in round y turn collision-dependent. Performing the inverses of ShiftRows and SubBytes the

four collision-dependent bytes are shift to the positions [1,0], [1,2], [3,0] and [3,2] at the output of the transformation AddRoundKey in round $y-1$.

Step 2: As the injection layout do not affect the bytes in the previous positions, then the bytes of the positions [1,0], [1,2], [3,0] and [3,2] at the output of MixColumns in round $y-1$ are collision-dependent, and another two equation systems are solved making collision-dependent too the bytes of the positions [0,0], [0,2], [2,0] and [2,2] at the input of MixColumns in round $y-1$. Performing the inverses of ShiftRows and SubBytes it is possible replace the message block $\overline{x}_{y-2}$ making the same one collision-dependent.



4 The proposal for alpha-MAC

In this section we present an improvement of alpha-MAC consisting in the replace of the fixed transformation ShiftRows for a random diffusion optimal permutation [2, 24] in every round. We will prove that this construction turns alpha-MAC in a resistant function to the internal collisions found in [3] and detailed in the previous section.

Exactly, we propose replace the transformation ShiftRows of the round function of AES and alpha-MAC for another one which acts randomly over the state, then we compute the success probability of the presented attack [3] using this transformation, and finally we give our considerations on the use of the same one. We also give an extra judgment about the replacing of the MDS matrix of the transformation MixColumns by another MDS matrix randomly generated.

The new transformation to replace ShiftRows consist in swap the bytes of every column of the state according to random permutations, transpose the resultant state, and swap again the bytes of every column according to random permutations [24]. It give us a set of $24^8 \simeq 2^{36.6}$ transformations that act like ShiftRows, such that the bytes inside every column of the state are placed into different columns after that. Also each of these possibilities is optimal for the minimal rounds claims to reach full diffusion [29], satisfying the design principles of AES [2, section 9.4].

4.1 Success probability of the attack

First we estimate the probability that four bytes of the state be located through the proposed transformation (or their inverse) at the intersections of any two rows and any two columns of the next state. The order of that bytes are not significant.

Let p be the previous probability, then we can prove the following results.

1. Assuming that the four bytes are located two of them in two different columns (like in the attack), thus only two permutations acts on this bytes before the transposition and only two permutations acts on this bytes after the transposition. Considering the positions of the bytes around the transposition, only four of the possible permutations give us this swaps by columns, and then

$$p = \frac{4^4}{24^4} = 6^{-4}$$

2. Assuming that the four bytes are located two in one column and the others in one column each, thus only three permutations acts on this bytes before the transposition and only two permutations acts on this bytes after the transposition. Considering the positions of the bytes around the transposition, with a similar analysis we have

$$p = \frac{4^3 \cdot 12 \cdot 6}{24^5} = \frac{3}{4} \cdot 6^{-4}$$

3. Assuming that the four bytes are located in one column each, thus all the four permutations acts on this bytes before the transposition and only two permutations acts on this bytes after the transposition. Considering the positions of the bytes around the transposition, with a similar analysis we have

$$p = \frac{4^2 \cdot 12 \cdot 18 \cdot 6^2}{24^6} = \frac{2}{3} \cdot 4^{-5}$$

Note that there are not more possibilities (three bytes located in one column or the four bytes located in one column) since the four bytes of the state before the proposed transformation are located to the intersections of two specific rows and two specific columns of the state after that, so an upper bound for p can be $2^{-8.96}$.

4.1.1 Developing the Backwards-aNd-Backwards search

This step allows to find the message block $\overline{x}_{y-2}$ such that the four bytes of the positions [0,1], [0,3], [2,1] and [2,3] at the output of the transformation MixColumns in round y collide. Even if one adversary is able to adapt the equation systems of the round $y-1$, finding four bytes collision-dependent at the input of MixColumns in one of the three possible scenarios studied above, still there is a probability lower than $2^{-8.96}$ such that these solutions are moved to the positions [0,0], [0,2], [2,0] and [2,2] at the output of the round $y-2$ when the transformation used to replace ShiftRows are unknown.

4.1.2 Developing the Backwards-aNd-Forwards search

Here we assume that the Backwards-aNd-Backwards search was successful and the adversary has the possibility to adapt the equation systems of the round y, finding four bytes collision-dependent at the input of MixColumns in one of the three possible scenarios, still there is a probability lower than $2^{-8.96}$ such that these solutions are moved to the positions [0,0], [0,2], [2,0] and [2,2] at the output of the round $y-1$.

Consequently if the adversary has the possibility to adapt the equation systems of the round $y+1$, finding four bytes collision-dependent at the input of MixColumns in one of the three possible scenarios, still there is a probability lower than $2^{-8.96}$ such that these solutions are moved to the same positions at the output of the round y. Here it is also assumed that the transformations used to replace ShiftRows in rounds y and $y+1$ respectively are unknown.

4.2 Final considerations

Assuming that the adversary knows the intermediate value at the input of the round $y-3$ and the message blocks $(x_{y-3}, x_{y-2}, x_{y-1}, x_y, x_{y+1})$, still the transformations used to swap the bytes of the state in the place of ShiftRows remains unknown. Our purpose is to keep hide these transformations under the previous assumption, pretending that the same ones can be randomly generated from some seed, possibly the secret key. Note that an attacker have not possibilities to replace these transformations without the knowledge of the receiver, on the contrary the true message and the corresponding tag do not correspond.

Under these terms, if the adversary is able to modify the attack taking in mind the three possible scenarios, the success probability of the complete attack is smaller than $2^{-26.88}$ when the transformations used to replace ShiftRows remains unknown. If the solutions of the equation systems are not moved to the positions of the injection when it is required, then the message blocks do not affect these bytes, producing an intermediate value collision-dependent at the input of the round $y-3$ different to the intermediate value known for the receiver.

Furthermore we consider another improvement of alpha-MAC consisting in the replace of the fixed MDS matrix of the transformation MixColumns by another MDS matrix randomly generated, such that the same one can be applied in all rounds of the algorithm AES as well as in the intermediate rounds of alpha-MAC. Although in the public literature exists many reports on the random generation of MDS matrices and so in their applications on the block cipher AES, we use the random MDS matrices proposed in [25] motived in the cardinality of the possible generated ones.

Taking in mind this change, if the resulting MDS matrix is known there is not extra security provided, but on the contrary if the MDS matrix remains unknown the complexity of the equation systems increase by the secrecy of the MDS matrix, providing more security against the attack proposed in [3].

5 Conclusion

Combining the Backwards-aNd-Backwards search and the Backwards-aNd-Forwards search is very simple to find a second preimage of alpha-MAC thanks to the algebraic properties of the underlying block cipher AES [3]. In this paper we present an improvement of alpha-MAC based on the replace of the fixed transformation ShiftRows for another one which is generated randomly from one set of 24^8 possibilities, give us a probability lower than $2^{-26.88}$ to make this attack effective when the random transformations used in every round are unknown.

In the studied attack it is assumed that one internal state (or the key) is known, being possible in other attacks like in [4] combining side channel collisions with Differential Power Analysis, and for this reason, side channel countermeasures for AES must be taken into account [4, 23, 26]. Although the collision attack of Biryukov et al [4] is effective on alpha-MAC, the presented construction has been offer resistance against Differential Power Analysis [27] and Differential Fault Analysis [28]. Further studies on the implementation or the security of this construction against other kinds of side channel attacks should still be done.

Finally, as the security analysis of the function alpha-MAC was incomplete and the efforts of the designers were directed to Pelican [26], we propose the same improvement for the MAC function Pelican 2.0, taking into account the generation of random diffusion optimal permutations for the block cipher Rijndael [29].

References

[1] Daemen, J. and Rijmen, V., "A New MAC Construction Alred and a Specific Instance Alpha-MAC", *Lecture Notes in Computer Science*, **3557** (2005), 1 — 17.

[2] Daemen, J. and Rijmen, V., "The Design of Rijndael: AES - The Advanced Encryption Standard", *Springer*, 2002.

[3] Huang, J., Seberry, J. and Susilo, W., "On the Internal Structure of ALPHA-MAC", *Lecture Notes in Computer Science*, **4341** (2006), 271 — 285.

[4] Biryukov, A. et all., "Collision Attacks on AES-based MAC: Alpha-MAC", *Lecture Notes in Computer Science*, **4727** (2007), 166 — 180.

[5] Yuan, Z. et all., "Distinguishing and Forgery Attacks on ALRED and its AES-based Instance Alpha-MAC", *Cryptology ePrint Archive*, 2008.

[6] Yuan, Z et all., "New Birthday Attacks on Some MACs Based on Block Ciphers.", *Lecture Notes in Computer Science*, **5677** (2009), 209 — 230.

[7] Wu, S., Wang, M. and Yuan, Z., "A Flaw in the Internal State Recovery Attack on Alpha-MAC", *Cryptology ePrint Archive*, 2010.

[8] Luykx, A. et all., "A MAC Mode for Lightweight Block Ciphers", International Conference on Fast Software Encryption, *Springer-Berlin-Heidelberg*, 2016, 43 — 59.

[9] Mennink, B. and Neves, S., "Encrypted Davies-Meyer and Its Dual: Towards Optimal Security Using Mirror Theory", Annual International Cryptology Conference, *Springer-Cham*, 2017, 556 — 583.

[10] Datta, N. et all., "Single Key Variant of PMAC-Plus", IACR Transactions on Symmetric Cryptology, 2017, 268 — 305.

[11] Naito, Y., "Blockcipher-based MACs: Beyond the Birthday Bound without Message Length", International Conference on the Theory and Application of Cryptology and Information Security, *Springer-Cham*, 2017, 446 — 470.

[12] Quang, T., "Considering Two MAC under SIG Variants of the Basic SIGMA Protocol", 7th Workshop on Current Trends in Cryptology (CTCrypt 2018), 2018, 232 — 249.

[13] Khoureich, A., "R-MAC - A lightweight authentication protocol for RFID Tags", *Cryptology ePrint Archive*, 2018.

[14] Ankele, R., Bohl, F. and Friedberger, S., "MergeMAC: A MAC for Authentication with Strict Time Constraints and Limited Bandwidth", *Cryptology ePrint Archive*, 2018.

[15] Datta, N. et all., "Encrypt or Decrypt? To Make a Single-Key Beyond Birthday Secure Nonce-Based MAC", Annual International Cryptology Conference, 2018, 631 — 661.

[16] Zoltak, B., "Message Authentication (MAC) Algorithm For The VMPC-R (RC4-like) Stream Cipher", *Cryptology ePrint Archive*, 2019.

[17] Ye, C. and Tian, T., "New Insights into Divide-and-Conquer Attacks on the Round-Reduced Keccak-MAC", *Cryptology ePrint Archive*, 2018.

[18] Luykx, A. and Preneel, B., "Optimal Forgeries Against Polynomial-Based MACs and GCM", Annual International Conference on the Theory and Applications of Cryptographic Techniques, *Springer-Cham*, 2018, 445 — 467.

[19] Leurent, G., Nandi, M. and Sibleyras, F., "Generic Attacks against Beyond-Birthday-Bound MACs", Annual International Cryptology Conference, *Springer-Cham*, 2018, 306 — 336.

[20] Iwata, T. et all., "Universal Forgery and Multiple Forgeries of MergeMAC and Generalized Constructions", *Cryptology ePrint Archive*, 2018.

[21] Liu, F., Cao, Z. and Wang, G., "Finding Ordinary Cube Variables for Keccak-MAC with Greedy Algorithm", *Cryptology ePrint Archive*, 2018.

[22] Daemen, J. and Rijmen, V., "The Pelican MAC Function", 2005.

[23] Daemen, J. and Rijmen, V., "The MAC Function Pelican 2.0", 2014.

[24] Alfonso, A., "Generación Aleatoria de Permutaciones con Óptima Difusión", III Seminario Científico Nacional de Criptografía (Universidad de la Habana, Cuba), 2016.

[25] Freyre, P., Díaz, N., Díaz, R. and Pérez, C., "Random Generation of MDS Matrices", 3rd Workshop on Current Trends in Cryptology (CTCrypt 2014), 2014, 105 — 114.

[26] Daemen, J. and Rijmen, V., "Refinements of the Alred construction and MAC security claims", *IET information security*, **4**:3 (2010), 149 — 157.

[27] Spain, M. and Varia, M., "Diversity within the Rijndael design principles for resistance to differential power analysis", International Conference on Cryptology and Network Security, *Springer-Cham*, 2016, 71 — 87.

[28] Alfonso, A. and Freyre, P., "How secure is the Advanced Encryption Standard with random ShiftRows against Fault Analysis?", *Journal of Science and Technology on Information Security*, **1**:07 (2018), 14 — 21.

[29] Alfonso, A. and Freyre, P., "Random Diffusion Optimal Permutations with a Look in Dynamic Rijndael", *Revista Ciencias Matemáticas*, **32**:1 (2018), 45 — 50.

Appendix

A Example in the first possible case

Let Π be the diffusion optimal permutation

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
5	2	10	15	3	7	16	9	12	4	6	13	14	11	8	1

constructed using the random permutations

$$\tau_1 = [2, 3, 4, 1] \qquad \tau_5 = [2, 1, 3, 4]$$
$$\tau_2 = [1, 3, 2, 4] \qquad \tau_6 = [1, 2, 4, 3]$$
$$\tau_3 = [2, 1, 4, 3] \qquad \tau_7 = [3, 1, 2, 4]$$
$$\tau_4 = [3, 4, 1, 2] \qquad \tau_8 = [4, 3, 2, 1]$$

such that τ_1, τ_2, τ_3 and τ_4 are applied on the columns of the state before the transposition and τ_5, τ_6, τ_7 and τ_8 are applied on the columns of the state after that, as shown below in the following steps

1	5	9	13
2	6	10	14
3	7	11	15
4	8	12	16

2	5	10	15
3	7	9	16
4	6	12	13
1	8	11	14

2	3	4	1
5	7	6	8
10	9	12	11
15	16	13	14

5	3	12	14
2	7	4	11
10	16	6	8
15	9	13	1

then the collision-dependent solutions of the equation systems must come from the positions [0,1], [3,1], [1,3] and [3,3] of the state after Π to be located in the positions of the injection layout.

B Example in the second possible case

Let Π be the diffusion optimal permutation

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
12	5	15	4	7	10	14	3	1	16	11	8	6	2	9	13

constructed using the random permutations

$$\tau_1 = [4, 3, 1, 2] \qquad \tau_5 = [3, 2, 4, 1]$$
$$\tau_2 = [1, 3, 4, 2] \qquad \tau_6 = [2, 3, 4, 1]$$
$$\tau_3 = [4, 2, 3, 1] \qquad \tau_7 = [1, 4, 3, 2]$$
$$\tau_4 = [3, 2, 4, 1] \qquad \tau_8 = [2, 1, 3, 4]$$

such that τ_1, τ_2, τ_3 and τ_4 are applied on the columns of the state before the transposition and τ_5, τ_6, τ_7 and τ_8 are applied on the columns of the state after that, as shown below in the following steps

1	5	9	13
2	6	10	14
3	7	11	15
4	8	12	16

4	5	12	15
3	7	10	14
1	8	11	16
2	6	9	13

4	3	1	2
5	7	8	6
12	10	11	9
15	14	16	13

12	7	1	6
5	10	16	2
15	14	11	9
4	3	8	13

then the collision-dependent solutions of the equation systems must come from the positions [3,1], [0,2], [2,2] and [2,3] of the state after Π to be located in the positions of the injection layout.

C Example in the third possible case

Let Π be the diffusion optimal permutation

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	15	5	10	7	3	12	14	2	8	9	16	13	6	4	11

constructed using the random permutations

$$\tau_1 = [1, 3, 2, 4] \qquad \tau_5 = [1, 4, 2, 3]$$
$$\tau_2 = [1, 3, 4, 2] \qquad \tau_6 = [2, 1, 3, 4]$$
$$\tau_3 = [2, 4, 1, 3] \qquad \tau_7 = [1, 2, 3, 4]$$
$$\tau_4 = [3, 2, 4, 1] \qquad \tau_8 = [4, 2, 1, 3]$$

such that τ_1, τ_2, τ_3 and τ_4 are applied on the columns of the state before the transposition and τ_5, τ_6, τ_7 and τ_8 are applied on the columns of the state after that, as shown below in the following steps

1	5	9	13
2	6	10	14
3	7	11	15
4	8	12	16

1	5	10	15
3	7	12	14
2	8	9	16
4	6	11	13

1	3	2	4
5	7	8	6
10	12	9	11
15	14	16	13

1	7	2	13
15	3	8	6
5	12	9	4
10	14	16	11

then the collision-dependent solutions of the equation systems must come from the positions [0,0], [1,1], [2,2] and [3,3] of the state after Π to be located in the positions of the injection layout.

Fault–Assisted Side Channel Analysis of HMAC–Streebog

Mabin Joseph[1,4], Gautham Sekar[2,5] and R. Balasubramanian[3,4]

[1]Indira Gandhi Centre for Atomic Research, Kalpakkam, Tamil Nadu, India
[2]Madras Fintech Services Pvt. Ltd, Chennai, India
[3]The Institute of Mathematical Sciences, Taramani, Chennai, India
[4]Homi Bhabha National Institute, Training School Complex, Anushakti Nagar, Mumbai, India
[5]Birla Institute of Technology & Science, Pilani, India
mebinjp@gmail.com, gautham.sekar@gmail.com, balu@imsc.res.in

Abstract

Streebog is a family of hash functions defined in the Russian cryptographic standard GOST R 34.11–2012. HMAC–Streebog, which is defined in RFC 7836, is a Streebog based message authentication code. It supports keys of size ranging from 256 bits to 512 bits. In this paper, we present fault–assisted side channel attacks on HMAC–Streebog–256 and HMAC–Streebog–512 that can recover the keys in real-time with $2^{12.98}$ and $2^{14.97}$ average number of fault injections, respectively, to ensure 95% success. The attacker is assumed to be able to simultaneously flip at the most 181 chosen bits of the inner hash if it is a 256–bit variant, and 361 chosen bits of the hash otherwise. In comparison to existing fault attacks on HMAC–Streebog, our attacks have a larger temporal window for fault injection, target a more accessible location and cannot be mitigated with output redundancy countermeasures. Some of the latest hardware vulnerabilities make the HMAC–Streebog implementations vulnerable to our attacks.

Keywords: Streebog, HMAC–Streebog, carry flag, side channel, fault analysis.

1 Introduction

HMAC–Streebog. Streebog is a family of hash functions developed by the Center for Information Protection and Special Communications of the Federal Security Service of the Russian Federation with the participation of the open joint-stock company Information Technologies and Communication Systems (InfoTeCS JSC) [8]. It is defined in the Russian cryptographic standard GOST R 34.11–2012 [8]. Streebog is comprised of two hash functions, Streebog–256 and Streebog–512, which generate 256–bit and 512–bit message digests, respectively.

HMAC is an algorithm to calculate a message authentication code (MAC), based on a hash function [15]. In its first phase, an inner hash is

derived from the message and the inner key. Later, it generates the outer hash from the inner hash and the outer key, and outputs it as the MAC. The specifications of the HMAC algorithms based on Streebog has been defined in RFC 7836 [22]. Corresponding to Streebog–256 and Streebog–512, two HMAC algorithms exist which we call HMAC–Streebog–256 and HMAC–Streebog–512, respectively. The recommended key sizes of HMAC–Streebog–256 and HMAC–Streebog–512 are at least 256 bits and 512 bits, respectively, and not more than 512 bits [15].

Considering its significance, Streebog was well studied over a period of time [1, 3, 4, 19, 20, 2, 21]. Except for the side channel attack by Sekar [21] and the differential fault attack by AlTawy et al. [2], attacks on Streebog are on its reduced–round variants. In [7], Dinur et al. presented the first key recovery attack on HMAC–Streebog–512 with a complexity of 2^{410}. Later, AlTawy et al. presented differential fault attacks [2] on HMAC–Streebog–512 and HMAC–Streebog–256 which can recover the key by injecting single–bit faults into the last two intermediate states of all the compression functions of the outer hash function. The authors claimed that the average number of faults required to recover the input of each compression function of Streebog vary between 338 and 1640.

Contributions of this paper. In this paper, we present side channel attacks on HMAC–Streebog. Carry flag is a bit of the status register, present in nearly every modern microprocessor, that indicate carry overflow in unsigned integer arithmetic. In [21], Sekar conjectured that the carry flag based side channel attack could speed up the key recovery of HMAC–Streebog–256 and HMAC–Streebog–512 by a factor of 2, in certain cases. Our investigation of this conjecture resulted in passive side channel attacks on HMAC–Streebog–256 and HMAC–Streebog–512 which can recover one bit of the respective key with a success rate of 75%. The attacks work under the assumption that the inner hash of the HMAC and the carry flag at the end of MAC generation are known to the attacker. We find that the attacks can be further improved by injecting faults into the output of the inner hash function and propose active attacks on HMAC–Streebog–256 and HMAC–Streebog–512 which can recover the keys with $2^{12.98}$ and $2^{14.97}$ average number of fault injections, respectively, for 95% success rate. We assume that the attacker is able to flip at the most 181 and 361 chosen bits at a time, respectively, for HMAC–Streebog–256 and HMAC–Streebog–512.

To the best of our knowledge, our passive attack is the best non-fault attack on HMAC–Streebog–256. Compared to the attack in [2], our attacks

have a longer temporal window for fault injection and cannot be mitigated using output redundancy countermeasures as we rely on the carry flag side channel.[1,2] Also, it is reasonable to assume that the inner hash being an intermediate output is more accessible than the internal state of Streebog. If the intermediate carries generated by the ripple carry adder implementation of the targeted addition are known, complexities of our passive attacks and fault requirements of our active attacks reduce as each adder can be attacked separately. When implemented in systems which are vulnerable to attacks like Rowhammer [13], Meltdown [17], Spectre [14], SplitSpectre [18], RAMBleed [16] and Cold Boot [12], HMAC–Streebog will be highly vulnerable to our attacks.

Organisation of the paper. The remaining paper is organised as follows. Section 2 describes Streebog and HMAC–Streebog. We present our motivational observations in Sect. 3. Our attacks on HMAC–Streebog–512 and HMAC–Streebog–256 are presented in Sect. 4. The validity of our assumptions is discussed in Sect. 5 and we conclude in Sect. 6.

2 Specification of HMAC–Streebog

Table 1 lists the notation and convention that we follow.

2.1 Description of Streebog

Streebog accepts any message M of length less than 2^{512} bits and returns a 256– or 512–bit digest. If the message length $|M|$ is not a multiple of 512, M is prefixed with a bit string $pad := \{0\}^{511-(|M| \bmod 512)} \parallel 1$. The padded message is then partitioned into $k+1$ 512–bit blocks M_k, M_{k-1}, ..., M_0 such that $pad \parallel M = M_k \parallel M_{k-1} \parallel \ldots \parallel M_0$.

The compression function G has 13 iterations out of which 12 involve a substitution-permutation layer which consists of the following components: a substitution function Γ, a permutation function $\mathcal{F}$, a linear transformation $\mathcal{L}$ and a function $\mathcal{X}[\cdot]$. The substitution function substitutes each byte of its 512–bit input by a byte from a permuted set of $\{0, 1, \ldots, 255\}$ and the permutation function shuffles the position of each byte in its 512–bit input. The linear transformation of a 512–bit input W is performed as follows:

[1] The data to be altered is available for more than $2t$ time, where t is the time taken by the compression function of Streebog, as the targeted modular addition is executed after two compression operations.

[2] In output redundancy countermeasures, data is processed via redundant channels and the output will not be generated unless all of them agree to it. Still, the carry flag side channel remains unaffected.

Symbol/notation	Meaning
$x_{(j)}$	$(j+1)$th bit of x ($j=0$ denotes the least significant bit)
$\oplus$	Exclusive OR
$I_{n,j}$	n–bit binary string such that $I_{n,j(i)} = 1 \iff i = j$
$\|$	Concatenation of two binary strings
$\gg i$	Right shift by i bits
$\|X\|$	Length of X in bits
b^k	$(k \cdot \|b\|)$–bit binary string $bbb\cdots b$
$\Psi_i(Y)$	ith 64–bit word of Y ($i=0$ denotes the least significant word)
$F_1F_2(x)$, where F_1 and F_2 are functions	$F_1(F_2(x))$

Table 1: Notation and convention

$\mathcal{L}(W) = l(\Psi_7(W)) \parallel l(\Psi_6(W)) \parallel \ldots \parallel l(\Psi_0(W))$, where l outputs the right multiplication of its input with a constant matrix $\mathbf{A}$ over $GF(2)$ (see (1)). If a and b are 512–bit strings, $\mathcal{X}[a](b) = a \oplus b$. The compression function G that processes the message block M_i takes as additional inputs the 512–bit chaining value π_i and a length counter n_i, and outputs π_{i+1} (see (3)). The initial value π_0 is a 512–bit IV which is different for Streebog–256 and Streebog–512. Algorithm 1 describes the working of Streebog.

$$l(\zeta) = \bigoplus_{i=0}^{63} \zeta_{(63-i)} \odot \mathbf{A}[i]\,, \tag{1}$$

where

$$\zeta_{(63-i)} \odot \mathbf{A}[i] = \begin{cases} \{0\}^{64}, \text{ if } \zeta_{(63-i)} = 0\,; \\ \mathbf{A}[i], \text{ if } \zeta_{(63-i)} = 1\,. \end{cases} \tag{2}$$

$$\pi_{i+1} := G(\pi_i, M_i, n_i) = E(\mathcal{L}(\mathcal{F}(\Gamma(\pi_i \oplus n_i))), M_i) \oplus \pi_i \oplus M_i\,, \tag{3}$$

where

$$E(\mathcal{L}(\mathcal{F}(\Gamma(\pi_i \oplus n_i))), M_i) = \mathcal{X}[\nu_{13}]\mathcal{L}\mathcal{F}\Gamma\mathcal{X}[\nu_{12}]\mathcal{L}\mathcal{F}\Gamma\mathcal{X}[\nu_{11}]\ldots\mathcal{L}\mathcal{F}\Gamma\mathcal{X}[\nu_1](M_i)\,, \tag{4}$$

and ν_1, ν_2, …, ν_{13} are derived as,

$$\nu_0 = \mathcal{L}\mathcal{F}\Gamma(\pi_i \oplus n_i)\,, \tag{5}$$

$$\nu_{j+1} = \mathcal{L}\mathcal{F}\Gamma(\nu_j \oplus C_j)\,, \text{ for } j = 0, 1, \ldots, 12, \text{ and constants } C_j. \tag{6}$$

2.2 Description of HMAC–Streebog

A HMAC algorithm employs a hash function h in conjunction with a secret key K and generates a MAC value as follows:

$$\text{HMAC}(K, M) = h((K_0 \oplus opad), h((K_0 \oplus ipad), M))\,, \tag{7}$$

Algorithm 1 The Streebog algorithm

Require: The message M, $|M| < 2^{512}$
Ensure: A 256–bit or a 512–bit digest

1. $M \rightarrow pad \parallel M \rightarrow M_k \parallel M_{k-1} \parallel \ldots \parallel M_0$;
2. $\pi_0 = IV$; $n_0 = 0$; $S = 0$;
3. **for** $i = 0$ to $(k-1)$ **do**
4. $\quad \pi_{i+1} = G(\pi_i, M_i, n_i)$;
5. $\quad n_{i+1} = n_i + 512 \bmod 2^{512}$;
6. $\quad S = S + M_i \bmod 2^{512}$;
7. **endfor**
8. $\pi_{k+1} = G(\pi_k, M_k, n_k)$;
9. $n_{k+1} = n_k + 512 - |pad| \bmod 2^{512}$;
10. $S = S + M_k \bmod 2^{512}$;
11. $\pi_{k+2} = G(\pi_{k+1}, n_{k+1}, 0)$;
12. $H = G(\pi_{k+2}, S, 0)$;
13. Output H if Streebog–512, else output $H \gg 256$;

Algorithm 2 The HMAC–Streebog algorithm

Require: The message M and secret key K in big-endian format
Ensure: A 256–bit or a 512–bit digest

1. $K_0 = \{0\}^{512-|k|} \parallel K$;
2. $ipad = \{00110110\}^{64}$;
3. $opad = \{01011100\}^{64}$;
4. $H_{in} = h(M \parallel (K_0 \oplus ipad)))$;
5. $H_{out} = h(H_{in} \parallel (K_0 \oplus opad)))$;
6. Output H_{out};

where M is the message, *opad* and *ipad* are public constants, and K_0 is the secret key if $|K|$ equals the block size of h, or a function of K otherwise. HMAC–Streebog will use either Streebog–512 or Streebog–256 as the underlying hash function. According to RFC 2104 [15], the recommended length of the secret key for a secure HMAC is equal to the digest length of h, and it cannot be more than the block size of h. Therefore the key size of HMAC–Streebog has to be at least 256 or 512 bits depending on whether Streebog–256 or Streebog–512 is used as the hash function. If K is shorter than the block size, it is padded with $\{0\}^{512-|K|}$ to get K_0. Algorithm 2 describes the working of HMAC–Streebog, where h is Streebog–256 or Streebog–512.

Table 2: Table showing the allowed values of the Boolean variables $x_{(i-1)}, y_{(i-1)}$, $c_{(i-1)}$, $c_{(i)}$ and $c'_{(i)}$, and the probability of each event where $p_i = \Pr(c_{(i)} = 0)$, for $i > 1$

$x_{(i-1)}$	$y_{(i-1)}$	$c_{(i-1)}$	$c_{(i)}$	$c'_{(i)}$	**Probability**
0	0	0	0	0	$\frac{1}{4}p_{i-1}$
0	0	1	0	1	$\frac{1}{4}(1-p_{i-1})$
0	1	0	0	1	$\frac{1}{4}p_{i-1}$
0	1	1	1	1	$\frac{1}{4}(1-p_{i-1})$
1	0	0	0	0	$\frac{1}{4}p_{i-1}$
1	0	1	1	0	$\frac{1}{4}(1-p_{i-1})$
1	1	0	1	0	$\frac{1}{4}p_{i-1}$
1	1	1	1	1	$\frac{1}{4}(1-p_{i-1})$

3 Motivational Observations

3.1 On Carry in Modular Addition

Let x and y, which are distributed uniformly at random, be the inputs to an n–bit modular addition, $c_{(i)}$ be the carry generated at the ith bit position of the addition, where $1 \leq i \leq n$ (for instance, outgoing carry at the LSB position is denoted by $c_{(1)}$), and c_{out} be the final carry which equals $c_{(n)}$. The carries generated at the ith bit position when input x is replaced with $x \oplus I_{n,i-1}$ is denoted by $c'_{(i)}$. The allowed values of the Boolean variables $x_{(i-1)}$, $y_{(i-1)}$, $c_{(i-1)}$, $c_{(i)}$ and $c'_{(i)}$, for $i > 1$ and the probability of each event (each possibility is considered an event) are tabulated in Table 2.[3] Let us consider two cases.

Case 1: The ith bit of x *and* $c_{(i)}$ *for any* $i \in \{n, n-1, \ldots, 1\}$ *are known.* From Table 2, we can deduce the following.
If $x_{(i-1)} \oplus c_{(i)} = 1$,

$$y_{(i-1)} \;=\; c_{(i)},\; n \geq i \geq 2\,, \tag{8}$$

$$c_{(i-1)} \;=\; c_{(i)},\; n \geq i \geq 2\,. \tag{9}$$

Else, assuming the events given in Table 2 are mutually exclusive,

$$\Pr(y_{(i-1)} = c_{(i)}) \;=\; \frac{2}{3},\; n \geq i \geq 2\,. \tag{10}$$

[3]We assume that $x_{(i-1)}$, $y_{(i-1)}$ and $c_{(i-1)}$ are independent, and $\Pr(x_{(i-1)} = 0) = \Pr(y_{(i-1)} = 0) = 0.5$.

Since $c_{(0)} = 0$,

$$y_{(0)} = c_{(1)} \iff x_{(0)} \oplus c_{(1)} = 1\,, \tag{11}$$

$$y_{(0)} = 1 \iff x_{(0)} = c_{(1)} = 1\,. \tag{12}$$

Case 2: The ith bit of x, $c_{(i)}$ *and* $c'_{(i)}$ *for any* $i \in \{n, n-1, \ldots, 1\}$ *are known.*
As in Case 1, equations (8), (9), (11) and (12) hold, if the respective conditions are satisfied. In addition to them, we get the following.
If $x_{(i-1)} \oplus c_{(i)} = 0$ and $c_{(i)} = c'_{(i)}$,

$$y_{(i-1)} = c_{(i)},\ n \geq i \geq 1\,, \tag{13}$$

$$c_{(i-1)} = c_{(i)},\ n \geq i \geq 2\,. \tag{14}$$

If $x_{(i-1)} \oplus c_{(i)} = 0$ and $c_{(i)} \neq c'_{(i)}$,

$$y_{(i-1)} = c_{(i-1)} \oplus 1,\ n \geq i \geq 2\,, \tag{15}$$

$$y_{(i-1)} = 1, \text{ for } i = 1\,. \tag{16}$$

3.2 On the Recovery of an Unknown Operand

Let $f_y^n(x)$ be a function which takes an n–bit secret input y, which is distributed uniformly at random, and an n–bit known input x, which is generated by a random oracle, and outputs $z := (x + y) \bmod 2^n$. If the carry c_{out} is known, based on the observations given in Sect. 3.1, the secret input y can be recovered quicker than exhaustive search. We describe two methods to recover y, one each for the Cases of Sect. 3.1.

Passive analysis. Based on Case 1, a passive analysis can be performed to recover the k most significant bits of y using (8)–(12), if c_{out} and the k most significant bits of x are known.

Active analysis. From Case 2, Algorithm 3 has been derived and it additionally requires the carries generated when a set of chosen x's are given as inputs to $f_y^n(x)$. Chosen x's are used in steps 19 and 21.[4]

[4]Since x and y are independent and distributed uniformly at random, $y'_u \parallel x'_l$ and x', which constitute the chosen x's, are also uniformly distributed. The definitions of y'_u, x'_l and x' are given in Algorithm 3.

Algorithm 3 Recovering the secret input y of $f_y^n(x)$ using active analysis

Require: Known input x, carry c_{out} and carries generated when x is modified
Ensure: n–bit y

1. $Y = \{0\}$; $C = \{c_{out}\}$;
2. **for** $i = n$ to 1 **do**
3. $Ynew = \{\}$; $Cnew = \{\}$; $cnt = 0$;
4. **for** $j = 0$ to **sizeof**$(Y) - 1$ **do**
5. $y \leftarrow Y[j]$; $c_{(i)} \leftarrow C[j]$;
6. **if** $x_{(i-1)} \oplus c_{(i)} = 1$
7. $y_{(i-1)} = c_{(i)}$;
8. **if** $i \neq 1$
9. $c_{(i-1)} = c_{(i)}$;
10. **endif**
11. **else if** $i = 1$ & $c_{(1)} = 1$
12. $y_{(0)} = 1$;
13. **else**
14. $x' = x \oplus I_{n,i-1}$;
15. **if** $i \neq n$
16. $x'_l = x'_{(i-1)} \parallel x'_{(i-2)} \ldots \parallel x'_{(0)}$;
17. $y_u = y_{(n-1)} \parallel y_{(n-2)} \ldots \parallel y_{(i)}$;
18. $y'_u = y_u \oplus (2^{n-i} - 1)$;
19. $c'_{(i)}$ = carry generated after executing $f_y^n(y'_u \parallel x'_l)$;[a]
20. **else**
21. $c'_{(i)}$ = carry generated after executing $f_y^n(x')$;
22. **endif**
23. **if** $c'_{(i)} = c_{(i)}$
24. $y_{(i-1)} = c_{(i)}$;
25. **if** $i \neq 1$
26. $c_{(i-1)} = c_{(i)}$;
27. **endif**
28. **else**
29. $y_{(i-1)} = 1$;
30. **if** $i \neq 1$
31. $Ynew[cnt] = y$; $Ynew[cnt]_{(i-1)} = 0$; $Cnew[cnt] = 1$;
32. $c_{(i-1)} = 0$;
33. $cnt = cnt + 1$;
34. **endif**
35. **endif**
36. **endif**
37. $Y[j] \leftarrow y$; $C[j] \leftarrow c_{(i-1)}$;
38. **endfor**
39. **for** $j = 0$ to $cnt - 1$ **do**
40. $Y[j+$**sizeof**$(Y)] = Ynew[j]$; $C[j+$**sizeof**$(Y)] = Cnew[j]$;
41. **endfor**
42. **endfor**
43. Choose y from Y using exhaustive search
44. Output y

[a] Since $y'_u + y_u = \{1\}^{n-i}$, carry generated at the ith bit position of $(y'_u \parallel x'_l) + y$ will be equal to that at the MSB.

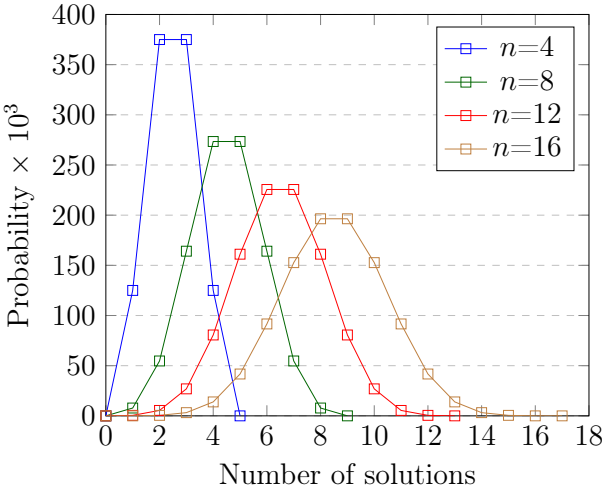


Figure 1: Probability distributions of the size of Algorithm 3's solution space for $n = 4$, 8, 12 and 16

3.3 Passive Analysis vs Active Analysis

According to Case 1 of Sect. 3.1, $x_{(i-1)} \oplus c_{(i)}$ and $x_{(i-2)} \oplus c_{(i-1)}$ equal 1, if and only if $x_{(i-1)}$ and $x_{(i-2)}$ are equal, for $i > 1$. Hence, to recover the k most significant bits of y, where $k \neq n$, with 100% success rate, either c_{out} must be 1 and the corresponding bits of x must be 0 or vice versa; probability of this event turns out to be 2^{-2k}. Therefore, only a few bits can be recovered with this method. Nevertheless, the most significant bit of y can be recovered with 75% success rate using passive analysis as it can be recovered with $\frac{2}{3}$ probability when $x_{(n-1)} \oplus c_{out}$ equals 0.[5]

Compared to passive analysis, active analysis can recover more number of bits with a better success rate. At steps 29, 31 and 32, $(y_{(i-1)}, c_{(i-1)})$ takes the values $(0, 1)$ and $(1, 0)$, following (15), due to which a set of possible solutions of y gets generated. Instead of performing an exhaustive search over the entire space, our search is limited to this set of solutions. In order to analyse the size of this solution space, the algorithm was tested with all possible values of x and y for $n = 4$, 8, 12 and 16. The results show that for any n, at the most n, and on an average $\frac{n}{2}$ or $\frac{n}{2} + 1$, solutions will be generated. The probability distributions of the size of solution space for $n = 4$, 8, 12 and 16 are shown in Figure 1.

The other factors affecting the active analysis are the maximum number of modifications of x required and the maximum number of simultaneous bit-flips needed at any instance of modification. Knowing $x_{(i-1)}$ and $c_{(i)}$, probabilities of passively recovering $y_{(i-1)}$ with 100% success rate for $i \neq 1$

[5]Success probability $:= \frac{1}{4} \cdot 1 + \frac{3}{4} \cdot \frac{2}{3} = 0.75$.

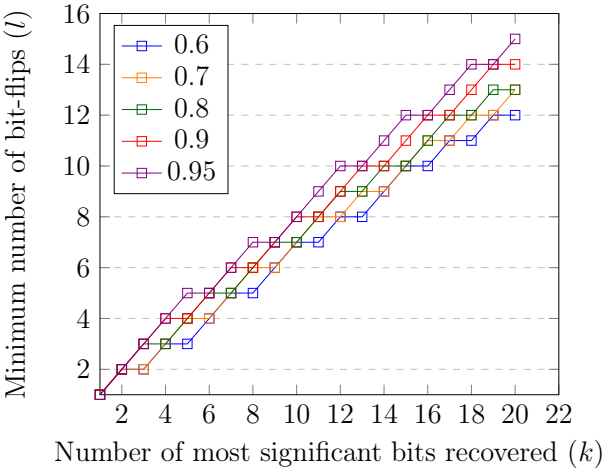


Figure 2: Minimum number of simultaneous bit-flips (l) required to recover the k most significant bits for each of the success probabilities 0.6, 0.7, 0.8, 0.9 and 0.95.

and $i = 1$ are 0.25 and 0.5, respectively. Hence, to recover k most significant bits of y, x has to be modified at the least $0.75k$ times. When multiple solutions of y are generated, the number of modifications of x will be more than $0.75k$ as it depends on the existing solutions of y (see step 19 of Algorithm 3). The average number of modifications of x required to recover the ith bit of y was experimentally computed to be equal to $\frac{3}{4} + \frac{1}{4}\sum_{j=0}^{n-i}(1 - 2^{-j})$ where $i \in \{n, n-1, \ldots, 1\}$. In order to understand the effect of the number of simultaneous bit-flips on the number of bits recovered, we performed another test using all possible inputs for $n = 20$. The minimum number of simultaneous bit-flips l required to recover the k most significant bits of y for each of the success probabilities 0.6, 0.7, 0.8, 0.9 and 0.95 where experimentally computed, and the results are plotted in Figure 2. The linear equations $0.6k - l + 0.5 = 0$, $0.626k - l + 0.574 = 0$, $0.636k - l + 0.821 = 0$, $0.677k - l + 0.837 = 0$ and $0.716k - l + 0.884 = 0$, which were inferred using curve fitting techniques, define the relationship between l and k for the success probabilities 0.6, 0.7, 0.8, 0.9 and 0.95, respectively.

4 Key Recovery Attacks on HMAC–Streebog

We shall now discuss how an attacker, who has access to the authentication device of HMAC–Streebog–512, can recover K with a complexity lesser than that of exhaustive key search. We make the following assumptions:

1. The attacker knows the inner hash H_{in} mentioned in Algorithm 2.

2. The attacker is able to detect the carry flag at the end of message authentication code generation.
3. The attacker is able to alter chosen bits of H_{in} on demand.

From step 5 of Algorithm 2 and step 10 of Algorithm 1, we see that the final operation of HMAC–Streebog–512 that affects the carry flag is $H_{in} + K' \bmod 512$, where $K' = K_0 \oplus opad$. This is because the functions Γ, $\mathcal{F}$, $\mathcal{X}[\cdot]$ and $\mathcal{L}$ do not affect the carry flag. Consequently, the compression function G does not affect the carry flag. We assume that the machine code generated from the reference C implementation of Streebog [6] does not contain any additional operation that affects the carry flag after the final modular addition.

If H_{in} and K' are considered to be x and y, respectively, then $H_{in} + K' \bmod 512$ equals $f_y^{512}(x)$, where $f_y^n(x)$ is the function defined in Sect. 3.2. Since H_{in} and the carry generated by $f_y^{512}(x)$ are known to the attacker, he may be able to recover some of the most significant bits of K' using passive analysis. The strength of the attacker to alter the chosen bits of H_{in} enables him to perform active analysis to recover the entire K' and, in turn, the knowledge of K' leads to the recovery of entire K.

4.1 Complexities of the Attacks

Knowing H_{in} and carry flag, the most significant bit of K can be recovered using passive analysis with 75% success rate. In other words, the time complexity of the key recovery attack using passive analysis for 75% success rate will be $O(2^{511})$. The complexity of recovering K using active analysis varies depending on the strength of the attacker. In Sect. 3.3, we have already discussed the average number of modifications and simultaneous bit-flips required to recover the bits of the unknown variable y. In order to recover the k most significant bits of K, H_{in} has to be modified τ_k times on an average, where τ_k can be calculated as:

$$\tau_k = \sum_{i=512}^{512-k+1} \left(\frac{3}{4} + \frac{1}{4} \sum_{j=0}^{512-i} (1 - 2^{-j}) \right) . \tag{17}$$

At least $0.716k$ simultaneous bit-flips are required for each modification to ensure a 95% success. Therefore, if the attacker has enough resources to flip l chosen bits of H_{in} simultaneously, $1.4l$ most significant bits of K can be recovered using Algorithm 3 with 0.95 success probability and, he can recover the remaining bits through brute force. The time complexity to recover K using active analysis will be $\chi(l) + O(2^{(512-1.4l)})$ for a success rate of 95%, where

Table 3: Variation of the number of key bits recovered ($i \cdot l$) using Algorithm 3 and the required number of modifications of H_{in} in logarithmic scale ($\log_2 \tau_{i \cdot l}$) with the possible number of chosen bit-flips (l) where $i = 1.4$, 1.48, 1.57, 1.6, 1.67 and $i \cdot l \leq 512$.

l	$1.4l$	$\log_2 \tau_{1.4l}$	$1.48l$	$\log_2 \tau_{1.48l}$	$1.57l$	$\log_2 \tau_{1.57l}$	$1.6l$	$\log_2 \tau_{1.6l}$	$1.67l$	$\log_2 \tau_{1.67l}$
1	1	-0.42	1	-0.42	1	-0.42	1	-0.42	1	-0.42
31	43	7.95	45	8.08	48	8.26	49	8.32	51	8.43
61	85	9.87	90	10.03	95	10.19	97	10.24	101	10.36
91	127	11.01	134	11.16	142	11.33	145	11.39	151	11.51
121	169	11.83	179	11.99	189	12.15	193	12.21	202	12.34
151	211	12.46	223	12.62	237	12.80	241	12.84	252	12.97
181	253	12.98	267	13.14	284	13.31	289	13.36	302	13.49
211	295	13.42	312	13.58	331	13.75	337	13.81	352	13.93
241	337	13.81	356	13.96	378	14.14	385	14.19	402	14.31
271	379	14.14	401	14.31	425	14.47	433	14.53	452	14.65
301	421	14.45	445	14.61	472	14.77	481	14.83	502	14.95
331	463	14.72	489	14.88	-	-	-	-	-	
361	505	14.97	-	-	-	-	-	-	-	-

$\chi(l)$ is the time required for $\tau_{1.4l}$ modifications of H_{in}. Assuming that each modification takes $O(\alpha)$ time, $\chi(l) = O(\alpha \cdot \tau_{1.4l})$. Therefore the complexity of our active attack for a success rate of 95% will be $O(\alpha \cdot \tau_{1.4l} + 2^{(512-1.4l)})$, which equals $O(2^{(512-1.4l)})$ if $\log_2 \alpha \cdot \tau_{1.4l} \ll 512 - 1.4l$. Similarly, the respective complexities of our attacks for the success rates of 90%, 80%, 70% and 60% will be $O(\alpha \cdot \tau_{1.48l} + 2^{(512-1.48l)})$, $O(\alpha \cdot \tau_{1.57l} + 2^{(512-1.57l)})$,$O(\alpha \cdot \tau_{1.6l} + 2^{(512-1.6l)})$ and $O(\alpha \cdot \tau_{1.67l} + 2^{(512-1.67l)})$. The variation of $i \cdot l$ and $\log \tau_{i \cdot l}$ with l, where $i = 1.4$, 1.48, 1.57, 1.6, 1.67, is listed in Table 3.

For HMAC–Streebog–256, if $|K| = 256$, the 256 most significant bits of K' and $pad \parallel H_{in}$ will be known to the attacker as they are the padding bits. Therefore, he will be able to compute $c_{(256)}$ from $c_{(512)}$, where $c_{(512)}$ is the carry flag. Hence the attack on 512–bit addition reduces to that on 256–bit addition, which will, in turn, result in the reduction of the attack complexity. For instance, the complexities of our passive attack with success rate of 75% and active attack with success rate of 95% will reduce to $O(2^{255})$ and $O(\alpha \cdot \tau_{1.4l} + 2^{(256-1.4l)})$, respectively.

If the attacker is able to flip at the most 181 and 361 chosen bits of H_{in} then he will be able to recover the keys of HMAC–Streebog–256 and HMAC–Streebog–512, respectively, with negligible complexity. The average number of modifications of H_{in} required for the attacks on HMAC–Streebog–256 and HMAC–Streebog–512 are $2^{12.98}$ and $2^{14.97}$, respectively.

5 Validity of the Assumptions

Our passive attack ceases to exist if any of the first two assumptions made in Sect. 4 is invalid. Similarly, key recovery using active analysis will be possible only if the three associated assumptions of Sect. 4 are valid. Therefore, it is necessary to examine the possibility of our assumptions in a real-world context.

5.1 Extraction of the Inner Hash H_{in}

In [17] and [14], Lipp et al. and Kocher et al. have respectively presented two attacks, the Meltdown attack and the Spectre attack, which enable a malicious application to access the memory space of a different application and read its secrets by exploiting critical vulnerabilities in modern processors. SplitSpectre [18], another speculative execution attack, is a recent variant of Spectre attack proposed by Mambretti et al. which requires a smaller piece of vulnerable code available in the victim's attack surface compared to the original attack. RAMBleed attack [16] by Kwong et al. is a more recent attack which exploits the Rowhammer vulnerability [13] in DRAM cells to read some of the bits in any DDR3 and DDR4 DRAM memories without accessing them. If the attacker has physical access to the hardware, Cold Boot attack [12] is yet another method to extract data from the RAM. The above mentioned attacks target a wide variety of platforms such as personal computers, mobile devices, embedded systems and even cloud environment. Since these attacks are based on hardware-related vulnerabilities, mitigating them without upgrading the hardware cannot be fully ensured. Moreover, efficient variants of the known attacks or even new line of attacks unveil in the course of time.

In order to extract H_{in} from the authentication device, the attacker can use an unprivileged spy software within the same device which utilises one of the above mentioned attacks to read the memory. Being a message digest, it is reasonable to assume that the extraction of H_{in} from memory will not be as difficult as that of a secret key.

5.2 Detection of Carry

The 512–bit addition in Streebog will be mostly implemented as a ripple carry adder where the word size of the full adders will be equal to or less than that of the accumulator of the underlying processor as in [6]. If they are equal, carry flag will be set when a carry is generated. An attacker who

has the necessary privileges to execute any code of his choice in the authentication device will be able to detect the carry flag by executing the add with carry (ADC) assembly instruction. If this method is not feasible, he can use techniques similar to those explained by Fouque et al. which will either exploit the electromagnetic side channel of the device [11] or inject a fault during the computation of the carry [10] to get the carry information. It is easier to detect the carry flag soon after the MAC generation rather than while it is under generation. But, if the above mentioned techniques are performed in synchronisation with the intermediate additions which implement the targeted addition, the attacker will be able to detect all the intermediate carries which, in turn, will reduce the complexities of our attacks as each intermediate addition can be attacked separately.

If the word size of the adder is less than that of the accumulator, carry flag or overflow flag will never be affected. Nevertheless, the intermediate carries will be stored in the buffer as in [6] to forward them to the adjacent adders. The attacker can extract them from the buffer using one of the methods mentioned in Sect. 5.1.

5.3 Chosen Bit Modification of Inner Hash H_{in}

In [13], Kim et al. showed the Rowhammer vulnerability that causes bits in the rows adjacent to the frequently activated rows of a DRAM memory to flip without access. Cojocar et al. [5] demonstrated that multiple bits could be flipped in a chosen manner exploiting this vulnerability and the bits tend to flip to the same value of the bits of the corresponding column in adjacent rows. Very recently, Kwong et al. [16] described and demonstrated the steps to be followed to flip the bits of a target data in the DRAM memory. An unprivileged spy software, similar to that used for the extraction of H_{in}, can be used for its modification too, using the Rowhammer attack.

Another method to flip the bits of H_{in} is by injecting multiple single–bit faults using multi–spot lasers, as mentioned in [9]. But the number of simultaneous bit-flips induced utilising this method will be lesser compared to the earlier one, which leads to an increased attack complexity. Still, if the word size of the adders used in the ripple carry adder is small, which will be the case when HMAC–Streebog is implemented in an embedded system, and the intermediate carries are known, the complexity of our attack can be reduced by applying Algorithm 3 on each adder seperately instead of 512–bit ripple carry adder.

6 Conclusion

In this paper, we have shown side channel attacks on HMAC–Streebog which is a HMAC algorithm based on the Russian standard Streebog and defined in RFC 7836. Our attacks use carry flag as the side channel to recover the keys with complexities lesser than that of exhaustive search. Under the assumption that the output of the inner hash function of HMAC–Streebog is known to the attacker, our passive side channel attacks on HMAC–Streebog–256 and HMAC–Streebog–512 can recover one bit of the respective key with 75% success rate. We have also presented fault–assisted side channel attacks on HMAC–Streebog–256 and HMAC–Streebog–512 which can recover the keys with $2^{12.98}$ and $2^{14.97}$ average number of fault injections, respectively, for 95% success rate under the assumption that the attacker is able to simultaneously flip at the most 181 chosen bits for HMAC–Streebog–256 and 361 chosen bits for HMAC–Streebog–512. We have highlighted some of the latest hardware vulnerabilities which make the HMAC–Streebog implementations vulnerable to our attacks. To the best of our knowledge, our passive attack is the best non-fault attack on HMAC–Streebog–256. Compared to the other fault attacks on HMAC–Streebog, our attacks have a larger temporal window for fault injection, target a more accessible location and cannot be mitigated with output redundancy countermeasures. The attacks presented here emphasises the importance of preventing the side channel leakage of carry bits.

References

[1] Abdelkhalek A., AlTawy R., Youssef A.M., "Impossible Differential Properties of Reduced Round Streebog", *LNCS*, C2SI-2015, **9084**, 2015, 274–286.

[2] AlTawy R., Youssef A.M., "Differential Fault Analysis of Streebog", *LNCS*, ISPEC 2015, **9065**, 2015, 35–49.

[3] AlTawy R., Youssef A.M., "Watch your constants: malicious Streebog", *IET Information Security*, **9(6)** (2015), 328–333.

[4] Biryukov A., Perrin L., Udovenko A., "Reverse-Engineering the S-Box of Streebog, Kuznyechik and STRIBOBr1", *LNCS*, EUROCRYPT 2016, **9665**, 2016, 372–402.

[5] Cojocar L., Razavi K., Giuffrida C., Bos H., "Exploiting Correcting Codes: On the Effectiveness of ECC Memory Against Rowhammer Attacks", IEEE SP 2019, 2019, 279–295.

[6] Degtyarev A., *GOST R 34.11-2012 hash function with 512/256 bit digest*, GitHub, 2019, https://github.com/adegtyarev/streebog.

[7] Dinur I., Leurent G., "Improved Generic Attacks against Hash-Based MACs and HAIFA", *LNCS*, CRYPTO 2014, **8616**, 2014, 149–168.

[8] Dolmatov V., Degtyarev A., "GOST R 34.11-2012: Hash Function", *RFC 6986*, Internet Engineering Task Force (IETF), 2013, https://tools.ietf.org/html/rfc6986.

[9] Hubert C., Côme M., *ERROL - SMART CARD SECURITY*, 2019, https://www.errol-laser.com/smart-cart-laser-bench.

[10] Fouque P., Masgana D., Valette F., "Fault Attack on Schnorr Based Identification and Signature Schemes", IEEE FDTC 2009, 2009, 32–38.

[11] Fouque P., Réal D., Valette F., Drissi M., "The Carry Leakage on the Randomized Exponent Countermeasure", *LNCS*, CHES 2008, **5154**, 2008, 198–213.

[12] Halderman J.A., Schoen S.D., Heninger N., Clarkson W., Paul W., Calandrino J.A., Feldman A.J., Appelbaum J., Felten E.W., "Lest We Remember: Cold Boot Attacks on Encryption Keys", *Communications of the ACM*, **52(5)** (2009), 91–98.

[13] Kim Y., Daly R., Kim J., Fallin C., Lee J.H., Lee D., Wilkerson C., Lai K., Mutlu O., "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors", ACM/IEEE ISCA 2014, 2014, 361–372.

[14] Kocher P., Horn J., Fogh A., Genkin D., Gruss D., Haas W., Hamburg M., Lipp M., Mangard S., Prescher T., Schwarz M., Yarom Y., "Spectre Attacks: Exploiting Speculative Execution", IEEE SP 2019, 2019, 19–37.

[15] Krawczyk H., Bellare M., Canetti R., "HMAC: Keyed-Hashing for Message Authentication", *RFC 2104*, Internet Engineering Task Force (IETF), 1997, https://tools.ietf.org/html/rfc2104.

[16] Kwong A., Genkin D., Gruss D., Yarom Y., "RAMBleed: Reading Bits in Memory Without Accessing Them", IEEE SP 2020, 2020, 310–326.

[17] Lipp M., Schwarz M., Gruss D., Prescher T., Haas W., Fogh A., Horn J., Mangard S., Kocher P., Genkin D., Yarom Y., Hamburg M., "Meltdown: Reading Kernel Memory from User Space", USENIX Security '18, 2018, 973–990.

[18] Mambretti A., Neugschwandtner M., Sorniotti A., Kirda E., Robertson W., Kurmus A., "Let's Not Speculate: Discovering and Analyzing Speculative Execution Attacks", *IBM Research Report*, **RZ 3933**, 2018, https://domino.research.ibm.com/library/cyberdig.nsf/papers/D66E56756964D8998525835200494B74.

[19] Perrin L., Udovenko A., "Exponential S-Boxes: a Link Between the S-Boxes of BelT and Kuznyechik/Streebog", *IACR Transactions on Symmetric Cryptology*, **2016(2)** (2017), 99–124.

[20] Perrin L., "Partitions in the S-Box of Streebog and Kuznyechik", *IACR Transactions on Symmetric Cryptology*, **2019(1)** (2019), 302–329.

[21] Sekar G., "Side Channel Cryptanalysis of Streebog", *LNCS*, SSR 2015, **9497**, 2015, 154–162.

[22] Smyshlyaev S., Alekseev E., Oshkin I., Popov V., Leontiev S., Podobaev V., Belyavsky D., "Guidelines on the Cryptographic Algorithms to Accompany the Usage of Standards GOST R 34.10-2012 and GOST R 34.11-2012", *RFC 7836*, Internet Engineering Task Force (IETF), 2016, https://tools.ietf.org/html/rfc7836.

On group generated by ciphers based on Feistel network

Vladimir Antipkin and Dmitriy Pasko

TVP Laboratories, Russia
pasko.do@yandex.ru

Abstract

In this paper we study Feistel network based ciphers with XSL-like round function and provide sufficient conditions for such ciphers to generate alternating group. We extend results obtained by A.S. Maslov [1] for XSL ciphers (SA-substitutions).

Keywords: Feistel network, alternating group, XSL, block cipher.

1 Introduction and preliminaries

Feistel network is one of the most popular approaches to construction of block ciphers. The particular Feistel network based cipher is determined by the round function. It this paper we study one type of round functions and provide sufficient conditions for ciphers with such round function to generate alternating group. Motivation for the investigation is the fact that the group generated by round function of block cipher contains as a subset the set of transformations performed by the actual cipher. Although it's preferably to study the set itself, in most cases it's a difficult problem, but some undesirable properties of the set, like imprimitivity or small size, can be determined the corresponding properties of the group generated by round functions.

Denote $\overline{a,b} = \{a, a+1, ..., b\}$ for positive integer $a \leq b$. Let V_t be vector space of dimension t over $GF(2)$. For $m, n > 1$ we consider $V_{mn} = V_m^n$ and if $\alpha \in V_{mn}$, then $\alpha = (\alpha_1, \ldots, \alpha_n), \alpha_i \in V_m, i \in \overline{1,n}$. Define by $\pi_c(x)$ a permutation $\pi_c : x \to x \oplus c$ of V_{mn}, $c \in V_{mn}$; $s(x), s'(x)$ – permutations of V_m; $S(\alpha) = (s(\alpha_1), \ldots, s(\alpha_n))$, $S'(\alpha) = (s'(\alpha_1), \ldots, s'(\alpha_n))$, $\alpha \in V_{mn}$; $L(x), L'(x)$ non-singular linear transformations of V_{mn}.

For simplicity we consider that permutations in each S and S' are all identical, but the following results are correct for different permutations in S and S' as well.

We write an application of a function f to an argument x as $f(x)$ or x^f depending on the situation. In the latter case $x^{f_1 f_2}$ means $f_2(f_1(x))$.

Let g_k^F be one round of Feistel network based cipher with round function $F_k(x) : V_{mn} \times X \to V_{mn}$, where X is a set of round keys. We will write function F for fixed $k \in X$ as F_k. The result of g_k^F applied to block $(a, b) \in V_{mn}^2$ is

$$g_k^F(a, b) = \left(a^{F_k} \oplus b, a\right).$$

Define by R_F the group generated by g_k^F with different k:

$$R_F = \langle g_k^F | k \in X \rangle.$$

We fix integer $l > 1$ and define $X = V_{mn}^l, k \in V_{mn}^l, k = (k_1, \ldots k_l)$. Let's consider round function F_k of the following form

$$F_k(x) = x^{Q\pi_{k_1} SL},$$

where $Q : V_{mn} \times V_{mn}^{l-1} \to V_{mn}$. Define $Q_{k_1 \ldots, k_{l-1}}(x) = Q(x, k_1, \ldots, k_{l-1})$. Our goal is to show that

$$R = A_{2^{2mn}},$$

where $A_{2^{mn}} = A(V_{2mn})$ – alternating group of permutations of V_{2mn}.

Let matrix L be constructed of $(m \times m)$ blocks $L_{ij} \in GL_n(2)$:

$$L = \|L_{ij}\| \in GL_{mn}(2).$$

If $x = (x_1, \ldots, x_n)$, $y = (y_1, \ldots, y_n)$, $x_i, y_i \in V_m$, $i \in \overline{1, n}$, and $y = L(x)$, then coordinate function y_j can be written as $y_j = \sum_{t=1}^{n} x_t L_{tj}$.

For transformation L we construct directed "diffusion graph" Γ_L with set of nodes $V = \{1, \ldots, n\}$ and set of edges X_L. Edge $(i, j) \in X_L$ iif coordinate function y_j depends substantially on x_i, or, in other words, if matrix L_{ij} is non-null.

For $\alpha = (\alpha_1, \ldots, \alpha_n) \in V_m^n$ we define the set of non-zero coordinates $N(\alpha) = \left\{i | \alpha_i \neq \vec{0}\right\}$. If $I \subseteq V$, then we define $N_I(\alpha) = N(\alpha) \cap I$. Define by $L(I)$ the set of tails of edges of directed graph Γ_L with heads in set I.

We impose the following conditions on linear transformation L and permutation S.

Condition FSA1. Directed graph Γ_L is strongly connected and greatest common divisor of its cycles' length is equal to 1.

For given $a_1, a_2 \in V_m$, $a_1 \neq a_2$, we construct vector spaces $V^{(a_1, a_2)} \subseteq V_{2m}$ generated by all vectors of the form

$$\left(a_1^{\pi_{k_1} s} \oplus a_1^{\pi_{k_2} s}, a_2^{\pi_{k_1} s} \oplus a_2^{\pi_{k_2} s}\right), k_1, k_2 \in V_m$$

Condition FSA2. For any a_1, $a_2 \in V_m$, $a_1 \neq a_2$, $V^{(a_1,a_2)} = V_{2m}$.

Condition **FSA2** means that dimension of $V^{(a_1,a_2)}$ equals to $2m$.

Define

$$M_i(L) = \left\{ j \in \overline{1,n} | L_{ij} \text{ is non-singular matrix} \right\}, i \in \overline{1,n}$$

Condition FSA3. In matrix L for any $i, j \in \overline{1,n}$ block L_{ij} is either non-singular matrix or null matrix, besides for any $t \in \overline{1,n-1}$ and pairwise different $i_1 \ldots, i_t \in \overline{1,n}$ inequality $\left| \bigcup_{j=1}^{t} M_{i_j}(L) \right| > t$ holds.

Condition FSA4. $\min\limits_{\alpha \in V_m \setminus \{0\}} |\{s(x \oplus \alpha) \oplus s(x) | x \in V_m\}| > \max\limits_{i \in \overline{1,n}} |M_i(L)|$.

Condition **FSA4** for permutation s is bound to the following

$$p_s = \max_{a,b \in V_m \setminus \{0\}} |\{x \in V_m | s(x \oplus a) \oplus s(x) = b\}| \cdot 2^{-m}.$$

For any $a \in V_m \setminus \{0\}$ the following inequality holds

$$2^m = \sum_{(*)} |\{x \in V_m | s(x \oplus a) \oplus s(x) = b\}|,$$

summing up for all $b \in V_m$, such that for certain $x \in V_m$ the equality $s(x \oplus a) \oplus s(x) = b$ holds. Since value in sum is less or equal to $2^m p_s$, then

$$p_s^{-1} \leq \sum_{(*)} 1 = |\{x \in V_m | s(x \oplus a) \oplus s(x) = b\}|$$

and condition **FSA4** on permutation s follows from condition $\max\limits_{i \in \overline{1,n}} |M_i(L)| < p_s^{-1}$

Note. If S includes different permutations $s_1, \ldots, s_n$, then condition **FSA4** transforms into the following: for any $i \in \overline{1,n}$ the inequality

$$\min_{\alpha \in V_m \setminus \{0\}} |\{s(x \oplus \alpha) \oplus s(x) | x \in V_m\}| > |M_i(L)|,$$

or $|M_i(L)| < p_{s_i}^{-1}$, holds.

First we prove a number of subsidiary statements.

Theorem 1. *For any $m > 2$, set $I \subseteq V$ and vector $\beta \in V_{mn}$ the following inequality holds*

$$\max_{\alpha : N(\alpha) \subseteq I} \left| N_{L(I)} \left(\alpha^L \oplus \beta \right) \right| \geq |I|.$$

This inequality is strict if $|L(I)| > |I|$.

For proof see appendix A. It's similar to the proof of theorem 2 in [1]

Corollary 1. *Let* $\alpha \in V_{mn}$ *be the vector that gives maximum* $\max\limits_{\alpha:N(\alpha)\subseteq I} |N_{L(I)}\left(\alpha^L \oplus \beta\right)|$. *Then* $\left|N\left(\alpha^L \oplus \beta\right)\right| \geq |I|$.

Proof. From theorem 1 we have $\max\limits_{\alpha:\, N(\alpha)\subseteq I} \left|N_{L(I)}\left(\alpha^L \oplus \beta\right)\right| \geq |I|$. By definition $N_{L(I)}\left(\alpha^L \oplus \beta\right) = N\left(\alpha^L \oplus \beta\right) \cap L(I)$, hence $\left|N\left(\alpha^L \oplus \beta\right)\right| \geq |I|$. ■

Note that if $|L(I)| < |I|$, then theorem 1 is wrong. However, the following statement holds.

Statement 1. *Let* L *be non-singular linear transformation of space* V_{mn}. *Then for any subset* $I \subseteq V$ *the following inequality holds*

$$|L(I)| \geq |I|$$

Proof. Let $a \in V_{mn}$ and $L(a) = (l_1(a), \ldots, l_n(a))$, where l_j are linear mappings from V_{mn} to V_m, $j \in \overline{1,n}$. For certain $I \subseteq V$ define by W_I a subspace of V_{mn} such that $N(w) \subseteq I$ for any $w \in W_I$, and $W_{L(I)} = L(W_I)$. Since L is non-singular, then $\dim W_I = \dim W_{L(I)}$.

On the other hand, for any $w \in W_I$ and $j \in \overline{1,n} \setminus L(I)$ equality $l_j(w) = 0$ holds, because l_j depends only on zero subvectors of w. So if $|L(I)| < |I|$, then $\left|\overline{1,n} \setminus L(I)\right| > \left|\overline{1,n} \setminus I\right|$, that is $\max\limits_{w\in W_{L(I)}} |N(w)| < \max\limits_{w\in W_I} |N(w)|$, and $\dim W_{L(I)} < \dim W_I$ – contradiction. Hence $|L(I)| \geq |I|$. ■

Lemma 1. *Let* $h_{k_1,\ldots,k_{2t}}(x) = x^{\pi_{k_1}S} \oplus x^{\pi_{k_2}S} \oplus \cdots \oplus x^{\pi_{k_{2t}}S}$ *for an arbitrary natural* $t > 0$, $k_1, \ldots k_{2t} \in V_{mn}$, *and components of transformation* S *satisfy the condition* ***FSA₂***. *Then for any* $\alpha, \beta, \gamma_1, \gamma_2 \in V_{mn}$, *such that* $N(\gamma_1 \oplus \gamma_2) \subseteq N(\alpha \oplus \beta)$, *there are* $t, k_1, \ldots, k_{2t}$ *that*

$$\begin{cases} h_{k_1,\ldots,k_{2t}}(\alpha) = \gamma_1, \\ h_{k_1,\ldots,k_{2t}}(\beta) = \gamma_2. \end{cases}$$

Proof. Since S acts independently on subvectors of dimension m, then transformation $h_{k_1,\ldots,k_{2t}}(x)$ can be considered acting on subvectors of dimension m independently, that is

$$h_{k_1,\ldots,k_{2t}}(x) = (h_1(x_1), h_2(x_2), \ldots, h_n(x_n))$$

for certain h_i. If we denote $k_j = \left(k_j^{(1)}, \ldots, k_j^{(n)}\right)$, then

$$h_i(x_i) = (x_i)^{\pi_{k_1^{(i)}}S} \oplus (x_i)^{\pi_{k_2^{(i)}}S} \oplus \cdots \oplus (x_i)^{\pi_{k_{2t}^{(i)}}S}.$$

Let $\alpha_i, \beta_i \in V_m,$, $\alpha_i \neq \beta_i, i \in \overline{1,n}$. Then

$$\begin{pmatrix} h_i(\alpha_i) \\ h_i(\beta_i) \end{pmatrix} = \begin{pmatrix} \alpha_i^{\pi_{k_1^{(i)}}S} \oplus \cdots \oplus \alpha_i^{\pi_{k_{2t}^{(i)}}S} \\ \beta_i^{\pi_{k_1^{(i)}}S} \oplus \cdots \oplus \beta_i^{\pi_{k_{2t}^{(i)}}S} \end{pmatrix} = \begin{pmatrix} \alpha_i^{\pi_{k_1^{(i)}}S} \\ \beta_i^{\pi_{k_1^{(i)}}S} \end{pmatrix} \oplus \begin{pmatrix} \alpha_i^{\pi_{k_2^{(i)}}S} \\ \beta_i^{\pi_{k_2^{(i)}}S} \end{pmatrix} \oplus \cdots \oplus \begin{pmatrix} \alpha_i^{\pi_{k_{2t}^{(i)}}S} \\ \beta_i^{\pi_{k_{2t}^{(i)}}S} \end{pmatrix}.$$

For different t and every possible $k_1^{(i)}, \ldots, k_{2t}^{(i)}$ the right part of the latter formula takes every possible value from the space $V^{(\alpha_i,\beta_i)}$. By the condition of lemma, permutation s satisfies the condition $\mathbf{FSA_2}$, that is $V^{(\alpha_i,\beta_i)} = V_{2m}$. It means that whatever $\alpha_i, \beta_i \in V_m$, $\alpha_i \neq \beta_i$, $i \in \overline{1,n}$, $\gamma_1^{(i)}, \gamma_2^{(i)} \in V_m$, we take, we can find $t, k_1^{(i)}, \ldots, k_{2t}^{(i)}$ such that $h_i(\alpha_i) = \gamma_1^{(i)}$, $h_i(\beta_i) = \gamma_2^{(i)}$. ■

2 Main result

Theorem 2. *Let permutation s satisfies the condition $\boldsymbol{FSA_2}$ and one of the following conditions holds:*

1) for any $a_1, a_2 \in V_{mn}$ there are $k_1, \ldots, k_{l-1} \in V_{mn}$ such that

$$|N(a_1 \oplus a_2)| < \left| N\left(a_1^{Q_{(k_1,\ldots,k_{l-1})}} \oplus a_2^{Q_{(k_1,\ldots,k_{l-1})}}\right)\right|$$

if $N(a_1 \oplus a_2) < n$ and

$$|N(a_1 \oplus a_2)| = \left| N\left(a_1^{Q_{(k_1,\ldots,k_{l-1})}} \oplus a_2^{Q_{(k_1,\ldots,k_{l-1})}}\right)\right|$$

if $N(a_1 \oplus a_2) = n$

2) for any $a_1, a_2 \in V_{mn}$ there are $k_1, \ldots, k_{l-1} \in V_{mn}$ such that

$$N(a_1 \oplus a_2) \subseteq N(a_1^{Q_{(k_1,\ldots,k_{l-1})}} \oplus a_2^{Q_{(k_1,\ldots,k_{l-1})}})$$

and transformation L satisfies the condition $\boldsymbol{FSA_1}$

Then group R_F equals to alternating group $A_{2^{2mn}}$

Proof. First we prove that group R_F is 2-transitive. To do this we show that an arbitrary distinct pair of blocks (a_1, b_1), (a_2, b_2) can be transformed into an arbitrary distinct pair of blocks (c_1, d_1), (c_2, d_2) under certain permutations from R. We make this transformation in three steps:

1. Transform (a_1, b_1), (a_2, b_2) transforms into a certain pair (a_1', b_1'), (a_2', b_2') such that $N(a_1' \oplus a_2') = N(b_1' \oplus b_2') = V$;
2. Transform (c_1, d_1), (c_2, c_2) transforms into a certain pair (c_1', d_1'), (c_2', d_2') such that $N(c_1' \oplus c_2') = N(d_1' \oplus d_2') = V$;
3. Transform (a_1', b_1'), (a_2', b_2') transforms into (c_1', d_1'), (c_2', d_2').

First we show that we can perform steps 1 and 2, that is we can always transform an arbitrary distinct pair into a pair of blocks which differ in every subvector.

Consider an action of permutation

$$g_{k^{(1)}}^F \left(g_{k^{(2)}}^F\right)^{-1} = g_{k^{(1)}} g_{k^{(2)}}^{-1},$$

where $k^{(1)}, k^{(2)} \in V_{mn}^l$, $k^{(i)} = \left(k_1^{(i)}, \ldots, k_l^{(i)}\right), k_j^{(i)} \in V_{mn}$, on a pair of blocks $(a_1, b_1), (a_2, b_2), a_1 \neq a_2$:

$$\begin{cases} (a_1, b_1)^{g_{k^{(1)}} g_{k^{(2)}}^{-1}} = \left(a_1, \left(a_1^{Q_{\left(k_1^{(1)}, \ldots, k_{l-1}^{(1)}\right)^{\pi_{k_l^{(1)}} S}} \oplus a_1^{Q_{\left(k_1^{(2)}, \ldots, k_{l-1}^{(2)}\right)^{\pi_{k_l^{(2)}} S}}}\right)^L \oplus b_1\right), \\ (a_2, b_2)^{g_{k^{(1)}} g_{k^{(2)}}^{-1}} = \left(a_2, \left(a_2^{Q_{\left(k_1^{(1)}, \ldots, k_{l-1}^{(1)}\right)^{\pi_{k_l^{(1)}} S}} \oplus a_2^{Q_{\left(k_1^{(2)}, \ldots, k_{l-1}^{(2)}\right)^{\pi_{k_l^{(2)}} S}}}\right)^L \oplus b_2\right) \end{cases}$$

Let keys of transformation Q be equal, that is $\left(k_1^{(1)}, \ldots, k_{l-1}^{(1)}\right) = \left(k_1^{(2)}, \ldots, k_{l-1}^{(2)}\right)$, and denote $\overline{a}_i = a^{Q_{\left(k_1^{(1)}, \ldots, k_{l-1}^{(1)}\right)}}, i = 1, 2$. Then

$$\begin{cases} (a_1, b_1)^{g_{k^{(1)}} g_{k^{(2)}}^{-1}} = \left(a_1, \left(\overline{a}_1^{\pi_{k_l^{(1)}} S} \oplus \overline{a}_1^{\pi_{k_l^{(2)}} S}\right)^L \oplus b_1\right), \\ (a_2, b_2)^{g_{k^{(1)}} g_{k^{(2)}}^{-1}} = \left(a_2, \left(\overline{a}_2^{\pi_{k_l^{(1)}} S} \oplus \overline{a}_2^{\pi_{k_l^{(2)}} S}\right)^L \oplus b_2\right) \end{cases}$$

In case 2 of theorem by theorem's condition we can choose $\left(k_1^{(1)}, \ldots, k_{l-1}^{(1)}\right) = \left(\tilde{k}_1^{(1)}, \ldots, \tilde{k}_{l-1}^{(1)}\right)$ such that $|N(a_1 \oplus a_2)| < |N(\overline{a}_1 \oplus \overline{a}_2)|$. In case 1 of theorem we can choose $\left(\tilde{k}_1^{(1)}, \ldots, \tilde{k}_{l-1}^{(1)}\right)$ such that $N(a_1 \oplus a_2) \subseteq N(\overline{a}_1 \oplus \overline{a}_2)$.

Now we take $2t$ keys $k^{(1)}, \ldots, k^{(2t)}, t \geq 1$, such that $\left(k_1^{(i)}, \ldots, k_{l-1}^{(i)}\right) = \left(\tilde{k}_1^{(1)}, \ldots, \tilde{k}_{l-1}^{(1)}\right)$ for any $i \in \overline{1, 2t}$ and apply permutation $g_{k^{(1)}} g_{k^{(2)}}^{-1} \ldots g_{k^{(2t-1)}} g_{k^{(2t)}}^{-1}$ to a pair of blocks $(a_1, b_1), (a_2, b_2)$:

$$\begin{cases} (a_1, b_1)^{g_{k^{(1)}} g_{k^{(2)}}^{-1} \cdots g_{k^{(2t-1)}} g_{k^{(2t)}}^{-1}} = \left(a_1, \left(\bigoplus_{i=1}^{2t} \overline{a}_1^{\pi_{k_l^{(i)}} S}\right)^L \oplus b_1\right), \\ (a_2, b_2)^{g_{k^{(1)}} g_{k^{(2)}}^{-1} \cdots g_{k^{(2t-1)}} g_{k^{(2t)}}^{-1}} = \left(a_2, \left(\bigoplus_{i=1}^{2t} \overline{a}_2^{\pi_{k_l^{(i)}} S}\right)^L \oplus b_2\right) \end{cases}$$

Transformation $h_{k_l^{(1)}, \ldots, k_l^{(2t)}}(x) = x^{\pi_{k_l^{(1)}} S} \oplus x^{\pi_{k_l^{(2)}} S} \oplus \cdots \oplus x^{\pi_{k_l^{(2t)}} S}$ satisfies the condition of lemma 1, so for any $\gamma_1, \gamma_2 \in V_{mn}, suchthat N(\gamma_1 \oplus \gamma_2) \subseteq N(\overline{a}_1 \oplus \overline{a}_2)$ there are $t, k_l^{(1)}, \ldots, k_l^{(2t)}$ such that

$$\begin{cases} h_{k_l^{(1)},\ldots,k_l^{(2t)}}\left(\overline{a}_1\right) = \gamma_1, \\ h_{k_l^{(1)},\ldots,k_l^{(2t)}}\left(\overline{a}_2\right) = \gamma_2 \end{cases}$$

and hence

$$\begin{cases} (a_1, b_1)^{g_{k^{(1)}} g_{k^{(2)}}^{-1} \cdots g_{k^{(2t-1)}} g_{k^{(2t)}}^{-1}} = \left(a_1, \gamma_1^L \oplus b_1\right), \\ (a_2, b_2)^{g_{k^{(1)}} g_{k^{(2)}}^{-1} \cdots g_{k^{(2t-1)}} g_{k^{(2t)}}^{-1}} = \left(a_2, \gamma_2^L \oplus b_2\right) \end{cases} \tag{1}$$

Now apply $g_{(0,\ldots,0)}$:

$$\begin{cases} \left(a_1, \gamma_1^L \oplus b_1\right)^{g_{(0,\ldots,0)}} = \begin{pmatrix} \gamma_1^L \oplus a_1^{Q_{(0,\ldots,0)}SL} \oplus b_1, a_1 \end{pmatrix}, \\ \left(a_2, \gamma_2^L \oplus b_2\right)^{g_{(0,\ldots,0)}} = \begin{pmatrix} \gamma_2^L \oplus a_2^{Q_{(0,\ldots,0)}SL} \oplus b_2, a_2 \end{pmatrix} \end{cases} \tag{2}$$

By theorem 1 for any $N_0 \subseteq N\left(\overline{a}_1 \oplus \overline{a}_2\right)$ the following inequality holds

$$\max_{\gamma : N(\gamma) \subseteq N_0} \left| N_{L(N_0)}\left(\gamma^L \oplus \beta\right) \right| \geq |N_0|,$$

where $\gamma = \gamma_1 \oplus \gamma_2, \beta = a_1^{Q_{(0,\ldots,0)}SL} \oplus a_2^{Q_{(0,\ldots,0)}SL} \oplus b_1 \oplus b_2$. In particular, by corollary 2, there is $\gamma \in V_{mn}, N(\gamma) \subseteq N_0$, such that $\left|N\left(\gamma^L \oplus \beta\right)\right| \geq |N_0|$.

In case 1 of the theorem denote $N_0 = N\left(\overline{a}_1 \oplus \overline{a}_2\right)$. Then $\left|N\left(\gamma^L \oplus \beta\right)\right| \geq |N_0| > |N(a_1 \oplus a_2)|$, that is we've got a new pair $\left(\delta_1, a_1\right), \left(\delta_2, a_2\right)$ such that $\left|N\left(\delta_1 \oplus \delta_2\right)\right| > \left|N\left(a_1 \oplus a_2\right)\right|$. Repeating the described procedure eventually we obtain a new pair $\left(a_1', b_1'\right), \left(a_2', b_2'\right)$ such that $N\left(a_1', \oplus a_2'\right) = N\left(b_1', \oplus b_2'\right) = \overline{1, n}$.

In case 2 of theorem denote $N_0 = N\left(a_1 \oplus a_2\right)$. Then $\left|N_{L(N_0)}\left(\gamma^L \oplus \beta\right)\right| \geq |N_0|$. Denote $N_1 = N_{L(N_0)}\left(\gamma^L \oplus \beta\right)$ and repeat the described procedure using set N_1 instead of N_0, so we obtain set N_2. Repeating it for N_2, N_3 and so on, we obtain the sequence of sets of non-decreasing power. Since power of this sets is bounded by n, then there are numbers t, r such that $N_{r+t} = N_r$. From the construction of sets we have $|N_r| \leq |N_{r+1}| \leq \ldots |N_{r+t}|$. From this and from equality $N_{r+t} = N_r$ we get

$$|N_r| = |N_{r+1}| = \ldots |N_{r+t}|$$

If for some $l \in \overline{1, t}$ the inequality $|L(N_{r+l-1})| > |N_{r+l-1}|$ does hold, then from theorem 1 we have $|N_{r+l-1}| < |N_{r+l}|$, which is wrong. So $|L(N_{r+l-1})| = |N_{r+l-1}| = |N_{r+l}|$. By definition, $L(N_{r+l-1}) \supseteq N_{r+l}$ and $L(N_{r+l-1}) = N_{r+l}$. From this we have

$$N_{r+l} = L(N_{r+l-1}) = \cdots = L^l(N_r), l \in \overline{1, t}.$$

Since $N_{r+t} = L^t(N_r) = N_r$, then $\left|L^l(N_r)\right| = |N_r|$ for any $l \geq 0$. Set $L^l(N_r)$ is the set of tails of paths in graph Γ_L of length $l = 1, 2 \ldots$ with heads in set N_r. Condition $\mathbf{FSA_1}$ is equivalent to the following: for certain natural d there is a path of length d between any pair of nodes in graph Γ_L. Then $L^d(N_r) = \{1, 2, \ldots, n\}$ and we obtain a pair $(a'_1, b'_1), (a'_2, b'_2)$ such that $N(a'_1 \oplus a'_2) = N(b'_1 \oplus b'_2) = V$ as well.

Transitions between pairs $(a'_1, b'_1), (a'_2, b'_2)$ and $(c'_1, d'_1), (c'_2, d'_2)$ for $N(a'_1, \oplus a'_2) = N(b'_1, \oplus b'_2) = N(c'_1, \oplus c'_2) = N(d'_1, \oplus d'_2) = \overline{1, n}$ can be performed using formulas 1 and 2 since then we can get an arbitrary pair $\gamma_1, \gamma_2 \in V_{mn}$ by choosing the appropriate keys.

If initial blocks have equal first subblocks, then we apply one-round transformation with an arbitrary key and get new blocks with distinct first subblocks.

We have shown that group R_F is 2-transitive. In order to show that it equals to alternating group we need the following lemma.

Lemma 2 ([1]). *Let R be 2-transitive group of even permutations of vector space V_l. The either $R = A(V_l)$, or R is conjugate to a subgroup of general affine group $AGL(V_l)$, or $q = 2^l - 1$ is prime and R is similar to the certain subgroup of normalizer $N(H)$ of group $H = PSL(2, q)$ in symmetric group $S(M)$ of permutations of set M of one-dimensional subspaces of two-dimensional vector space over $\mathbb{F}_q$.*

Ciphers we are looking at are based on Feistel network, therefore permutations generated by one-round transformations are even. In our case $l = 2mn$, $2^{2mn} - 1$ is not prime and third option of lemma is impossible. Let's have a look at the second option. It's easy to verify that permutation $g_{(0,0,\ldots,0,k_1)} g^{-1}_{(0,0,\ldots,0,k_2)}$ with $k_1 \neq k_2$, $k_1, k_2 \in V_{mn}$, has no fixed points. Indeed, for any $a, b \in V_{mn}$:

$$(a, b)^{g_{(0,0,\ldots,0,k_1)} g^{-1}_{(0,0,\ldots,0,k_2)}} = \left(a, a^{Q_{(0,0,\ldots,0)} \pi_{k_1} SL} \oplus a^{Q_{(0,0,\ldots,0)} \pi_{k_2} SL} \oplus b\right)$$

If (a, b) is a fixed point of $g_{(0,0,\ldots,0,k_1)} g^{-1}_{(0,0,\ldots,0,k_2)}$, then

$$b = a^{Q_{(0,0,\ldots,0)} \pi_{k_1} SL} \oplus a^{Q_{(0,0,\ldots,0)} \pi_{k_2} SL} \oplus b.$$

And since Q, π_{k_i}, S and L are permutations, then we have $k_1 = k_2$.

Let's find transformation $A \in AGL(V_{2mn})$ without fixed points. Permutation A performs transformation $x \mapsto Cx \oplus \beta$ of space V_{2mn} , where $C \in GL(V_{2mn})$, $\beta \in V_{2mn}$. All fixed points satisfy the equation $(C \oplus E)x = \beta$. This equation doesn't have solutions if $C = E$, $\beta \neq 0$, that is if A performs mapping $x \mapsto x \oplus \beta$.

Let's estimate the order of group R_F. We have $G = \langle g_{k^{(1)}} g_{k^{(2)}}^{-1}, k^{(1)} | k^{(1)}, k^{(2)} \in V_{mn}^l \rangle \subseteq R_F$. It's obvious that $|G| \geq \left|V^{(a_1,a_2)}\right|^n$, and from **FSA₂** we have

$$|G| \geq \left|V^{(a_1,a_2)}\right| = 2^{2mn}.$$

Permutation $g_{(0,\ldots,0,k_1)}^{-1} g_{(0,\ldots,0,k_1)} \in R$ performs transformation of vector $(a,b) \in V_{2mn}$ into $\left(b^{Q_{(0,\ldots,0)}\pi_{k_1} SL} \oplus b^{Q_{(0,\ldots,0)}\pi_{k_2} SL} \oplus a, b\right)$, so $g_{(0,\ldots,0,k_1)}^{-1} g_{(0,\ldots,0,k_2)} \in R$ with $k_1 \neq k_2$, therefore $|R_F| \geq |G| + 1 > 2^{2mn} = |\langle x \mapsto x \oplus \beta | \beta \in V_{2mn} \rangle|$, and group $G = \langle g_{k^{(1)}} g_{k^{(2)}}^{-1}, k^{(1)} | k^{(1)}, k^{(2)} \in V_{mn}^l \rangle$ doesn't conjugate to a subgroup of translation group $\langle x \mapsto x \oplus \beta | \beta \in V_{2mn} \rangle$. Hence second option of lemma 2 is impossible as well and group R_F equals to the alternating group of degree 2^{2mn}. ■

Now we provide an example of case 1 of theorem 2.

Statement 2. *Let function $Q_{(k_1,\ldots,k_{l-1})}$ can be represented in the following form*

$$Q_{(k_1,\ldots,k_{l-1})} = {Q'}_{(k_1,\ldots,k_{l-2})} \pi_{k_{l-1}} S' L'$$

where $Q' : V_{mn} \times V_{mn}^{l-2} \to V_{mn}, {Q'}_{(k_1,\ldots,k_{l-2})}(x) = Q'(x, k_1, \ldots, k_{l-2})$ and the following conditions are satisfied:

1) for any $a_1, a_2 \in V_{mn}, a_1 \neq a_2$ there are $k_1, \ldots, k_{l-2} \in V_{mn}$ such that

$$\left| N\left(a_1^{{Q'}_{(k_1,\ldots,k_{l-2})}} \oplus a_2^{{Q'}_{(k_1,\ldots,k_{l-2})}} \right) \right| \geq N(a_1 \oplus a_2);$$

*2) L' satisfies the condition **FSA₃**;*

*3) S' satisfies the condition **FSA₄**.*

Then $Q_{(k_1,\ldots,k_{l-1})}$ satisfies case 1 of theorem 2.

Proof. Let's take an arbitrary $a_1, a_2 \in V_{mn}, a_1 \neq a_2$. Choose $k_1, \ldots, k_{l-2}$ such that

$$\left| N\left(a_1^{{Q'}_{(k_1,\ldots,k_{l-2})}} \oplus a_2^{{Q'}_{(k_1,\ldots,k_{l-2})}} \right) \right| \geq \left| N(a_1 \oplus a_2) \right|,$$

and define $\overline{a}_1 = a_1^{{Q'}_{(k_1,\ldots,k_{l-2})}}$, $\overline{a}_2 = a_2^{{Q'}_{(k_1,\ldots,k_{l-2})}}$, $\overline{a}_i = (\overline{a}_{i,1}, \ldots, \overline{a}_{i,n}), \overline{a}_{i,j} \in V_m$, $i \in \{1,2\}$,$j \in \overline{1,n}$. We show that we can choose k_{l-1} such that

$$\left| N\left(\overline{a}_1^{\pi_{k_{l-1}} S'L'} \oplus \overline{a}_2^{\pi_{k_{l-1}} S'L'} \right) \right| > \left| N(\overline{a}_1 \oplus \overline{a}_2) \right|$$

if $|N(\overline{a}_1 \oplus \overline{a}_2)| < n$, and

$$\left| N\left(\overline{a}_1^{\pi_{k_{l-1}} S'L'} \oplus \overline{a}_2^{\pi_{k_{l-1}} S'L'} \right) \right| = \left| N(\overline{a}_1 \oplus \overline{a}_2) \right|$$

if $|N(\overline{a}_1 \oplus \overline{a}_2)| = n$.

Let $u = |N(\overline{a}_1 \oplus \overline{a}_2)|$. Without loss of generality we assume that $N(\overline{a}_1 \oplus \overline{a}_2) = \{1, \ldots, u\}$ and $\bigcup_{j=1}^{u} M_j(L') = \{1, \ldots, t(u)\}$ for certain $t(u) \in \overline{1, n}$.

By the condition of statement we have $t(u) > u$ if $u < n$ and $t(u) = n$ if $u = n$. For any $c = (c_1, \ldots, c_n) \in V_{mn}$ define $(\delta_1(c_1), \ldots, \delta_n(c_n)) = \overline{a}_1^{\pi_c S'} \oplus \overline{a}_2^{\pi_c S'}$, where $\delta_i(c_i) = s(\overline{a}_{1,i} \oplus c_i) \oplus s(\overline{a}_{2,i} \oplus c_i) \in V_m, i \in \overline{1, n}$ and $\Delta_i(c_i) = (0, \ldots, 0, \delta_i(c_i), 0 \ldots, 0)$. For $j \in M_i(L')$ j-th coordinate of vector $\Delta_i(c_i)^{L'}$ equals to $\delta_i(c_i)^{L'_{ij}}$. Since matrix L_{ij} is non-singular, then for any $e \in V_m$ there is a single $\delta_i(c_i)$ such that $\delta_i(c_i)^{L'_{ij}} = e$. It means that for any vector $E \in V_{mn}$ there are no more than $|M_i(L')|$ values of $\delta_i(c_i)$ such that for at least one $j \in M_i(L')$ j-th coordinates of vectors $\Delta_i(c_i)^{L'}$ and E are equal. By condition, the number of distinct $\delta_i(c_i)$ are greater then $|M_i(L')|$ and so there are $\delta_i(c_i)$ such that vectors $\Delta_i(c_i)^{L'}$ and E are not equal in every coordinate with index from the set $M_i(L')$.

Since

$$(\delta_1(c_1), \ldots, \delta_u(c_u), 0, \ldots, 0)^{L'} = \sum_{i=1}^{u} \Delta(c_i)^{L'} =$$

$$= \left(\ldots\left(\left(\Delta_1(c_1)^{L'}\right) \oplus \Delta_2(c_2)^{L'}\right) \oplus \ldots\right) \oplus \Delta_u(c_u)^{L'}$$

then we can successively choose values $\delta_1(c_1), \ldots, \delta_u(c_u)$ such that vector $(\delta_1(c_1), \ldots, \delta_u(c_u))^{L'}$ has exactly $t(u) = \left|\bigcup_{j=1}^{u} M_j(L')\right|$ non-zero coordinates.

Now we only need to set $k_{l-1} = (c_1 \ldots, c_u, 0, \ldots, 0)$ ■

3 Examples: MIBS, E2 and Camelia ciphers

As an example let's look at MIBS, E2 and Camelia ciphers and show that their one-round functions generate alternating groups.

MIBS cipher [2] has 64-bit block, parameters $n = 8$, $m = 4$, and round function $F_k(x) = x^{\pi_k SL}$. Permutation s is the following permutation of V_4: $(4, 15, 3, 8, 13, 10, 12, 0, 11, 5, 7, 14, 2, 6, 1, 9)$. Linear transformation L consists of two parts – multiplication by matrix and permutation of 4-bit blocks.

Matrix L of compositions of this parts has the following form

$$\begin{pmatrix} E & O & E & O & E & E & E & E \\ E & E & E & E & O & E & E & E \\ O & E & E & E & E & O & E & E \\ E & E & O & E & E & E & O & O \\ E & E & E & O & E & E & O & O \\ O & E & E & O & O & E & E & E \\ E & E & O & E & O & O & E & E \\ E & O & E & E & E & O & O & E \end{pmatrix}$$

where E is identity matrix 4×4, O is null matrix 4×4.

In order to check the condition **FSA₂** for permutation s we ran through values $k_1, \ldots, k_4 \in V_4$ and computed vectors $\left(0^{\pi_{k_1} s} \oplus 0^{\pi_{k_2} s} \oplus 0^{\pi_{k_3} s} \oplus 0^{\pi_{k_4} s}, a_2^{\pi_{k_1} s} \oplus a_2^{\pi_{k_2} s} \oplus a_2^{\pi_{k_3} s} \oplus a_2^{\pi_{k_4} s}\right)$ for every $a_2 \in V_4$. It was determined that for every $a_2 \in V_4$ the derived set of vectors equals to V_8. In order to check the condition **FSA₁** for matrix L we constructed the part of graph Γ_L, it is shown in fig. 1. Since graph Γ_L has loops and is

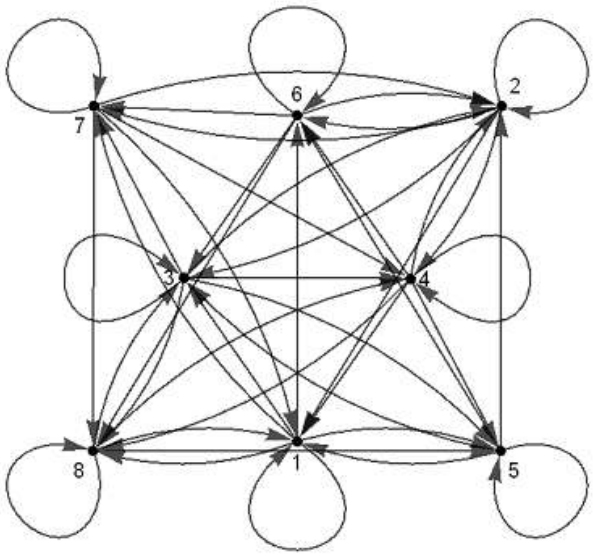


Figure 1: Fragment of graph Γ_L for MIBS

strongly connected, then matrix L satisfies the condition **FSA₁**. So round function of MIBS satisfies case 2 of theorem 2 with $Q_{(k_1,\ldots,k_{l-1})}(x) = x$ and the corresponding group equals to $A_{2^{64}}$.

E2 cipher [3] has block size 120 bit, parameters $n = 8$, $m = 8$, and round function $F_k(x) = x^{\pi_{k_1} S' L' \pi_{k_2} S L}$, $k = (k_1, k_2) \in V_{128}$. Permutations

s and s' of transformations S and S' respectively are equal and perform transformation $x \to s(x)$, $s(x) = 97 \cdot x^{127} + 225$, where the exponentiation is performed in $GF(256)$, and summing and multiplication – modulo 256. Linear transformation L' represents multiplication of vector from V_{mn} by the following matrix

$$L' = \begin{pmatrix} O & E & E & E & E & E & E & O \\ E & O & E & E & O & E & E & E \\ E & E & O & E & E & O & E & E \\ E & E & E & O & E & E & O & E \\ E & E & O & E & E & E & O & O \\ E & E & E & O & O & E & E & O \\ O & E & E & E & O & O & E & E \\ E & O & E & E & E & O & O & E \end{pmatrix}$$

where E is identity matrix 8×8, O is null matrix 8×8. Besides, there is transformations IT before the first round and transformation FL after the last round, which we ignore.

It's obvious that matrix L' satisfies the condition **FSA₃**. In order to check the condition **FSA₂** for permutation s we ran through values $k_1, \ldots, k_4 \in V_8$ and computed vectors $\left(0^{\pi_{k_1} s} \oplus 0^{\pi_{k_2} s} \oplus 0^{\pi_{k_3} s} \oplus 0^{\pi_{k_4} s}, a_2^{\pi_{k_1} s} \oplus a_2^{\pi_{k_2} s} \oplus a_2^{\pi_{k_3} s} \oplus a_2^{\pi_{k_4} s}\right)$ for every $a_2 \in V_8$. It was determined that for every $a_2 \in V_8$ the derived set of vectors equals to V_{16}. The differential characteristic p_s of permutation s equals to $\frac{10}{256}$, so $25.6 = p^{-1} > \max\limits_{i \in \overline{1,n}} |M_i(L)| = 6$ and condition **FSA₄** is satisfied as well. Summarizing, round of E2 cipher satisfies the conditions of statement 2 for $Q'_{(k_1,\ldots,k_{l-2})}(x) = x$ and case 1 of theorem 2 and the corresponding group equals to $A_{2^{128}}$

Camelia cipher [4] was created using the ideas of E2. It also has parameters $n = 8$, $m = 8$, and round function $F_k(x) = x^{\pi_k SL}$, $k \in V_{64}$. After every six iterations left subblock is being transformed with FL operation, and right subblock - by FR operation. Matrix L is tensor product of nonsingular $(0,1)$-matrix of order 8 and identity matrix of order 8. The diffusion graph of $(0,1)$-matrix if strongly connected and has a loop, so it satisfies the condition **FSA₁**. The S transformation uses 4 different permutations of V_8. The experiment showed that they satisfy the condition **FSA₂**. Hence, iteration transformation of Camelia satisfies the conditions of case 2 of theorem 2 for $Q_{(k_1,\ldots,k_{l-1})}(x) = x$ and the corresponding group equals to $A_{2^{128}}$.

In conclusion we underscore that the above conditions are not necessary for round function to generate alternating group. For example, the round

function of DES does generate alternatig group [5], but its non-linear part are not permutations, which is suffictient for our proof.

References

[1] A. S. Maslov, "On sufficient conditions to generate the alternating group by SA-permutations.", *Trudy Instituta Matematiki*, **15**:2 (2007), 58 — 68, In Russian.

[2] Maryam Izadi, Babak Sadeghiyan, Seyed Saeed Sadeghian, Hossein Arabnezhad Khanooki, "MIBS: A New Lightweight Block Cipher.", *LNCS*, **5888** (2009), 334 — 348.

[3] Wentan Yi, Shaozhen Chen, "Integral Cryptanalysis of the Block Cipher E2.", arXiv: pdf/405.6483v2.

[4] Kazumaro Aoki, Tetsuya Ichikawa, Masayuki Kanda, Mitsuru Matsui, Shiho Moriai, Junko Nakajima, Toshio Tokita, *Camelia: A 128-bit Block Cipher Suitable for Multiple Platforms.*, 2002, http://info.isl.ntt.co.jp/crypt/eng/camellia/technology/.

[5] Ralph Wernsdorf, "The One-Round Functions of the DES Generate the Alternating Group.", *R.A. Rueppel (Ed.): Advances in Cryptology - EUROCRYPT '92, LNCS 658*, 1993, 99 — 112.

Appendix A. Proof of Theorem 1

Theorem 1 For any $m > 2$, set $I \subseteq V$ and vector $\beta \in V_{mn}$ the following inequality holds

$$\max_{\alpha: N(\alpha) \subseteq I} \left| N_{L(I)} \left(\alpha^L \oplus \beta \right) \right| \geq |I|.$$

This inequality is strict if $|L(I)| > |I|$.

Proof. We start from the second part. Let $|L(I)| > |I|$. Denote

$$\Gamma_1 = \{\gamma \in V_{mn} \setminus \{0\} | N(\gamma) \subseteq I\},$$

$$\Gamma_2 = \left\{\gamma \in V_{mn} \setminus \{0\} | N(\gamma) \subseteq I, \left| N_{L(I)} \left(\gamma^L \oplus \beta \right) \right| \leq |I| \right\},$$

$$n_i = |\Gamma_i|.$$

We need to show that $n_2 < n_1$, then $\Gamma_1 \setminus \Gamma_2 \neq \varnothing$ and for any $\gamma \in \Gamma_1 \setminus \Gamma_2$ inequality $\left| N_{L(I)} \left(\gamma^L \oplus \beta \right) \right| > |I|$ holds.

We have $n_1 = 2^{m|I|} - 1$. Without loss of generality we assume that $I = \{1, \ldots, |I|\}$, $L(I) = \{1, \ldots, |L(I)|\}$. Let c be $(m|I| \times mn)$-submatrix of matrix L constructed from rows with numbers $1, \ldots, m|I|$. Divide matrix c into n fragments with m consecutive columns in each one. Since $\operatorname{rang} c = m|I|$, then we can pick $m|I|$ linearly independent columns of matrix c.

Let's show that these columns can be picked from $|I| + 1$ fragments at least. Assume that we have picked columns from $|I|$ fragments. Denote by $c_j^{\downarrow}$ the jth columns of matrix c, by T – the set of numbers of picked columns. Since $I = \{1, \ldots, |I|\}$ and $N(\gamma) \subseteq I$, we need to consider only first $m|I|$

columns of matrix L to describe the action of transformation L on γ, that is, matrix c. At the same time, an image of every vector γ is fully defined by projection of vector γ^L on coordinates from T. Let c_T be the respective linear transformation.

Since $n \geq |L(I)| > |I|$ then there is at least one non-zero column $c_j^{\downarrow}$, $j \notin T$, such that

$$c_j^{\downarrow} = \sum_{i \in T} \alpha_i c_i^{\downarrow}$$

where coefficients $\alpha_i \in GF(2)$ are not all zeroes at the same time. Let $\alpha_i \neq 0$ for certain i. We replace column $c_i^{\downarrow}$ with $c_j^{\downarrow}$ (and i with j in set T respectively). Since $m \geq 2$, then we obtain a set of $m|I|$ linearly independent columns from at least $|I| + 1$ fragments.

Denote by m_i the number of columns of matrix c picked from ith fragment, $m_i \leq m$, $i \in \overline{1,n}$. For simplicity we assume that $m_i > 0$ for $i \in \overline{1,t}$ and $m_i = 0$ for $i \in \overline{t+1,n}$. Inequality $t < 2$ corresponds to the trivial case $I = \varnothing$, which obviously satisfies the theorem, so we assume $t \geq 2$. We have $m_1 + \cdots + m_t = m|I|$. Denote by $\bar{\beta}$ the projection of vector β on coordinates T and by $\bar{\beta}_i$ the projection of β on coordinates $\{(i-1)m+1, \ldots, (i-1)m+m\} \cap T$.

Mapping c_T is injective on set Γ_2 and different images correspond to different elements of set

$$\Lambda(m_1, \ldots, m_n) = \{(\lambda_1, \ldots, \lambda_t) | \lambda_i \in V_{m_i}, w(\lambda_1) + \cdots + w(\lambda_t) \leq |I|\} \subseteq V_{m|I|},$$

where $w(\lambda_i)$ is the indicator of the event $(\lambda_i \neq \bar{\beta}_i)$. It means that $n_2 \leq |\Lambda_I(m_1, \ldots, m_n)|$. Number of elements of set $\Lambda(m_1, \ldots, m_n)$ is fully defined by values $m_1, \ldots, m_n$ and $|I|$. Denote

$$n_2(m_1, \ldots, m_n) = |\Lambda_I(m_1, \ldots, m_n)|.$$

Our goal is to show that

$$n_2(m_1, \ldots, m_n) \leq n_2(\underbrace{m, \ldots, m}_{|I|-1}, m-1, 1, 0, \ldots, 0). \tag{3}$$

In this case

$$n_2(\underbrace{m, \ldots, m}_{|I|-1}, m-1, 1, 0, \ldots, 0) \leq |V_m|^{|I|-1} |V_{m-1}| |V_1| -$$

$$- (|V_m| - 1)^{|I|-1} (|V_{m-1} - 1|) (|V_1| - 1) = 2^{m|I|} - (2^m - 1)^{|I|-1} (2^{m-1} - 1).$$

And so we obtain the upper bound for any T with condition $t \geq |I| + 1$. Assume that $m > m_i \geq m_j > 0$ for certain i, j, for example, $i = 1, j = 2$. Let's show that then

$$n_2(m_1, \ldots, m_n) \leq n_2(m_1 + 1, m_2 - 1, m_3, \ldots m_n). \tag{4}$$

Let $\lambda = (\lambda_1, \ldots, \lambda_t) \in V_{m|I|}, \lambda_i \in V_{m_i}$.

Case 1: $m_2 = 1, t > 2$. Vectors λ and $\bar{\beta}$ can be written as $\lambda = (\lambda_1^*, \lambda_3, \ldots, \lambda_t)$, $\bar{\beta} = \left(\bar{\beta}_1^*, \bar{\beta}_3, \ldots, \bar{\beta}_t\right)$, where $\lambda_t^* = (\lambda_1, \lambda_2)$, $\bar{\beta}_t^* = \left(\bar{\beta}_1, \bar{\beta}_2\right)$. Then

$$\Lambda^* \left\{(\lambda_1^*, \lambda_3, \ldots, \lambda_t) | w(\lambda_1^*) + w(\lambda_3) + \cdots + w(\lambda_t) \leq |I|\right\} \supseteq \Lambda_I(m_1, \ldots, m_n),$$

since $w(\lambda_1) + w(\lambda_2) \geq (\lambda_1^*)$. Therefore,

$$n_2(m_1, \ldots, m_n) \leq |\Lambda^*| = n_2(m_1 + 1, m_2 - 1, m_3, \ldots, m_n).$$

Case 2: $m_2 = 1, t = 2$. This case is possible if only $|I| = 1$. But then inequality (3) turns into equality since both of its parts has the same value.

Case 3: $m_2 > 1, t > 2$. If α is the first coordinate of vector λ_2, then we remove it from λ_2 and add it to λ_1. We do the same with vector $\bar{\beta}_2$. Then vectors λ and $\bar{\beta}$ will be written as $\lambda = (\lambda_1^*, \lambda_2^*, \lambda_3, \ldots, \lambda_t)$, $\bar{\beta} = (\bar{\beta}_1^*, \bar{\beta}_2^*, \bar{\beta}_3, \ldots, \bar{\beta}_t)$, where $\lambda_1^*, \bar{\beta}_1^* \in V_{m_1+1}$, $\lambda_2^*, \bar{\beta}_2^* \in V_{m_2-1}$. Let

$$\Lambda^* = \left\{(\lambda_1^*, \lambda_2^*, \lambda_3, \ldots, \lambda_t) | w(\lambda_1^*) + w(\lambda_2^*) + w(\lambda_3) + \cdots + w(\lambda_t) \leq |I|\right\}.$$

At the same time $|\Lambda^*| = n_2(m_1 + 1, m_2 - 1, m_3, \ldots, m_n)$.

Note that containment $\lambda \in \Lambda^* \setminus \Lambda_I(m_1, \ldots, m_n)$ is equivalent to the following: α doesn't equal to the first coordinate of $\bar{\beta}_2$, $\lambda_1 \neq \bar{\beta}_1$, $\lambda_2^* = \bar{\beta}_2^*$ and $w(\lambda_3) + \cdots + w(\lambda_t) = |I| - 1$. There are $|\Lambda^* \setminus \Lambda_I(m_1, \ldots, m_n)| = (2^{m_1} - 1)d$ vectors λ with this restriction, where

$$d = \left|\left\{(\lambda_3, \ldots, \lambda_t) | \lambda_3 \in V_{m_3}, \ldots, \lambda_t \in V_{m_t}, w(\lambda_3) + \cdots + w(\lambda_t) = |I| - 1\right|\right\}.$$

Similarly to this, $\lambda \in \Lambda_I(m_1, \ldots, m_n) \setminus \Lambda^*$ iif α doesn't equal to the first coordinate of $\bar{\beta}_2$, $\lambda_1 = \bar{\beta}_1^*$, $\lambda_2^* \neq \bar{\beta}_2^*$ and $w(\lambda_3) + \cdots + w(\lambda_t) = |I| - 1$, and

$$|\Lambda_I(m_1, \ldots, m_n) \setminus \Lambda^*| = (2^{m_2 - 1} - 1)d \leq |\Lambda^* \setminus \Lambda_I(m_1, \ldots, m_n)|.$$

Hence moving a coordinate in vector from $\Lambda_I(m_1, \ldots, m_n)$ doesn't decrease the number of proper vectors and (3) holds.

Case 4: $m_1 > 1, t = 2$. Proving the inequality (3) is similar to proving it in case $m_2 > 1, t > 2$. Sets $\Lambda_I(m_1, m_2)$ and Λ^* are

$$\Lambda_I(m_1, m_2) = \left\{(\lambda_1, \lambda_2) | w(\lambda_1) + w(\lambda_2) \leq |I|\right\}$$

$$\Lambda^* = \{(\lambda_1^*, \lambda_2^*) | w(\lambda_1^*) + w(\lambda_2^*) \leq |I|\}$$

Using (4) we obtain the required estimation (3). So we have the inequality

$$n_2 \leq n_2(m_1, \ldots, m_n) \leq 2^{m|I|} - (2^m - 1)^{|I|-1}(2^{m-1} - 1).$$

The right part of it is less then $n_1 = 2^{m|I|} - 1$ when $m \geq 3$, so the proof of the second part of the theorem is done.

To prove the first part we need only to show that

$$n_3 = \left|\left\{\gamma | N(\gamma) \subseteq I, \left|N_{L(I)}\left(\gamma^L \oplus \beta\right)\right| < |I|\right\}\right| < n_1.$$

Indeed, we have

$$n_3 = 2^{m|I|} - (2^m - 1)^{|I|} < n_1.\blacksquare$$

An Algorithm for Bounding Non-minimum Weight Differentials in 2-round LSX-ciphers

Vitaly Kiryukhin

JSC «InfoTeCS», LLC «SFB Lab», Russia
Vitaly.Kiryukhin@infotecs.ru

Abstract

This article describes some approaches to bounding non-minimum weight differentials (EDP) and linear hulls (ELP) in 2-round LSX-cipher. We propose a dynamic programming algorithm to solve this problem. For 2-round Kuznyechik the nontrivial upper bounds on all differentials (linear hulls) with 18 and 19 active Sboxes was obtained. These estimates are also holds for other differentials (linear hulls) with a larger number of active Sboxes. We obtain a similar result for 2-round Khazad. As a consequence, the exact value of the maximum expected differential (linear) probability (MEDP/MELP) was computed for this cipher.

Keywords: Kuznyechik, Khazad, SPN, LSX, differential cryptanalysis, linear cryptanalysis, MEDP, MELP

1 Introduction

Differential [2] and linear [3] cryptanalysis are the two most known statistical attacks applicable to block ciphers. In this paper we will focus on the first method. The analogous results for linear cryptanalysis will be obtained in a similar way, due to the existing well-known duality [4].

There are several approaches to estimating the security of ciphers against differential attacks. Many papers are devoted to the differential characteristics. The maximal probability of such characteristics (EDCP) decreases when the number of active Sboxes within R rounds increases. The upper bound on such probability can be analytically obtained for many LSX-ciphers (AES [11], Khazad [12], Kuznyechik [1], etc.). In particular, these results are presented in [11, 17].

However, many researchers note that differential cryptanalysis exploits differentials and not characteristics (see for example [16, 14, 5]). The probability (EDP) of a differential $(\Delta x, \Delta y)$ corresponds to the sum of the probabilities of all characteristics with input difference Δx and output difference Δy [8]. So from this point of view security of a cipher against differential

attacks is based on the maximum expected differential probability (MEDP) over $R \geq 2$ rounds.

Related works. For 2-round LSX-ciphers, some approaches to computing upper bounds on the MEDP are known [13, 14, 15].

An algorithm for computing the exact MEDP of 2-round AES was proposed in [5]. Article [10] describes upper bounds on the MEDP for so-called «nested» LSX-ciphers (e.g. 4-round AES).

In [16] was shown that for some 2-round LSX-ciphers the MEDP is achieved by differentials involving a number of active Sboxes which exceeds the branch number of the linear layer (non-minimum weight differentials).

Some results about differential properties of 2-round Kuznyechik was obtained in [18]. The cited paper contains an algorithm for constructing the best minimum weight differentials and a proof that all other differentials have a lower EDP. Thanks to these two results, the exact value of the 2-round MEDP was computed.

Our contribution. We propose a dynamic programming algorithm designed for bounding non-minimum-weight differentials in 2-round LSX-ciphers. It uses only the difference distribution table and the differential branch number of the linear layer. The algorithm minimizes the number of high probability differential trails and does not try to minimize the total number of trails. Because of this reason, the algorithm is not effective for ciphers with small block size (for example, 32-bit 2-round AES).

We applied the developed algorithm to the 2-round Kuznyechik (Section 4 and Appendix B): the probability of any 2-round differential (linear hull) with $n + 3 = 19$ active Sboxes is bounded by $2^{-88.34}$ ($2^{-79.63...}$ correspondingly). These bounds also holds for any differential (linear hull) with $a \geq n+3$ active Sboxes. Similar results were obtained for 2-round Khazad (Appendix C), and as a result, the exact values of $\mathrm{MEDP} = 2^{-45} + 2^{-60}$ and $\mathrm{MELP} = 2^{-37.80...}$ are also proved.

The set of estimates obtained by us can be used in further researches to calculate the bounds on the MEDP (MELP) for more rounds. We plan to use our new results together with a modified KMT2-DC (KMT2) algorithm [6, 7]. The approach [7] allows to incorporate other upper bounds when those bounds are superior to the values determined directly by the original algorithm [6]. In this way, we hope to prove the greater security of Kuznyechik to differential and linear cryptanalysis.

2 Notations and definitions

An LSX cipher E consists of sequence of rounds. Each of them contains three operations: X – modulo 2 addition of an input block with an iterative key; S – parallel application of a fixed bijective substitution; L – linear transformation which can be represented as multiplication by the binary matrix.

To simplify the text and notations, we consider only byte-oriented LSX-ciphers.

Denote: n – block size in bytes; $\oplus$ – bitwise XOR operation; $\mathbf{x}[i]$ – i-th element of a vector or a sequence $\mathbf{x}$, $1 \leq i \leq l$, where l is size of the $\mathbf{x}$; $\mathrm{Supp}(\mathbf{x}) = \{i : \mathbf{x}[i] \neq 0\}$ – the support of a vector $\mathbf{x}$; $\mathrm{wt}(\mathbf{x}) = \#\{i : \mathbf{x}[i] \neq 0\}$ – the weight of a vector $\mathbf{x}$;

$\mathbf{F}_q$ – finite field of q elements; $\mathbf{F}_q^*$ – all nonzero elements of a field $\mathbf{F}_q$; $\mathbf{F}_q^l$ – vector of l elements over $\mathbf{F}_q$. Depending on the context, we will interpret a value $x \in \overline{0, 2^l - 1}$ as element of $\mathbf{F}_{2^l}$ or $\mathbf{F}_2^l$ or as integer.

Definition 1. *Let $f : \mathbf{F}_2^l \to \mathbf{F}_2^l$, let Δx, $\Delta y \in \mathbf{F}_2^l$ be fixed, and let $x \in \mathbf{F}_2^l$ be a uniformly distributed random variable. The differential probability is*

$$\mathrm{DP}(\Delta x, \Delta y) = \Pr(f(x) \oplus f(x \oplus \Delta x) = \Delta y). \tag{1}$$

Definition 2. *Let E be a cipher with key-size κ. Then, the expected probability of differential $(\Delta x, \Delta y)$ is*

$$\mathrm{EDP}(\Delta x, \Delta y) = 2^{-\kappa} \sum_{K \in \mathbf{F}_2^\kappa} \Pr(E_K(x) \oplus E_K(x \oplus \Delta x) = \Delta y),$$

where E_K is a cipher with fixed key K. We futher assume that all round keys are independent and uniformly distributed.

Definition 3. *The maximum expected differential probability is*

$$\mathrm{MEDP} = \max_{\Delta x \neq 0, \Delta y} \mathrm{EDP}(\Delta x, \Delta y)$$

Definition 4. *Let s be a function from $\mathbf{F}_2^8 \to \mathbf{F}_2^8$,*

$$\delta(a, b) = \#\left\{x \in \mathbf{F}_2^8,\ \mathsf{s}(x) \oplus \mathsf{s}(x \oplus a) = b\right\}, \forall a, b \in \mathbf{F}_2^8.$$

$\delta_{\max} = \max\limits_{a \neq 0, b} \delta(a, b)$ *is the differential uniformity of s, $p_{\max} = 2^{-8} \cdot \delta_{\max}$. The differential distribution table is a $2^8 \times 2^8$ matrix of transition probabilities such that*

$$\mathrm{DDT}[a][b] = \frac{\delta^{\mathsf{s}}(a, b)}{2^8} = \mathrm{DP}(a, b),\ a, b \in \mathbf{F}_2^8.$$

Definition 5. *Let* L*-transformation (from* $\mathbf{F}_{2^8}^n$ *to* $\mathbf{F}_{2^8}^n$*) be a* $\mathbf{F}_{2^8}$*-linear. We associate with* L *the code* $\mathcal{C}_{\mathsf{L}}$ *of length* $2n$ *over* $\mathbf{F}_{2^8}$ *defined by*

$$\mathcal{C}_{\mathsf{L}} = \{(\mathbf{c}, \mathsf{L}(\mathbf{c}))\,,\ \mathbf{c} \in \mathbf{F}_{2^8}^n\}\,.$$

The differential branch number $\mathcal{B}_{\mathsf{L}}$ *of the linear transformation* L *is the minimum distance of the code* $\mathcal{C}_{\mathsf{L}}$

$$\mathcal{B}_{\mathsf{L}} = \min_{\mathbf{c}\neq 0} \mathrm{wt}\,(\mathbf{c}, \mathsf{L}(\mathbf{c}))\,.$$

Further, to simplify the text, we assume that $\mathcal{C}_{\mathsf{L}}$ is an MDS code and $\mathcal{B} = \mathcal{B}_{\mathsf{L}} = n + 1$.

2-round LSX-cipher can be represented as a sequence of the operations

$$y = K_3 \oplus \mathsf{S}\,(K_2 \oplus \mathsf{LS}\,(K_1 \oplus x))\,,$$

where $x, y \in \mathbf{F}_{2^8}^n$ are the plaintext and the ciphertext, $K_1, K_2, K_3 \in \mathbf{F}_{2^8}^n$ are round keys derived from masterkey K. The linear transformation on the last round was omitted without loss of generality.

A differential trail $\Omega = (\Delta x, \Delta_1, \Delta_2, \Delta y)$ in 2-round LSX is a collection of four differences, where $\Delta x = x \oplus x'$, Δ_1 is the difference after the first nonlinear transformation, $\Delta_2 = \mathsf{L}\,(\Delta_1)$, $\Delta y = y \oplus y'$, x and x' are plaintext blocks, y and y' are the corresponding ciphertext blocks.

Definition 6. *The expected probability of the 2-round trail* Ω *is defined as*

$$\mathrm{EDCP}\,(\Omega) = 2^{-\kappa} \sum_{K\in\mathbf{F}_2^\kappa} \Pr\left(\Delta_1{=}x_1{\oplus}x_1^{'} \text{ and } \Delta_2{=}x_2{\oplus}x_2^{'} \text{ and } \Delta y{=}y{\oplus}y' \text{ if } x'{=}\Delta x{\oplus}x\right),$$

where x *is a uniformly distributed random variable;* $x_1, x_1^{'}$ *are states after the first* S*-transformation;* $x_2, x_2^{'}$ *are states before the second* S*-transformation;* κ *is a size of the masterkey* K*.*

According to the assumption about round keys

$$\mathrm{EDCP}\,(\Delta x, \Delta_1, \Delta_2, \Delta y) = \left(\prod_{j=1}^{n} \mathrm{DP}(\Delta x[j], \Delta_1[j])\right)\left(\prod_{j=1}^{n} \mathrm{DP}(\Delta_2[j], \Delta y[j])\right).$$

Note, that if $\mathrm{EDCP}\,(\Delta x, \Delta_1, \Delta_2, \Delta y) \neq 0$, then $\mathrm{Supp}\,(\Delta x) = \mathrm{Supp}\,(\Delta_1)$, $\mathrm{Supp}\,(\Delta_2) = \mathrm{Supp}\,(\Delta y)$ and (Δ_1, Δ_2) is a codeword of the code $\mathcal{C}_{\mathcal{L}}$. Therefore $\mathrm{EDP}\,(\Delta x, \Delta y) =$

$$= \sum_{\substack{(\Delta_1,\Delta_2)\in\mathcal{C}_{\mathcal{L}},\\ \mathrm{Supp}(\Delta x)=\mathrm{Supp}(\Delta_1),\\ \mathrm{Supp}(\Delta_2)=\mathrm{Supp}(\Delta y)}} \prod_{j\in\mathrm{Supp}(\Delta x)} \mathrm{DP}\,(\Delta x[j], \Delta_1[j]) \prod_{j\in\mathrm{Supp}(\Delta y)} \mathrm{DP}\,(\Delta_2[j], \Delta y[j])\,.$$

The equality between the above formula of $\mathrm{EDP}\,(\Delta x, \Delta y)$ and the definition 2 was proved in [8].

We define the weight (in bytes) of the differential $(\Delta x, \Delta y)$ or the differential trail $(\Delta x, \Delta_1, \Delta_2, \Delta y)$ as $\mathrm{wt}(\Delta x) + \mathrm{wt}(\Delta y)$. Denote

$$\mathrm{MEDP}_w = \max_{\Delta x \neq 0, \Delta y, \mathrm{wt}(\Delta x)+\mathrm{wt}(\Delta y)=w} \mathrm{EDP}\,(\Delta x, \Delta y)\,,$$

$$\mathrm{MEDP}^+_w = \max_{\Delta x \neq 0, \Delta y, \mathrm{wt}(\Delta x)+\mathrm{wt}(\Delta y)\geq w} \mathrm{EDP}\,(\Delta x, \Delta y)\,,\ \mathcal{B} \leq w \leq 2 \cdot n.$$

Note that all mentioned definitions EDP, EDCP, MEDP are 2-round, unless otherwise stated.

Our main goal is to compute the nontrivial upper bound on $\mathrm{MEDP}^+_{\mathcal{B}+1}$, $\mathrm{MEDP}^+_{\mathcal{B}+2}$ and others.

3 Upper bound on non-minimum weight differentials

The strategy of our approach is as follows. Each differential trail $\Omega = (\Delta x, \Delta_1, \Delta_2, \Delta y)$ in 2-round differential $(\Delta x, \Delta y)$ uniquely corresponds to codeword (Δ_1, Δ_2) in $\mathcal{C}_{\mathcal{L}}$. All possible trails (codewords) in the differential have the form $\mathrm{Supp}(\Delta x) = \mathrm{Supp}(\Delta_1)$, $\mathrm{Supp}(\Delta_2) = \mathrm{Supp}(\Delta y)$. Derive constraints («maximum cost») for the entire set of such codewords. Divide the set into several subsets. Compute contribution to the constraints («cost») and the corresponding upper bound («value») for each possible subset. Select subsets so that the upper bound («total value») is maximum and the selection satisfies all constraints («total cost» does not exceed «maximum cost»). Thus, we obtain the upper bound on the differential.

3.1 Auxiliary lemmas

Lemma 1 (The rearrangement inequality [9]). *Let $l \in \mathbb{N}$, and suppose $c_1, c_2, \ldots, c_l$ and $d_1, d_2, \ldots, d_l$ are sequences of nonnegative values. Let $\widetilde{c}_1, \widetilde{c}_2, \ldots, \widetilde{c}_l$ and $\widetilde{d}_1, \widetilde{d}_2, \ldots, \widetilde{d}_l$ be the sequences obtained by sorting original sequences in nonincreasing order. Then*

$$\sum_{i=1}^{l} c_i d_i \leq \sum_{i=1}^{l} \widetilde{c}_i \widetilde{d}_i.$$

Lemma 2. *Let $l \in \mathbb{N}$, and suppose $c_1, c_2, \ldots, c_l$, and $\widetilde{c}_1, \widetilde{c}_2, \ldots, \widetilde{c}_l$, and $d_1, d_2, \ldots, d_l$ are sequences of nonnegative values. Each of them sorted in nonincreasing order. Suppose there exists l', $1 \leq l' \leq l$, such that*

1) $\widetilde{c}_i \geq c_i$, *for* $1 \leq i \leq l'$
2) $\widetilde{c}_i \leq c_i$, *for* $l'+1 \leq i \leq l$
3) $\sum_{i=1}^{l} c_i \leq \sum_{i=1}^{l} \widetilde{c}_i$
Then $\sum_{i=1}^{l} c_i d_i \leq \sum_{i=1}^{l} \widetilde{c}_i d_i$.

Proof. The proof of the lemma is given in particular in [6]. □

If statements 1-3 holds for some sequences $\widetilde{\boldsymbol{c}}$ and $\boldsymbol{c}$, then we will say that $\widetilde{\boldsymbol{c}}$ is greater than $\boldsymbol{c}$ under the conditions of Lemma 2.

Lemma 3. *Let* D *be a* $h \times v$ *matrix. Let*

$$D[i][j] \in \{p_1, p_2, ..., p_t, p_{\max}\},\ 1 \leq i \leq h,\ 1 \leq j \leq v,\ t \in \mathbb{N},$$
$$0 \leq p_1 < p_2 < \ldots < p_t < p_{\max} \leq 1,\ p_k, p_{\max} \in \mathbb{R},\ 1 \leq k \leq t.$$

Denote

$$\begin{gathered}\nu_k = \#\{(i,j) : D[i][j] = p_k,\ 1 \leq i \leq h,\ 1 \leq j \leq v\},\ 1 \leq k \leq t,\\ \nu_{\max}(D) = \#\{(i,j) : D[i][j] = p_{\max},\ 1 \leq i \leq h,\ 1 \leq j \leq v\}.\end{gathered} \tag{2}$$

Denote $\omega_l(D)$ *the number of rows containing exactly* l *elements* $p_{\max}$

$$\begin{gathered}\omega_l(D) = \#\{i : \#\{j : D[i][j] = p_{\max},\ 1 \leq j \leq v\} = l,\ 1 \leq i \leq h\},\\ \sum_{l=1}^{v} \omega_l(D) \cdot l = \nu_{\max}(D),\\ l_{\max} = \max_{\omega_l(D) \neq 0} (l)\,.\end{gathered} \tag{3}$$

Let $\widetilde{D}$ *be the reordered matrix* D *(see Fig. 1). Distributions from (2) and (3) are also holds for* $\widetilde{D}$.

The reordering procedure consists of three following steps:
1) sort each row of $\widetilde{D}$ *in nonicreasing order;*
2) sort each column of $\widetilde{D}$ *in nonicreasing order;*
3) reorder each unequal to $p_{\max}$ *element:*

$$\begin{gathered}\forall i, j, i', j' :\ \widetilde{D}[i][j] = p_{\max} \text{ or } \widetilde{D}[i'][j'] = p_{\max} \text{ or}\\ \left(\widetilde{D}[i][j] \geq \widetilde{D}[i'][j'],\ i' > i \text{ or } i' = i,\ j' > j\right),\\ 1 \leq i, i' \leq h,\ 1 \leq j, j' \leq v.\end{gathered}$$

Then

$$\sum_{i=1}^{h} \prod_{j=1}^{v} D[i][j] \leq \sum_{i=1}^{h} \prod_{j=1}^{v} \widetilde{D}[i][j].$$

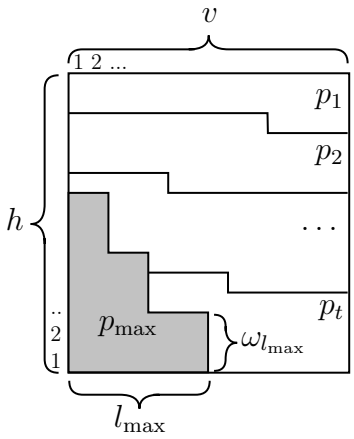


Figure 1: Example of matrix $\widetilde{D}$ after the reordering procedure.

Proof. The proof of the lemma is given in [18]. □

Lemma 4. *Let D and $\widetilde{D}$ be given as in Lemma 3. Suppose $c_1, c_2, \ldots, c_h$ is a sequence of nonnegative values. Let $\widetilde{c}_1, \widetilde{c}_2, \ldots, \widetilde{c}_h$ be obtained by sorting the above sequence in nonincreasing order. Then*

$$\sum_{i=1}^{h} c_i \prod_{j=1}^{v} D[i][j] \le \sum_{i=1}^{h} \widetilde{c}_i \prod_{j=1}^{v} \widetilde{D}[i][j].$$

Proof. Directly follows from Lemmas 1 and 3. □

3.2 Representation of trails in the differential

Consider an arbitrary differential $(\Delta x, \Delta y)$, $\mathrm{wt}(\Delta x) + \mathrm{wt}(\Delta y) = \mathcal{B} + 1$. The differential consists only of trails $(\Delta x, \Delta_1, \Delta_2, \Delta y)$ such that $\mathrm{Supp}(\Delta x) = \mathrm{Supp}(\Delta_1) = \{k_1, k_2, \ldots, k_t\}$, $\mathrm{Supp}(\Delta y) = \mathrm{Supp}(\Delta_2) = \{m_1, m_2, \ldots, m_r\}$, $t + r = \mathcal{B} + 1 = n + 2$.

It is easy to show that the number of differential trails does not exceed $T \le (2^8 - 1)^2$. Otherwise, there is a pair of codewords (Δ_1, Δ_2) and (Δ_1', Δ_2') such that

$$\mathrm{wt}\left((\Delta_1, \Delta_2) \oplus (\Delta_1', \Delta_2')\right) < \mathcal{B}.$$

Let's imagine a set of differential trails in the form of a table. Such a table, called Trails, has a size of $T \times (n + 2)$. Each row is non-zero bytes of the corresponding codeword

$$\mathrm{Trails}[i] = \Delta_1[k_1], \ldots, \Delta_1[k_t], \Delta_2[m_1], \ldots, \Delta_2[m_r],\ 1 \le i \le T,$$

$$\mathrm{EDP}\left(\Delta x, \Delta y\right) = \sum_{i=1}^{T} \prod_{j=1}^{t} \mathrm{DP}(\Delta x[k_j], \mathrm{Trails}[i][j]) \cdot \prod_{j=t+1}^{t+r} \mathrm{DP}(\mathrm{Trails}[i][j], \Delta y[m_{j-t}]). \tag{4}$$

For definiteness let's sort the table by the byte value in the first column (see Fig.2).

Let an arbitrary byte of Δx with an index k_j, $1 \le j \le t$ be fixed. Consider j-th column of Trails. Bytes with the same value $\mathtt{x}$ will have the same probability $\mathrm{DP}(\Delta x[k_j], \mathtt{x})$. Similarly for Δy. Let us denote the corresponding table by $\mathrm{DP}^*(\mathrm{Trails})$, where

$$\begin{gathered}\mathrm{DP}^*(\mathrm{Trails}[i][j]) = \mathrm{DP}(\Delta x[k_j], \mathrm{Trails}[i][j]),\ 1 \le i \le T,\ 1 \le j \le t,\\ \mathrm{DP}^*(\mathrm{Trails}[i][j]) = \mathrm{DP}(\mathrm{Trails}[i][j], \Delta y[m_{j-t}]),\ 1 \le i \le T,\ t < j \le t + r.\end{gathered} \tag{5}$$

We will divide table columns into 3 groups (subtables). The group C contains exactly 1 column. In the group $\mathrm{Trails}_{\mathbb{I}}$ there are u columns. The third group has v columns, $1 + u + v = n + 2$.

$$\begin{gathered}\mathrm{Trails} = \mathrm{C}||\mathrm{Trails}_{\mathbb{I}}||\mathrm{Trails}_{\mathbb{III}},\\ \mathrm{DP}^*(\mathrm{Trails}) = \mathrm{DP}^*(\mathrm{Trails}_{\mathbb{I}})\,||\mathrm{DP}^*(\mathrm{Trails}_{\mathbb{I}})\,||\mathrm{DP}^*(\mathrm{Trails}_{\mathbb{III}}),\end{gathered} \tag{6}$$

where $||$ is concatenation. We also denote

$$\mathrm{Block}_j = \{\mathrm{Trails}_{\mathbb{I}}[i]||\mathrm{Trails}_{\mathbb{III}}[i] : \mathrm{C}[i] = j,\ 1 \le i \le T\},\ j \in \mathbf{F}_{2^8}^*. \tag{7}$$

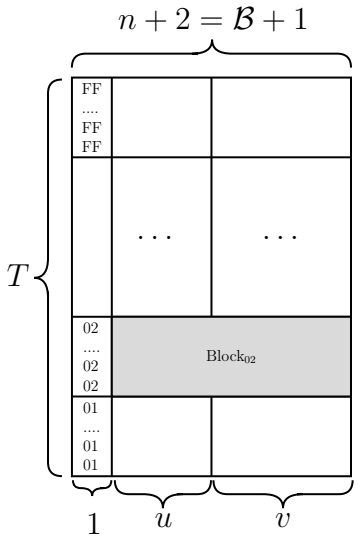


Figure 2: Representation of Trails

3.3 DDT simplification

Let all elements in each row (column) of the DDT be sorted in nonincreasing order. The row and the column with zero indexes are ignored. Let

us denote the such table $\mathrm{DDT}_{\mathrm{row}}$ ($\mathrm{DDT}_{\mathrm{col}}$ correspondingly)

$$\mathrm{DDT}_{\mathrm{row}}[x][1] \geq \mathrm{DDT}_{\mathrm{row}}[x][2] \geq \ldots \geq \mathrm{DDT}_{\mathrm{row}}[x][2^8-1],\ x \in \mathbf{F}_{2^8}^*,$$
$$\mathrm{DDT}_{\mathrm{col}}[1][y] \geq \mathrm{DDT}_{\mathrm{col}}[2][y] \geq \ldots \geq \mathrm{DDT}_{\mathrm{col}}[2^8-1][y],\ y \in \mathbf{F}_{2^8}^*.$$

We define sequences $\boldsymbol{m}_x$, $\boldsymbol{m}_y$ and $\boldsymbol{m}$ as

$$\boldsymbol{m}_x[i] = \max_{a \in \mathbf{F}_{2^8}^*} \mathrm{DDT}_{\mathrm{row}}[a][i], \quad \boldsymbol{m}_y[i] = \max_{a \in \mathbf{F}_{2^8}^*} \mathrm{DDT}_{\mathrm{col}}[i][a],\ i \in \mathbf{F}_{2^8}^*, \tag{8}$$
$$\boldsymbol{m}[i] = \max(\boldsymbol{m}_x[i], \boldsymbol{m}_y[i]),\ 1 \leq i \leq 2^8 - 1.$$

The sequence $\boldsymbol{m}$ is «greater» than any sorted nontrivial row/column of the DDT. Let $\boldsymbol{r}$ be any nontrivial sorted row/column of the DDT. Then, $\boldsymbol{m}[i] \geq \boldsymbol{r}[i]$, $1 \leq i \leq 2^8-1$. Denote $\nu_{\max}(\boldsymbol{m}) = \#\{i : \boldsymbol{m}[i] = p_{\max},\ {\scriptstyle 1 \leq i \leq 2^8-1}\}$. Note, that $\sum_{i=1}^{2^8-1} \boldsymbol{m}[i] \geq 1$.

We also define the sequences $\boldsymbol{\rho}$, $\boldsymbol{\rho}_x$, $\boldsymbol{\rho}_y$ as follows. Let $\boldsymbol{\rho}_x$ ($\boldsymbol{\rho}_y$) be one of the nontrivial sorted row (column) of the DDT. The sequence $\boldsymbol{\rho}_x$ ($\boldsymbol{\rho}_y$) must be greater than any other sorted row (column) of the DDT under the conditions of Lemma 2, $\sum_{i=1}^{2^8-1} \boldsymbol{\rho}_x[i] = \sum_{i=1}^{2^8-1} \boldsymbol{\rho}_y[i] = 1$. If $\boldsymbol{\rho}_x$ is greater than $\boldsymbol{\rho}_y$ under the conditions of Lemma 2, then $\boldsymbol{\rho} = \boldsymbol{\rho}_x$ otherwise $\boldsymbol{\rho} = \boldsymbol{\rho}_y$.

3.4 Constraints

We formulate a Lemma giving us some restrictions on the set of codewords.

Lemma 5. *Let table* $\mathrm{Trails}_{\mathbb{III}}$ *and sequence* $\boldsymbol{m}$ *be given as above. The table* $\mathrm{DP}^*(\mathrm{Trails}_{\mathbb{III}})$ *is defined by analogy with (5). Let us denote* $\omega_l(\mathrm{DP}^*(\mathrm{Trails}_{\mathbb{III}}))$ *the number of rows containing exactly* l *elements* $p_{\max}$*:*

$$\omega_l(\mathrm{DP}^*(\mathrm{Trails}_{\mathbb{III}})) = \#\{i : \#\{j : \mathrm{DP}^*(\mathrm{Trails}_{\mathbb{III}}[i][j]) = p_{\max},\ {\scriptstyle 1 \leq j \leq v}\} = l,\ {\scriptstyle 1 \leq i \leq T}\}. \tag{9}$$

Then

$$\omega_2 \leq \binom{v}{2} \cdot (\nu_{\max}(\boldsymbol{m}))^2, \tag{10}$$

and finally

$$\sum_{l=2}^{v} \omega_l \cdot \binom{l}{2} \leq \binom{v}{2} \cdot (\nu_{\max}(\boldsymbol{m}))^2. \tag{11}$$

Proof. Let's consider two arbitrary columns of $\mathrm{Trails}_{\mathbb{III}}$. These columns do not contain any identical byte pairs. The total number of different byte pairs does

not exceed $T \leq \left(2^8-1\right)^2$. In each column not more than $\nu_{\max}(\boldsymbol{m})$ values are mapped in $p_{\max}$. Hence, not more than $\left(\nu_{\max}(\boldsymbol{m})\right)^2$ byte pairs are mapped in $(p_{\max}, p_{\max})$. The number of ways to select 2 columns is $\binom{v}{2}$.

Thus we have (10).

Suppose there is a row containing 3 elements $p_{\max}$. Then $\binom{3}{2} = 3$ pairs of columns are generated, each of which contains a pair $(p_{\max}, p_{\max})$. Similarly for rows with l elements $p_{\max}$. Each of them «takes» $\binom{l}{2}$ pairs. Thereby we obtain (11). □

3.5 Bounds on $\mathrm{DP}^*(\mathrm{Block})$

Suppose that we are given an arbitrary $\mathrm{Block} \in \{\mathrm{Block}_j,\ j \in \mathbf{F}_{2^8}^*\}$. The block dimensions are $h \cdot (n+1)$, $h \leq 2^8-1$. We will give an upper bound on Block's contribution to the differential $\sum_{i=1}^{h}\prod_{j=1}^{n+1}\mathrm{DP}^*\left(\mathrm{Block}[i][j]\right)$. We will use Lemmas 2, 3, 4.

Consider $v=0$ and $u=n+1$. Then we have

$$\sum_{i=1}^{h}\prod_{j=1}^{u}\mathrm{DP}^*\left(\mathrm{Block}[i][j]\right) \leq \max\left(\max_{x\in\mathbf{F}_{2^8}^*}\sum_{i=1}^{2^8-1}\left(\mathrm{DDT}[x][i]\right)^u, \max_{y\in\mathbf{F}_{2^8}^*}\sum_{i=1}^{2^8-1}\left(\mathrm{DDT}[i][y]\right)^u\right). \tag{12}$$

The inequality (12) is so-called «FSE 2003 bound» on MEDP [14]. Lemma 2 allows us to select a row (column) that maximizes expression (12). Then we can rewrite inequality (12)

$$\sum_{i=1}^{h}\prod_{j=1}^{u}\mathrm{DP}^*\left(\mathrm{Block}[i][j]\right) \leq \sum_{i=1}^{2^8-1}\left(\boldsymbol{\rho}[i]\right)^u. \tag{13}$$

Let $v>0$. We will divide Block into two parts:

$$\mathrm{Block} = \mathrm{Block}_{\mathbb{I}} || \mathrm{Block}_{\mathbb{III}},$$

$$\sum_{i=1}^{h}\prod_{j=1}^{n+1}\mathrm{DP}^*\left(\mathrm{Block}[i][j]\right) = \sum_{i=1}^{h}\prod_{j=1}^{u}\mathrm{DP}^*\left(\mathrm{Block}_{\mathbb{I}}[i][j]\right)\prod_{j=1}^{v}\mathrm{DP}^*\left(\mathrm{Block}_{\mathbb{III}}[i][j]\right), \tag{14}$$

where $\mathrm{Block}_{\mathbb{I}}$ contains u columns, and $\mathrm{Block}_{\mathbb{III}}$ contains v columns, $u+v=n+1$. We will evaluate the contribution of $\mathrm{Block}_{\mathbb{I}}$ by using the sequence

$$\left(\boldsymbol{\rho}[1]\right)^u, \left(\boldsymbol{\rho}[2]\right)^u, \ldots, \left(\boldsymbol{\rho}[2^8-1]\right)^u. \tag{15}$$

We will also get a bound on the contribution of $\mathrm{Block}_{\mathbb{III}}$ by using Lemma 3. Suppose that each column of $\mathrm{DP}^*\left(\mathrm{Block}_{\mathbb{III}}\right)$ contains elements from the

sequence $\boldsymbol{m}$. Assume also that we know

$$\omega_l\left(\mathrm{DP}^*\left(\mathrm{Block}_{\mathbb{III}}\right)\right) = \#\{i : \#\{j : \mathrm{DP}^*\left(\mathrm{Block}_{\mathbb{III}}[i][j]\right) = p_{\max}, 1\le j\le v\} = l,\ 1\le i\le h\},$$
$$0 \le l \le v,$$
$$\sum_{l=1}^{v} \omega_l \cdot l \le \nu_{\max}(\boldsymbol{m}) \cdot v. \tag{16}$$

In other words, ω_l is the number of rows containing exactly l elements $p_{\max}$. Let $\widetilde{\mathrm{Block}_{\mathbb{III}}}$ be a table obtained by the reordering procedure from Lemma 3. Then we get

$$\sum_{i=1}^{h}\prod_{j=1}^{v}\mathrm{DP}^*\left(\mathrm{Block}_{\mathbb{III}}[i][j]\right) \le \sum_{i=1}^{h}\prod_{j=1}^{v}\mathrm{DP}^*\left(\widetilde{\mathrm{Block}_{\mathbb{III}}}[i][j]\right)$$

Thanks to Lemma 4 , we finally obtain

$$\sum_{i=1}^{h}\prod_{j=1}^{n+1}\mathrm{DP}^*\left(\mathrm{Block}[i][j]\right) \le \sum_{i=1}^{h}\left(\boldsymbol{\rho}[i]\right)^u\prod_{j=1}^{v}\mathrm{DP}^*\left(\widetilde{\mathrm{Block}_{\mathbb{III}}}[i][j]\right). \tag{17}$$

Thus, if we know the distribution ω_l, $0 \le l \le v$, then we can calculate the upper bound on $\sum_{i=1}^{h}\prod_{j=1}^{n+1}\mathrm{DP}^*\left(\mathrm{Block}[i][j]\right)$.

3.6 Optimization problem

Let's will form all possible sets

$$s_i = \{(l, \omega_l), 0 \le l \le v\},\ 1 \le i \le N. \tag{18}$$

For each set $\sum_{l=1}^{v}\omega_l \cdot l = \nu_{\max}(\boldsymbol{m}) \cdot v$ is true. In fact, we construct all possible partitions of the number $\nu_{\max}(\boldsymbol{m}) \cdot v$. The maximum term in the partition does not exceed v.

For each set s_i, calculate the estimate π_i using (17) and «contribution» ζ_i for constraints (11): $\zeta_i = \sum_{l=2}^{v}\omega_l \cdot \binom{l}{2}$. We can choose such u and v, which would *minimize* the final estimation. For most practical cases we use $u = 1$ and $v = n$. We get a set of pairs

$$(\pi_1, \zeta_1), (\pi_2, \zeta_2), \ldots, (\pi_N, \zeta_{N'}). \tag{19}$$

Pairs with the same ζ_i value can be removed. The pair with the largest π_i must be left. Hence $N' \le \binom{v}{2} \cdot \left(\nu_{\max}(\boldsymbol{m})\right)^2$.

We can estimate the first column of $\mathrm{DP}^*\left(\mathrm{Trails}\right)$ using the sequence $\boldsymbol{\rho}_x$ (or $\boldsymbol{\rho}_y$). Due to the fact that $\mathrm{wt}(\Delta x) \ge 1$ and $\mathrm{wt}(\Delta y) \ge 1$, we can choose

$\boldsymbol{\rho}_x$ or $\boldsymbol{\rho}_y$. We will choose so as to *minimize* the final value. For certainty, we assume that $\boldsymbol{\rho}_x$ has been chosen.

Denote $I = i_1, i_2, \ldots, i_{2^8-1}$, $1 \le i_j \le N'$, $1 \le j \le 2^8 - 1$. Then

$$\mathrm{MEDP}_{\mathcal{B}+1} \le \overline{\mathrm{MEDP}_{\mathcal{B}+1}} = \max_I \sum_{j=1}^{2^8-1} \boldsymbol{\rho}_x[j] \cdot \pi_{i_j} \text{ and } \sum_{i \in I} \zeta_i \le \binom{v}{2} \cdot \left(\nu_{\max}(\boldsymbol{m})\right)^2. \tag{20}$$

The optimal I is chosen by us using dynamic programming (see non-optimized version of the pseudocode in Appendix A, Algorithm 4).

There is a trivial estimate on $\mathrm{MEDP}_{\mathcal{B}+2} \le \sum_{i=1}^{2^8-1} \boldsymbol{\rho}[i] \cdot \overline{\mathrm{MEDP}_{\mathcal{B}+1}} = \overline{\mathrm{MEDP}_{\mathcal{B}+1}}$. Similar can be done for $\mathrm{MEDP}_{\mathcal{B}+3}$ etc. Thus, we proved that $\mathrm{MEDP}^+_{\mathcal{B}+1} \le \overline{\mathrm{MEDP}_{\mathcal{B}+1}}$.

3.7 Another constraints

We can compute the estimate on $\mathrm{MEDP}^+_{\mathcal{B}+1}$ more precisely.

Consider the table $\mathrm{DP}^*(\mathrm{Trails}_{\mathbb{III}})$. The number of rows that contains many elements $p_{\max}$ is quite small.

Recall that $\mathrm{wt}(\mathrm{Trails}_{\mathbb{III}}[i] \oplus \mathrm{Trails}_{\mathbb{III}}[j]) \ge v - 1$, $i \ne j$. Otherwise, there is a codeword $c \in \mathcal{C}_{\mathsf{L}}$, $\mathrm{wt}(c) < \mathcal{B}$. Thus, any two rows of $\mathrm{Trails}_{\mathbb{III}}$ have exactly one equal byte, or these rows do not have any matches.

In each column of $\mathrm{Trails}_{\mathbb{III}}$, no more than $\nu_{\max}(\boldsymbol{m})$ bytes are mapped in $p_{\max}$. $\mathrm{Trails}_{\mathbb{III}}$ has v columns. Denote $W = \nu_{\max}(\boldsymbol{m}) \cdot v$.

Suppose that some row of $\mathrm{DP}^*(\mathrm{Trails}_{\mathbb{III}})$ contains w_1 elements $p_{\max}$.

Let's say w_1 bytes of W were involved. Let the other row contain w_2 elements $p_{\max}$. These two rows can intersect at most one byte. Therefore, at least $w_2 - 1$ bytes are selected from W. The third row can intersect with the first and the second rows. Hence we subtract $w_3 - 2$ from W. Continue until $W \ge 0$.

Let us have a series w_1, w_2, w_3, ..., w_T sorted in noninreasing order, where T is the number of rows in $\mathrm{Trails}_{\mathbb{III}}$. Then

$$\left(W - \sum_{i=1}^{l} (w_i - (i-1))\right) \ge 0 \tag{21}$$

must be true for all $l \le T$.

Let's form all series $\psi = w_1, w_2, \ldots, w_l$ for which the inequality (21) is true. Denote the set of such series by Ψ. We will use a relatively small value of l (about 5, 6).

We can modify the algorithm from Subsection 3.6 as follows. For each set s_i from (18), we form a series $\psi = w_1, w_2, \ldots, w_l$. We obtain a sequence similar to (19): $(\pi_1, \zeta_1, \psi_1), (\pi_2, \zeta_2, \psi_2), \ldots, (\pi_N, \zeta_N, \psi_N)$.

Hence, another constraint is added to the optimization problem (20):

$$\mathrm{sort}_l \left(\psi_{i_1} || \psi_{i_2} || \ldots || \psi_{i_{2^8-1}} \right) \in \Psi,\ 1 \le i_j \le N, 1 \le j \le 2^8 - 1,$$

where sort_l is l largest elements of the sequence. Note that we do not need to store the entire sequence $\psi_{i_1} || \psi_{i_2} || \ldots || \psi_{i_{2^8-1}}$ in memory. We only need the first l values. Using the limitations described in this subsection requires a lot of computing resources. Therefore, this modification is not used in the calculation of bound on $\mathrm{MEDP}^+_{\mathcal{B}+2}$.

3.8 Computing $\mathrm{MEDP}^+_{\mathcal{B}+2}$ and other

Let us have $(\Delta x, \Delta y)$ such that $\mathrm{wt}(\Delta x) + \mathrm{wt}(\Delta y) = \mathcal{B} + 2 = n + 3$. Then Lemma 5 can be reformulated by analogy as follows.

Lemma 6. *Let the conditions of Lemma 5 be hold, but weight of the differential be equal to $n + 3$. Then*

$$\sum_{l=3}^{v} \omega_l \cdot \binom{l}{3} \le \binom{v}{3} \cdot \left(\nu_{\max}(\boldsymbol{m})\right)^3. \tag{22}$$

The algorithm is similar to Subsection 3.6, but the optimization problem is solved in two steps. As in Subsection 3.6:

– form all possible sets
$s_i = \{(l, \omega_l), 0 \le l \le v\}$, $1 \le i \le N$, $\sum_{l=1}^{v} \omega_l \cdot l = \nu_{\max}(\boldsymbol{m}) \cdot v$;

– for each set s_i, calculate the estimate π_i by (17); $\zeta_i = \sum_{l=2}^{v} \omega_l \cdot \binom{l}{2}$; $\eta_i = \sum_{l=3}^{v} \omega_l \cdot \binom{l}{3}$.

We obtain the sequence $(\pi_1, \zeta_1, \eta_1), (\pi_2, \zeta_2, \eta_2), \ldots, (\pi_N, \zeta_N, \eta_N)$.

Let's solve first optimization problem for all values $\eta' \le \binom{v}{3} \cdot \left(\nu_{\max}(\boldsymbol{m})\right)^3$. Denote $I = i_1, i_2, \ldots, i_{2^8-1}$, $i_j \in \mathbb{N}$, $1 \le j \le 2^8 - 1$.

$$\pi' = \max_I \sum_{j=1}^{2^8-1} \boldsymbol{\rho}_x[j] \cdot \pi_{i_j}, \text{ under condition } \sum_{i \in I} \zeta_i \le \binom{v}{2} \cdot \left(\nu_{\max}(\boldsymbol{m})\right)^2 \text{ and } \sum_{i \in I} \eta_i = \eta'.$$

We can get all the values η' by solving the optimization problem once.

Thus, the sequence $(\pi'_1, \eta'_1), (\pi'_2, \eta'_2), \ldots, (\pi'_{N'}, \eta'_{N'})$ will be obtained, $N' \le \binom{v}{3} \cdot \left(\nu_{\max}(\boldsymbol{m})\right)^3$.

We will solve the second optimization problem

$$\mathrm{MEDP}^{+}_{\mathcal{B}+2} \le \overline{\mathrm{MEDP}_{\mathcal{B}+2}} = \max_{I} \sum_{j=1}^{2^8-1} \boldsymbol{\rho}_x[j]\cdot\pi'_{i_j} \text{ and } \sum_{i\in I}\eta'_i \le \binom{v}{3}\cdot(\nu_{\max}(\boldsymbol{m}))^3 .$$

The pseudocode in Appendix A contains a non-optimized version of the algorithm. Application of the described approach is computationally infeasible for $\mathrm{MEDP}^{+}_{\mathcal{B}+3}$ in most cases. Furthermore, the potential estimation shift is very small (see summary table 1).

4 New bounds on MEDP for 2-round Kuznyechik

Kuznyechik block cipher [1] consists of a sequence of 9 rounds and a post-whitening key addition. The block size is 128 bits ($n = 16$ bytes), the key has a size of 256 bits. The cipher Sbox has no explicit analytical form [19], such as in AES. The rows and columns of the DDT have different unbalanced distributions. The sequence $\boldsymbol{m}_y$ is «greater» than $\boldsymbol{m}_x$. L-transformation is defined as a LFSR over $\mathbf{F}_{2^8}$, the differential branch number $\mathcal{B} = n + 1$.

In [18] was proved that each 2-round best differential contains only one differential trail
$\mathrm{MEDP} = \mathrm{MEDP}_{\mathcal{B}} = \max\limits_{\Omega\neq 0} \mathrm{EDCP}(\Omega) = \left(\frac{8}{256}\right)^{13}\left(\frac{6}{256}\right)^{4} = 2^{-86.66\ldots}$.

Using the proposed algorithms we showed that

$$\mathrm{MEDP}^{+}_{\mathcal{B}+1} \le 2^{-87.54\ldots}, \quad \mathrm{MEDP}^{+}_{\mathcal{B}+2} \le 2^{-88.34\ldots}.$$

The calculation $\mathrm{MEDP}^{+}_{\mathcal{B}+1}$ and $\mathrm{MEDP}^{+}_{\mathcal{B}+2}$ used the fact that $\mathrm{wt}\,(\Delta x) \ge 2$. We can use $\boldsymbol{\rho}_x$ instead of $\boldsymbol{\rho}$ (the rows of DDT instead the columns) in at least two coordinates. Obtained bound on $\mathrm{MEDP}^{+}_{\mathcal{B}+3}$ will be not less than $2^{-88.42\ldots}$.

Table 1 shows all computed values. The numbers are rounded to the second decimal place. The second data column presents the bounds we obtained using «FSE 2003 bounds» [14]. The last column (*) shows the limitation on the capabilities of the presented algorithm. For information about the linear method, see Appendix B.

$(p_{\max})^{\mathcal{B}}$	$\overset{\text{FSE2003}}{\mathrm{MEDP}_{\mathcal{B}}\le}$	$\mathrm{MEDP}_{\mathcal{B}} =$	$\mathrm{MEDP}^{+}_{\mathcal{B}+1} \le$	$\mathrm{MEDP}^{+}_{\mathcal{B}+2} \le$	$(*)\mathrm{MEDP}^{+}_{\mathcal{B}+3} \le$
-85	-83.97	-86.66	-87.54	-88.34	-88.42
$(p_{\mathrm{lin,max}})^{\mathcal{B}}$	$\overset{\text{FSE2003}}{\mathrm{MELP}_{\mathcal{B}}\le}$	$\mathrm{MELP}_{\mathcal{B}} =$	$\mathrm{MELP}^{+}_{\mathcal{B}+1} \le$	$\mathrm{MELP}^{+}_{\mathcal{B}+2} \le$	$(*)\mathrm{MELP}^{+}_{\mathcal{B}+3} \le$
-74.54	-73.54	-76.73	-77.15	-79.63	-80.50

Table 1: Summary table of results for Kuznyechik ($\log_2$ scale).

5 Conclusion

We propose a dynamic programming algorithm for bounding non-minimum weight differentials (linear hulls) in 2-round LSX-ciphers. Thanks to the presented algorithm, we derive some new bounds on differentials and linear hulls for 2-round Kuznyechik (Table 1). Similar results were obtained for 2-round Khazad (Table 2), and as a result, the exact values of $\mathrm{MEDP} = 2^{-45} + 2^{-60}$ and $\mathrm{MELP} = 2^{-37.80...}$ are also proved.

The source codes of the presented algorithms can be found at:
`https://gitlab.com/v.kir/diff2rLSX`

For any LSX-cipher with independent round keys, the R-round MEDP (MELP) is the upper bound for $(R+1)$-round MEDP (MELP). The presented results are a step towards obtaining new nontrivial bounds on R-round MEDP (MELP), i.e. new proofs of Kuznyechik strength against differential and linear cryptanalysis.

References

[1] *GOST R 34.12-2018 – National standard of the Russian Federation – Information technology – Cryptographic data security – Block ciphers*, 2018.

[2] Biham, E., Shamir, A., "Differential cryptanalysis of DES-like cryptosystems", *Journal of Cryptology*, 1991, 3–72.

[3] Matsui M., "Linear cryptanalysis method for DES cipher", *Advances in Cryptology – EUROCRYPT'93*, **765**, Springer, Berlin, Heidelberg, 1994, 386–397.

[4] Biham E., "On Matsui's linear cryptanalysis", *LNCS*, Advances in Cryptology – EUROCRYPT'94, **950**, Springer, Berlin, Heidelberg, 341–355..

[5] Keliher L., Sui. J., "Exact Maximum Expected Differential and Linear Probability for 2-Round Advanced Encryption Standard (AES)", *IET Information Security 1(2)*, 2007, 53–57.

[6] Keliher L., "Linear Cryptanalysis of Substitution-Permutation Networks, PhD Thesis", 2003.

[7] Keliher L., "Refined Analysis of Bounds Related to Linear and Differential Cryptanalysis for the AES", *LNCS*, Advanced Encryption Standard – AES, **3373**, ed. Dobbertin H., Rijmen V., Sowa A., Springer, Berlin, Heidelberg, 2005, 42–57.

[8] Lai, X., Massey, J.L., Murphy, S., "Markov ciphers and differential cryptanalysis", *LNCS*, Advances in Cryptology – EUROCRYPT'91, **547**, Springer-Verlag, 1991, 17–38.

[9] Hardy G.H., Littlewood J.E., Polya G., "Inequalities", *Cambridge Mathematical Library (2. ed.)*, Cambridge: Cambridge University Press, 1952.

[10] Sano F., Ohkuma K., Shimizu H., Kawamura S., "On the security of nested SPN cipher against the differential and linear cryptanalysis", *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, **E86-A, No. 1** (2003), 37–46.

[11] Daemen J., Rijmen V., "The Design of Rijndael: AES – The Advanced Encryption Standard Heidelberg etc.: Springer", 2002.

[12] Barreto P., Rijmen V., "The Khazad legacy-level block cipher", First open NESSIE Workshop. Leuven., November 2000.

[13] Kang J.-S., Hong S., Lee S., Yi O., Park C., Lim J., "Practical and provable security against differential and linear cryptanalysis for substitution-permutation networks", *ETRI Journal*, **23, No. 4,** (December 2001).

[14] Park, S., Sung, S.H., Lee, S., Lim, J., "Improving the Upper Bound on the Maximum Differential and the Maximum Linear Hull Probability for SPN Structures and AES.", *LNCS*, Fast Software Encryption - FSE 2003, **2887**, Springer, Berlin, Heidelberg, 2003, 247–260.

[15] Canteaut, A., Roue, J., "On the behaviors of affine equivalent sboxes regarding differential and linear attacks", *Advances in Cryptology – EUROCRYPT 2015*, **9056**, Springer, Berlin, Heidelberg, 2015, 45–74.

[16] Canteaut, A., Roue, J., "Differential Attacks Against SPN: A Thorough Analysis", *Codes, Cryptology, and Information Security*, C2SI 2015, May 2015, Rabat, Morocco, **9084**, Springer International Publishing, Cham, 2015, 45–62.

[17] Malyshev F.M., Trifonov D.I., "Diffusion properties of XSLP-ciphers", *Mat. Vopr. Kriptogr.*, **7:3** (2016), 47–60.

[18] Kiryukhin V., "Exact maximum expected differential and linear probability for 2-round Kuznyechik", *Mat. Vopr. Kriptogr.*, **10:2** (2019), 107–116.

[19] Shishkin V., Marshalko G., "A Memo on Kuznyechik S-Box", ISO/IEC JTC 1/SC 27/WG 2 Officer's Contribution N1804, September 2018.

A Pseudocode of algorithms

Algorithm 4: Computing $\overline{\mathrm{MEDP}_{\mathcal{B}+1}}$

Require: $(\pi_1, \zeta_1), (\pi_2, \zeta_2), \ldots, (\pi_{N'}, \zeta_{N'})$, and $\boldsymbol{\rho}_x$, and $s = \binom{v}{2} \cdot (\nu_{\max}(\boldsymbol{m}))^2$
Ensure: $\overline{\mathrm{MEDP}_{\mathcal{B}+1}}$

$\widetilde{\boldsymbol{\rho}}_x :=$ nondecreasing_sort($\boldsymbol{\rho}_x$) $\{0, \ldots, 0, \frac{2}{256}, \ldots, p_{\max}\}$
$\widetilde{\boldsymbol{\rho}}_x :=$ nonzero_elements($\widetilde{\boldsymbol{\rho}}_x$) $\{\frac{2}{256}, \ldots, p_{\max}\}$
state[s] := $[0, \ldots, 0]$ {indexing from 0}
for $j := 1$ **to** len($\widetilde{\boldsymbol{\rho}}_x$) **do**
  new_state[s] := $[0, \ldots, 0]$ {indexing from 0}
  $\mathrm{pr}_x := \widetilde{\boldsymbol{\rho}}_x[j]$
  for $c := 0$ **to** s **do**
    for $i := 1$ **to** N' **do**
      $\mathrm{pr} := \mathrm{pr}_x \cdot \pi_i + \mathrm{state}[c]$
      $\mathrm{pairs} := \zeta_i + c$
      if pairs $\leq s$ **then**
        if new_state[pairs] $<$ pr **then**
          new_state[pairs] := pr
        end if
      end if
    end for
  end for
  state := new_state
end for
return max(state)

The pseudocode above (Algorithm 4) contains a non-optimized version of the algorithm. The complexity of the algorithm is

$$O\left(\mathrm{len}(\widetilde{\boldsymbol{\rho}}_x) \cdot N' \cdot \binom{v}{2} \cdot (\nu_{\max}(\boldsymbol{m}))^2\right),$$

where $\mathrm{len}(\widetilde{\boldsymbol{\rho}}_x)$ is a number of nonzero elements in $\boldsymbol{\rho}_x$.

If $v = 16$, $\nu_{\max}(\boldsymbol{m}) = 2$, $\mathrm{len}(\widetilde{\boldsymbol{\rho}}_x) \leq 2^7$ (Kuznyechik), then the approximate number of operations is 2^{25} (less than a minute on a common PC). The number of distinct pairs $N' = 7665$.

Algorithm 5: Computing $\overline{\mathrm{MEDP}_{\mathcal{B}+2}}$

Require: $(\pi_1, \zeta_1, \eta_1), (\pi_2, \zeta_2, \eta_2), \ldots, (\pi_N, \zeta_N, \eta_N)$, and $\boldsymbol{\rho}_x$, and $s_{\text{pairs}} = \binom{v}{2} \cdot (\nu_{\max}(\boldsymbol{m}))^2$, $s_{\text{triplets}} = \binom{v}{3} \cdot (\nu_{\max}(\boldsymbol{m}))^3$

Ensure: $\overline{\mathrm{MEDP}_{\mathcal{B}+2}}$

1: $\widetilde{\boldsymbol{\rho}}_x :=$ nondecreasing_sort$(\boldsymbol{\rho}_x)$ $\{0, \ldots, 0, \frac{2}{256}, \ldots, p_{\max}\}$
2: $\widetilde{\boldsymbol{\rho}}_x :=$ nonzero_elements$(\widetilde{\boldsymbol{\rho}}_x)$ $\{\frac{2}{256}, \ldots, p_{\max}\}$
3: state[$\mathrm{s}_{\text{pairs}}$][$\mathrm{s}_{\text{triplets}}$] := $[0, \ldots, 0]$ {indexing from 0,0}
4: **for** $j := 1$ **to** len$(\widetilde{\boldsymbol{\rho}}_x)$ **do**
5: new_state[$\mathrm{s}_{\text{pairs}}$][$\mathrm{s}_{\text{triplets}}$] := $[0, \ldots, 0]$ {indexing from 0,0}
6: $\mathrm{pr}_x := \widetilde{\boldsymbol{\rho}}_x[j]$
7: **for** $c_{\text{pairs}} := 0$ **to** s_{pairs} **do**
8: **for** $c_{\text{triplets}} := 0$ **to** s_{triplets} **do**
9: **for** $i := 1$ **to** N **do**
10: pr := $\mathrm{pr}_x \cdot \pi_i +$ state$[c_{\text{pairs}}][c_{\text{triplets}}]$
11: pairs := $\zeta_i + c_{\text{pairs}}$
12: triplets := $\eta_i + c_{\text{triplets}}$
13: **if** pairs $\le s_{\text{pairs}}$ **and** triplets $\le s_{\text{triplets}}$ **then**
14: **if** new_state[pairs][triplets] $<$ pr **then**
15: new_state[pairs][triplets] := pr
16: **end if**
17: **end if**
18: **end for**
19: **end for**
20: **end for**
21: state := new_state
22: **end for**
23: $(\pi'_1, \eta'_1), \ldots, (\pi'_{N'}, \eta'_{N'}) :=$ (state[$\mathrm{s}_{\text{pairs}}$][0], 0), $\ldots$, (state[$\mathrm{s}_{\text{pairs}}$][$\mathrm{s}_{\text{triplets}}$], $\mathrm{s}_{\text{triplets}}$)
24: **return call** Algorithm 4 $((\pi'_1, \eta'_1), (\pi'_2, \eta'_2), \ldots, (\pi'_{N'}, \eta'_{N'}), \boldsymbol{\rho}_x, s = s_{\text{triplets}})$

The complexity of Algorithm 5 is estimated as trivial as Algorithm 4. If $v = 16$, $\nu_{\max}(\boldsymbol{m}) = 2$, $\mathrm{len}(\widetilde{\boldsymbol{\rho}}_x) \le 2^7$, then $N = 7665$ and the approximate number of operations is 2^{41} (about an hour on common PC).

B Application to Linear Cryptanalysis

There is a certain duality between differential and linear cryptanalysis [4]. It allows us to apply the algorithms described above to calculate linear

characteristics.

We make the appropriate substitutions.

Differential probability (DP, EDP, EDCP, MEDP) is replaced by linear probability (LP, ELP, ELCP, MELP correspondingly). DDT is replaced by Linear Approximation Table (LAT). Input/output differences Δx and Δy are replaced by input/output masks μ_x and μ_y correspondingly.

$$\mathrm{LP}(\mu_x, \mu_y) = (2\Pr(\mu_x \bullet x = \mu_y \bullet f(x)) - 1)^2,\ \mu_x, \mu_y \in \mathbf{F}_2^l,\ f : \mathbf{F}_2^l \to \mathbf{F}_2^l,$$

where $\bullet$ is the inner product over $\mathbf{F}_2$, and $x \in \mathbf{F}_2^l$ is a uniformly distributed random variable.

Differential branch number is replaced by linear branch number. If a linear transformation generates an MDS code both values are equal to $n+1$.

The value $p_{\max} = \max\limits_{a \neq 0, b} \mathrm{DDT}[a][b]$ is replaced by
$p_{\mathrm{lin,max}} = \max\limits_{a \neq 0, b} \mathrm{LAT}[a][b] = \mathrm{LP}(a, b),\ a, b \in \mathbf{F}_2^8$.

By analogy with the differential trail a linear characteristic $\Omega = (\mu_x, \mu_1, \mu_2, \mu_y)$ for 2 rounds is introduced. $\mathrm{ELCP}(\Omega)$ is equal to

$$\mathrm{ELCP}(\Omega) = \left(\prod_{j=1}^{n} \mathrm{LP}(\mu_x[j], \mu_1[j])\right)\left(\prod_{j=1}^{n} \mathrm{LP}(\mu_2[j], \mu_y[j])\right),$$

where $\mu_2 = \mathbb{L}^{\mathrm{T}} \cdot \mu_1$, $\mathbb{L}$ is a binary matrix such that $y = \mathsf{L}(x) = \mathbb{L} \cdot x$ and $\mathbb{L}^{\mathrm{T}}$ is a transposed matrix.

The linear code $\mathcal{C}_{\mathsf{L}}$ is replaced by the code $\mathcal{C}_{\mathbb{L}^{\mathrm{T}}}$.

The linear hull (similar to differential) is the set of all linear characteristics having input mask μ_x and output mask μ_y.

The expected probability of the 2-round linear hull (μ_x, μ_y) is equal to:

$$\mathrm{ELP}\,(\mu_x, \mu_y) = \sum_{(\mu_1, \mu_2) \in \mathbf{F}_2^{2 \cdot 8 \cdot n}} \left(\prod_{j=1}^{n} \mathrm{LP}(\mu_x[j], \mu_1[j])\right)\left(\prod_{j=1}^{n} \mathrm{LP}(\mu_2[j], \mu_y[j])\right) \text{ and}$$

$$\mathrm{MELP} = \max_{\mu_x \neq 0, \mu_y} \mathrm{ELP}\,(\mu_x, \mu_y)\,. \tag{23}$$

In order to go to linear cryptanalysis, one needs to replace all formulas in Section 3 according to the above analogies.

For 2-round Kuznyechik the only best linear hull containing 37 linear characteristics Ω_1,Ω_2, ..., Ω_{37} is found [18].

$$\mathrm{MELP} = \mathrm{MELP}_{\mathcal{B}} = \sum_{i=1}^{37} = \mathrm{ELCP}(\Omega_i) = 2^{-76.73\ldots}.$$

We show that

$$\mathrm{MELP}^{+}_{\mathcal{B}+1} \le 2^{-77.15\ldots}, \quad \mathrm{MELP}^{+}_{\mathcal{B}+2} \le 2^{-79.63\ldots}.$$

A bound on $\mathrm{MELP}^{+}_{\mathcal{B}+3}$ will be not less than $2^{-80.50\ldots}$.

C Khazad

Khazad [12] is a 64-bit ($n = 8$ byte) block cipher using a 128-bit key. It is an 8-round SP network. The plaintext is initially XORed with the whitening key and then undergoes 8 identical rounds.

S-transformation and L-transformation are involutions, $\mathsf{S} = \mathsf{S}^{-1}$, $\mathsf{L} = \mathsf{L}^{-1}$.

The sequences $\boldsymbol{m}_x$ and $\boldsymbol{m}_y$ are equal (see definition 8).

Due to this involution structure, we can consider only half of the subsets of codewords. Let's assume that for some 2-round differential $(\Delta x, \Delta y)$ we know the value of $\mathrm{EDP}(\Delta x, \Delta y)$. Then we know the value of $\mathrm{EDP}(\Delta y, \Delta x) = \mathrm{EDP}(\Delta x, \Delta y)$.

We have shown that each best differential contains two differential trails Ω_1 and Ω_2.

$$\mathrm{EDCP}(\Omega_1) = p^{\mathcal{B}}_{\max} = \left(\frac{8}{256}\right)^9 = 2^{-45}, \quad \mathrm{EDCP}(\Omega_2) = 2^{-60}.$$

Eight best differentials $(\Delta x, \Delta y)$ and eight differentials $(\Delta y, \Delta x)$ were found. For each of them $\mathrm{MEDP}_{\mathcal{B}} = \mathrm{EDP}(\Delta x, \Delta y) = \mathrm{EDP}(\Delta y, \Delta x) = \mathrm{EDCP}(\Omega_1) + \mathrm{EDCP}(\Omega_2)$.

We proved that $\mathrm{MEDP}^{+}_{\mathcal{B}+1} \le 2^{-44.99\ldots}$ and with improvements described in Subsection 3.7 $\mathrm{MEDP}^{+}_{\mathcal{B}+1} \le 2^{-45.02\ldots}$. Using algorithm from Subsection 3.8, we get $\mathrm{MEDP}^{+}_{\mathcal{B}+2} \le 2^{-45.09\ldots}$. Thus

$$\mathrm{MEDP} = \mathrm{MEDP}_{\mathcal{B}} = 2^{-45} + 2^{-60}.$$

We also found 16 best linear hulls: eight in the form (μ_x, μ_y) and eight in the form (μ_y, μ_x). Each of them contains 108 linear characteristics Ω_1, Ω_2, Ω_3, ..., Ω_{108}.

$$\mathrm{ELCP}(\Omega_1) = 2^{-37.80\ldots} < p^{\mathcal{B}}_{\mathrm{lin,max}} = 2^{-36}, \quad \mathrm{ELCP}(\Omega_2) = 2^{-67.70\ldots}.$$

$$\mathrm{MELP}_{\mathcal{B}} = \sum_{i=1}^{108} = \mathrm{ELCP}(\Omega_i) = 2^{-37.80\ldots}. \tag{24}$$

$$\mathrm{MELP}^{+}_{\mathcal{B}+1} \le 2^{-37.83\ldots}, \quad \mathrm{MELP}^{+}_{\mathcal{B}+2} \le 2^{-37.92\ldots}.$$

Because of this, we get

$$\mathrm{MELP} = \mathrm{MELP}_{\mathcal{B}} = 2^{-37.80\ldots}.$$

The obtaining of $\mathrm{MEDP}^{+}_{\mathcal{B}+3}$ and $\mathrm{MELP}^{+}_{\mathcal{B}+3}$ is computationally infeasible task for us. Furthermore, the result of the algorithm will be not less than $2^{-45.11\ldots}$ and $2^{-37.94\ldots}$ respectively.

Khazad					
$(p_{\max})^{\mathcal{B}}$	$\overset{\text{FSE2003}}{\mathrm{MEDP}_{\mathcal{B}}\leq}$	$\mathrm{MEDP}_{\mathcal{B}} =$	$\mathrm{MEDP}^{+}_{\mathcal{B}+1} \leq$	$\mathrm{MEDP}^{+}_{\mathcal{B}+2} \leq$	$(*)\mathrm{MEDP}^{+}_{\mathcal{B}+3} \leq$
-45	-43.36	-44.99	-45.02	-45.09	-45.11
$(p_{\mathrm{lin,max}})^{\mathcal{B}}$	$\overset{\text{FSE2003}}{\mathrm{MELP}_{\mathcal{B}}\leq}$	$\mathrm{MELP}^{\mathcal{B}} =$	$\mathrm{MELP}^{+}_{\mathcal{B}+1} \leq$	$\mathrm{MELP}^{+}_{\mathcal{B}+2} \leq$	$(*)\mathrm{MELP}^{+}_{\mathcal{B}+3} \leq$
-36	-35.86	-37.80	-37.83	-37.92	-37.94

Table 2: Table of results ($\log_2$ scale).

The best differentials

We show only 8 of the 16 differentials $(\Delta x, \Delta y)$. The remaining differentials $(\Delta y, \Delta x)$ can be easy obtained by swapping Δx and Δy.

Δx	1208f0000000000f		$\log_2 \mathrm{EDCP}(\Omega_i)$
Ω_1	1248f0000000000f	0000b548fbeb4800	-45
Ω_2	c8070a0000000023	0000130753a60700	-60
Δy		0000bf0818910800	
Δx	081200f000000f00		$\log_2 \mathrm{EDCP}(\Omega_i)$
Ω_1	481200f000000f00	000048b5ebfb0048	-45
Ω_2	07c8000a00002300	00000713a6530007	-60
Δy		000008bf91180008	
Δx	f0001208000f0000		$\log_2 \mathrm{EDCP}(\Omega_i)$
Ω_1	f0001248000f0000	b54800004800fbeb	-45
Ω_2	0a00c80700230000	13070000070053a6	-60
Δy		bf08000008001891	
Δx	00f008120f000000		$\log_2 \mathrm{EDCP}(\Omega_i)$
Ω_1	00f048120f000000	48b500000048ebfb	-45
Ω_2	000a07c823000000	071300000007a653	-60
Δy		08bf000000089118	
Δx	0f00000000f00812		$\log_2 \mathrm{EDCP}(\Omega_i)$
Ω_1	0f00000000f04812	0048ebfb48b50000	-45
Ω_2	23000000000a07c8	0007a65307130000	-60
Δy		0008911808bf0000	
Δx	000f0000f0001208		$\log_2 \mathrm{EDCP}(\Omega_i)$
Ω_1	000f0000f0001248	4800fbebb5480000	-45
Ω_2	002300000a00c807	070053a613070000	-60
Δy		08001891bf080000	
Δx	00000f00081200f0		$\log_2 \mathrm{EDCP}(\Omega_i)$
Ω_1	00000f00481200f0	ebfb0048000048b5	-45
Ω_2	0000230007c8000a	a653000700000713	-60
Δy		91180008000008bf	
Δx	0000000f1208f000		$\log_2 \mathrm{EDCP}(\Omega_i)$
Ω_1	0000000f1248f000	fbeb48000000b548	-45
Ω_2	00000023c8070a00	53a6070000001307	-60
Δy		189108000000bf08	

Table 3: The best 2-round Khazad differentials

The best linear hulls

As in the previous subsection, we show only 8 of the 16 linear hulls.

μ_x	6f078e0000000500	μ_x	076f008e00000005
μ_y	00006f0eb400e153	μ_y	00000e6f00b453e1
μ_x	8e006f0705000000	μ_x	050000008e006f07
μ_y	6f0e0000e153b400	μ_y	e153b4006f0e0000
μ_x	008e076f00050000	μ_x	00050000008e076f
μ_y	0e6f000053e100b4	μ_y	53e100b40e6f0000
μ_x	000005006f078e00	μ_x	00000005076f008e
μ_y	b400e15300006f0e	μ_y	00b453e100000e6f

Table 4: The best 2-round Khazad linear hulls

Ω_i	μ_1	μ_2	$\log_2 \mathrm{ELCP}(\Omega_i)$	Ω_i	μ_1	μ_2	$\log_2 \mathrm{ELCP}(\Omega_i)$
1	8e4c6f0000002c00	00008ee31300e11e	-37.80	22	e9645e0000004000	0000e973a800b716	-75.71
2	a3a9c1000000e300	0000a3fccd0062d8	-67.71	23	b1476b0000007f00	0000b15d3000dae4	-75.71
3	039d5d0000007100	00000319b6005e40	-70.37	24	2de5ae000000cf00	00002d1fde0083c6	-75.71
4	f15a660000008b00	0000f19f540097eb	-70.71	25	1deceb0000008800	00001dceb500f602	-75.81
5	8803e10000001e00	000088d8ec0069a1	-70.92	26	05d2d30000004300	000005224900d6ff	-75.91
6	a4927b0000000100	0000a4cfa000df58	-71.47	27	daf0460000007600	0000dabb75009c92	-76.03
7	f9639d0000007d00	0000f9c371006455	-71.77	28	32e02c000000e600	000032c04f001ebb	-76.34
8	1ba365000000ba00	00001bf54a007ebd	-71.85	29	465283000000c600	000046f99b00c5b0	-76.40
9	0ba4a60000008700	00000b459300adfe	-72.05	30	af36dd0000008600	0000af8a330072a6	-76.54
10	849cfd0000007b00	000084ae120079df	-72.56	31	d66f5a0000001300	0000d6cd8b008cec	-76.88
11	2f0cc70000006e00	00002f0efa00e8b9	-72.71	32	6167bf0000005e00	000061ab4400deb7	-76.92
12	bb97f90000002800	0000bb1031004225	-72.90	33	bf311e000000bb00	0000bf3aea00a1e5	-76.96
13	d3bd890000005000	0000d3efc2005a13	-72.98	34	c5f5c40000005f00	0000c564e40001ef	-77.40
14	ecb68d0000000300	0000ec51e10061e9	-73.32	35	42f4640000005500	000042d340002670	-77.51
15	6728310000006c00	00006790bb005608	-74.23	36	a0349c0000009200	0000a0e57b003c98	-77.54
16	064f8e0000003200	0000063bff0088bf	-74.28	37	4726b70000001600	000047f10900f08f	-77.81
17	9aed4b0000008200	00009a791100d19d	-74.62	38	bae3cd000000f800	0000ba18a300771a	-77.85
18	b5e18c000000ec00	0000b577eb003924	-74.92	39	d020d40000002100	0000d0f674000453	-78.15
19	35db960000000400	000035f32200a33b	-75.32	40	c96ad80000003a00	0000c9121a001191	-78.15
20	f715e8000000b900	0000f7a4ab001f54	-75.51	41	a80d670000006400	0000a8b95e00cf26	-78.30
21	1007c30000003d00	000010b0d900d343	-75.66	42	9804220000002300	000098683500bae2	-78.49

Table 5: One of the best 2-round Khazad linear hull,
$\mu_x =$ 6f078e0000000500, $\mu_y =$ 00006f0eb400e153 (part 1)

Ω_i	μ_1	μ_2	$\log_2 \mathrm{ELCP}(\Omega_i)$	Ω_i	μ_1	μ_2	$\log_2 \mathrm{ELCP}(\Omega_i)$
43	e5fb420000002500	0000e5055600a768	-78.68	76	a70f260000007000	0000a7d616008118	-85.22
44	ff2c130000004f00	0000fff88e00ecea	-78.76	77	82d3730000004900	00008295ed00f160	-85.32
45	701448000000b300	000070130f0038cb	-78.90	78	2943490000005c00	0000293505006006	-85.40
46	665c05000000bc00	0000669829006337	-79.02	79	903dd9000000d500	000090341000495c	-85.60
47	f02e520000005b00	0000f097c600a2d4	-79.20	80	9f3f98000000c100	00009f5b58000762	-85.85
48	7e623d0000007700	00007e74d50043ca	-79.32	81	de56a1000000e500	0000de91ae007f52	-85.85
49	aae40e000000c500	0000aaa87a00a459	-79.54	82	b708e50000004d00	0000b766cf00525b	-86.19
50	b67cd10000009d00	0000b66e5d006764	-79.71	83	d96d1b0000000700	0000d9a2c300c2d2	-86.49
51	6f11ca0000009a00	00006fcc9e00a5b6	-79.85	84	9dd6f10000006000	00009d4a7c006c1d	-86.49
52	93a084000000a400	0000932da600171c	-80.03	85	4950c2000000d200	00004996d3008b8e	-86.49
53	71607c0000006300	0000711b9d000df4	-80.15	86	8f385b000000fc00	00008feb8100d421	-86.49
54	2ade140000002d00	00002a2cb3003e46	-80.25	87	be452a0000006b00	0000be32780094da	-86.71
55	6bb72d0000000900	00006be645004676	-80.34	88	b2da360000000e00	0000b244860084a4	-86.83
56	75c69b000000f000	000075314600ee34	-80.37	89	9ca2c5000000b000	00009c42ee005922	-86.83
57	b0335f000000af00	0000b055a200efdb	-80.83	90	8977d5000000ce00	000089d07e005c9e	-87.60
58	c481f00000008f00	0000c46c760034d0	-81.02	91	a67b12000000a000	0000a6de8400b427	-87.66
59	1f05820000002900	00001fdf91009d7d	-81.34	92	8dd1320000005d00	00008dfaa500bf5e	-87.85
60	fb8af4000000dc00	0000fbd255000f2a	-81.40	93	caf7850000004b00	0000ca0bac004fd1	-88.19
61	6df8a30000003b00	00006dddba00cec9	-81.85	94	a5e64f000000d100	0000a5c73200ea67	-88.49
62	6c8c97000000eb00	00006cd52800fbf6	-82.05	95	5c85d2000000ac00	00005c0443008e32	-89.02
63	7dff600000000600	00007d6d63001d8a	-82.19	96	fe58270000009f00	0000fef01c00d9d5	-89.66
64	814e2e0000003800	0000818c5b00af20	-82.37	97	4e6b780000003000	00004ea5be00360e	-89.91
65	217ab2000000aa00	00002169200093b8	-82.57	98	52f3a70000006800	000052639900f533	-90.19
66	04a6e70000009300	0000042adb00e3c0	-82.82	99	682a700000007800	000068fff3001836	-90.49
67	eaf9030000003100	0000ea6a1e00e956	-82.83	100	e48f76000000f500	0000e40dc4009257	-90.49
68	d8192f000000d700	0000d8aa5100f7ed	-82.90	101	317d710000009700	000031d9f90040fb	-91.22
69	74b2af0000002000	00007439d400db0b	-83.66	102	738915000000c200	0000730ab900668b	-91.66
70	c027170000001c00	0000c046ad00d710	-83.74	103	62fae20000002f00	000062b2f20080f7	-92.19
71	eb8d37000000e100	0000eb628c00dc69	-83.85	104	0c9f1c0000006500	00000c76fe00107e	-92.19
72	15d5100000007e00	00001592900005bc	-84.03	105	173c79000000df00	00001783b4006ec3	-92.49
73	ccb80b0000007900	0000cc305300c76e	-84.57	106	dcbfc80000004400	0000dc808a00142d	-93.02
74	28377d0000008c00	0000283d97005539	-84.68	107	0f02410000001400	00000f6f48004e3e	-94.49
75	55c81d0000008a00	00005550f40048b3	-85.02	108	3ad9d70000001000	00003a9c6a00ed05	-97.66

Table 6: One of the best 2-round Khazad linear hull,
$\mu_x =$ `6f078e0000000500`, $\mu_y =$ `00006f0eb400e153` (part 2)

Protection against adversarial attacks with randomisation of recognition algorithm

Svetlana Koreshkova[1,2] and Grigory Marshalko[3]

[1]Lomonosov Moscow State University, Russia
[2]JSC «Kryptonite», Russia
[3]Technical committee for standardisation TC 26, Russia
marshalko_gb@tc26.ru

Abstract

We study a randomised variant of one type of biometric recognition algorithms which is intended to mitigate adversarial attacks. We show that the problem of an estimation of security of the method can be formulated in the form of an estimation of statistical distance between the probability distributions induced by initial and randomized algorithm. A variant of practical password-based implementation is discussed. The results of experimental evaluation are given.

Keywords: Biometric recognition, statistical distance, local binary patterns, password based authentication

1 Introduction

The active implementation of biometric identification and authentication technologies has led to the development of a wide range of attacks on systems that use them. One of the most widespread types of attacks on biometric systems are so-called adversarial attacks [4, 5], which consist in a careful modification of the attacking biometric image that comes to the input of the classifier. The danger of this class of attacks is that the biometric image is transformed in such a way that it allows to get access to the system, and at the same time in some sense does not disclose the fact of the attack.

To date, a wide range of approaches to counteracting this type of attack have been proposed (see the draft NIST technical report [8]), which can be divided into proactive (preventing the attack) and reactive (revealing the fact of the attack).

The most interesting question is how to build proactive methods, or, in other words, how to build biometric identification systems, a priori resistant to adversarial attacks. The main problem in this case arises in connection with the probabilistic nature of identification systems, namely the presence

of identification errors. Most of the attacks use variants of gradient method to modify attacking image so that its modified variant falls into the area of second type errors of the classifier used in the system.

Mitigation methods for this attacks were actively developed for neural networks, where we can recall adversarial training, gradient masking, Gaussian augmentation of the training set. In a more general case, you can specify approaches that use differential privacy and homomorphic encryption. With the exception, perhaps, of homomorphic encryption, which has extremely implementation unfriendly, these approaches do not provide complete protection against adversarial attacks.

In this paper we propose a method to mitigate adversarial attacks [3] on LBP[1]-based recognition systems [1, 2], which is build with randomization of the statistical criterion used in decision-making. This approach allows, for example, to build password-biometric authentication systems, where the recognition algorithm is significantly dependent on the password. In its turn, this approach allows to get rid of drawback of existing methods of counteraction, which is associated with the need to simultaneously preserve the quality of recognition of a legitimate biometric image, which, in its turn, leads to the preservation of the possibility of building adversarial attacks.

The further work is organised in the following way: in section 2 necessary notations are given and LBP algorithm is described, in section 3 the modified algorithm is described, its properties and implementation aspects are discussed, in section 4 experimental results are given.

2 Notation. LBP algorithm

We will use the following notations:

- N - number of people in database (number of classes);
- M_k - number of images in k-th class of database, $1 \leq k \leq N$;
- Y - input image described by its brigthness matrix: $Y = (y_{ij})_{m\times n}, y_{ij} \in [0, 255], i = \overline{1, m}, j = \overline{1, n}$;
- X_{kl} - l-th image of k-th class in database, $1 \leq k \leq N$, $1 \leq l \leq M_k$, described by its brigthness matrix: $X_{kl} = (x_{ij})_{m\times n}, x_{ij} \in [0, 255], i = \overline{1, m}, j = \overline{1, n}$;
- t - size of the block (parameter of LBP algorithm);

[1]Local binary patterns

– $B_{ij}^{m'\times n'}\big|_Z$ - matrix block of image Z, $B_{ij}^{m'\times n'}\big|_Z = (b_{xy})_{m'\times n'}, b_{xy} = z_{i+x,j+y}, 0 \le x < m', 0 \le y < n'$;

– $P_{ij}^{m'\times n'}\big|_Z$ — set of elements of block $B_{ij}^{m'\times n'}\big|_Z$ without the central element,
$P_{ij}^{m'\times n'}\big|_Z = b_{11}, b_{12}, \ldots, b_{1n'}, b_{2n'}, \ldots, b_{m'n'}, b_{m',n'-1}, \ldots, b_{m'1}, b_{m'-1,1}, \ldots, b_{21}$;

– $s(x)$ - Heaviside step function;

– $H_{ij}^{m'\times n'}\big|_Z$ - histogram of block $B_{ij}^{m'\times n'}\big|_Z$;

– H_Z - histogram of image Z;

– T - statistic threshold;

– $A||B$ - concatenation of vectors;

– $\mathcal{S}_n$ - the set of all permutations of the length n;

– $\mathcal{U}$ - uniform distribution.

The Local Binary Pattern Method (LBP) [1, 2] was proposed in 1996 for texture classification and later found wide application for image analysis. The idea is not to consider the whole image as a multidimensional array, but to highlight some local features of the object. It is a theoretically simple and at the same time effective method, allows a simple implementation and high performance. It is also resistant to monotonous change of illumination and scale.

Consider a block $B_{ij}^{t\times t}\big|_Z, t = 2q+1, q = 1, 2, ...$ of brigtness matrix of image Z. Then operator $LBP : [0, 255]^{t\times t} \to [0, 2^{t^2-1} - 1]$ when applied to the block is defined by the following formula:

$$LBP(B_{ij}^{t\times t}\big|_Z) = \sum_{q=0}^{t^2-1} 2^q s(p_q - p_{(x_{ij}^c, y_{ij}^c)}),$$

where (x_{ij}^c, y_{ij}^c) — coordinates of the central pixel $B_{t\times t}^Z\big|_{ij}$, p_q — brigtness of a pixel with number q (in a certain ordering) from $P_{ij}^{t\times t}\big|_Z$.

When building an identification system based on the LBP operator, an LBP matrix is built for each X_{kl} image from the database and the Y input image.

At the same time for image Z the blocks $B_{ij}^{t\times t}\big|_Z$ are formed for any $i, j : 0 \le i \le m-t, 0 \le j \le n-t$. Then LBP matrix is defined as

$$L_Z = (LBP(B_{ij}^{t\times t}\big|_Z))_{\hat{m}\times\hat{n}}, 0 \le i \le m-t, 0 \le j \le n-t, \hat{m} = m-t+1, \hat{n} = n-t+1.$$

To improve the recognition quality, all LBP matrices are combined into large blocks of (n_w, n_h) size. In this way L_Z is divided into non-intersecting blocks $B_{i'j'}^{\breve{m}\times\breve{n}}\big|_{L_Z}, \breve{m} = \frac{\hat{m}}{n_w}, \breve{n} = \frac{\hat{n}}{n_h}$:

$$Z = \begin{bmatrix} B_{00}^{\breve{m}\times\breve{n}}\big|_{L_Z} & B_{\breve{m},0}^{\breve{m}\times\breve{n}}\big|_{L_Z} & \dots & B_{\hat{m}-\breve{m},0}^{\breve{m}\times\breve{n}}\big|_{L_Z} \\ B_{0,\breve{n}}^{\breve{m}\times\breve{n}}\big|_{L_Z} & B_{\breve{m},\breve{n}}^{\breve{m}\times\breve{n}}\big|_{L_Z} & \dots & B_{\hat{m}-\breve{m},\breve{n}}^{\breve{m}\times\breve{n}}\big|_{L_Z} \\ \dots & & & \\ B_{0,\hat{n}-\breve{n}}^{\breve{m}\times\breve{n}}\big|_{L_Z} & B_{\breve{n},\hat{n}-\breve{n}}^{\breve{m}\times\breve{n}}\big|_{L_Z} & \dots & B_{\hat{m}-\breve{m},\hat{n}-\breve{n}}^{\breve{m}\times\breve{n}}\big|_{L_Z} \end{bmatrix}.$$

For each obtained block $B_{i'j'}^{\breve{m}\times\breve{n}}\big|_{L_Z}$ histogram $H_{ij}\big|_Z$ is calculated. The histogram describes the distribution of values in LBP matrix block. To get a full histogram of the image, the calculated histograms for blocks are concatenated:

$$H_Z = H_{00}^{\breve{m}\times\breve{n}}\big|_Z || H_{\breve{m},0}^{\breve{m}\times\breve{n}}\big|_Z || ... || H_{\hat{m}-\breve{m},\hat{n}-\breve{n}}^{\breve{m}\times\breve{n}}\big|_Z.$$

Within the identification system for images from the database and the input image there are histograms for which the distance is calculated. For example, L_1 metric can be used as a metric to estimate the proximity of histograms:

$$d(H_1, H_2) = \sum_i |H_1(i) - H_2(i)|.$$

The metric is used to determine the closest class of images with the nearest neighbour method:

$$Near(H_Y) = \arg \min_{1 \le k \le N} d(H_Y, H_{X_{kl}}), 1 \le l \le M_k.$$

For biometric system quality estimation FRR and FAR are ususally considered. A false positive response (FRR) is taken as the case in which the identification algorithm accepts the input image Y as an element of the class $k' \neq k$, but Y lies in the class k. As a false negative response (FAR), we consider the case in which the identification algorithm accepts the input image Y as an element of the class k, but Y lies in the class $k' \neq k$. Then the result of the algorithm is described by the following cases:

- If $k' = k$ and $d(H_{kj}, H_Y) \le T$, then the algorithm correctly accepts the image (TP — true positive);
- If $k' = k$ and $d(H_{kj}, H_Y) > T$, then false positive error occurs (FP — false positive);
- If $k' \neq k$ and $d(H_{ij}, H_Y) \le T$, then false negative error occurs (FN — false negative);

– If $k' \neq k$ and $d(H_{ij}, H_Y) > T$, then the algorithm correctly rejects the image (TN — true negative).

Then the error values are calculated as

$$FRR = \frac{FP}{TP + FP + TN + FN},$$

$$FRR = \frac{FN}{TP + FP + TN + FN}.$$

So, the recognition algorithm is described by the following parameters:

– t - block size,

– (n_w, n_h) - large block size,

– T - threshold.

3 Modified LBP algorithm

3.1 Description

To build a facial identification system that can resist adversarial attacks, an algorithm for calculating image characteristics is an important element. The original LBP algorithm is not resistant to relatively low changes of pixel brightness. This vulnerability can be successfully used to build an adversarial attack as in [3]. Attackers take advantage of the fact that they know how LBP works. This is why we offer a modification of the algorithm that uses a pseudo-random permutation when constructing the operator.

For modification, it is proposed that for each case of identification a permutation $\sigma \xrightarrow{U} \mathcal{S}_n$, where $n = t^2 - 1$ is generated. According to this permutation, the elements from the set $P_{ij}^{t\times t}\big|_Z$ are selected. So the LBP operator function takes the following form:

$$LBP'(P_{ij}^{t\times t}\big|_Z) = \sum_{q=0}^{t^2-1} 2^q s(p_{\sigma(q)} - p_{(x_{ij}^c, y_{ij}^c)}).$$

3.2 On the properties of the modified algorithm

We consider the black box model, i.e. the situation when the attacker is unable to directly observe the process of recognition system functioning and can only receive responses to the input data. This situation is encountered, for example, in remote biometric identification. A classic approach for the

attacker when mounting an adversarial attack in this case is teaching a local model close to the one used in the identification system, and building an adversarial image with its help. From this point of view, we need to use the proposed method to ensure that the attacker cannot evaluate the true parameters of the recognition algorithm and, in particular, the true histogram.

Since the histograms of H_Z are obtained by the concatenation of non-intersecting block histograms, we will assume the independence of the corresponding histograms of H_{ij} and estimate the method parameters for the latter.

Consider probability distribution f, describing a corresponding histogram of a block, $P(x_i = f_i), i = \overline{0, 2^{t^2-1} - 1}$. Without loss of generality consider $f_i < f_j, i < j$. Then we can evaluate the security of the proposed approach in terms of the statistical distance between the initial f distribution and the perturbed $f^\pi, P(x_i = f_i^\pi), i = \overline{0, 2^{t^2-1} - 1}$. Note that for the permuted distribution, the monotonicity property is no longer fulfilled.

In general, we can describe the difference between distributions in terms of the statistical distance between them $D_{f,f^\pi} = max_i|f_i - f_i^\pi|$, in this case the attacker will not be able to build an adversarial image if $D_{f,f^\pi} > \epsilon(T)$, where $\epsilon(T)$ depends on threshold T and ensures that both distributions are sufficiently far from each over. For example, if $\epsilon(T) > 2T$ then the attacker will never be able to get adversarial image close to both distributions at the same time.

Considering that the original image is divided into $n_w n_h$ independent blocks, if $D_{f,f^\pi} > \frac{2T}{n_w n_h}$ for each block, after concatenation the statistical distance of resulting distributions will be larger than $2T$ (if we use the L_1 metric), which means that the attacker will not be able to build an adversarial image.

Definition 1. *A «strong» permutation for distribution f is a permutation with $D_{f,f^\pi} > \epsilon(T)$.*

By a «weak»permutation we call a permutation which is not «strong». It is interesting to estimate the number of «strong» permutations, for which we can get the given value of the statistical distance. In general, such an estimate is difficult to obtain, but experimental studies (see Section 4) show a large number of zero elements in the H_{ij} histograms (see Fig. 2).

Lemma 1. *The number of «strong» permutations is equal to*

$$r! - \sum_{k=0}^{min\{|F_0|,|F_1|\}} \binom{|F_0|}{k} \binom{|F_1|}{|F_1| - k} \binom{|F_1| + |F_\epsilon| - k}{|F_\epsilon|} |F_0|!|F_1|!|F_e|!. \quad (1)$$

Proof. Consider the set of values $f_0, ..., f_{2^{t^2-1}-1}$ as a union of non-intersecting sets $F_0 = \{f_i : f_i = 0, i = \overline{0, 2^{t^2-1}-1}\}, F_1 = \{f_i : 0 < f_i \leq \epsilon, i = \overline{0, 2^{t^2-1}-1}\}$ and $F_\epsilon = \{f_i : f_i > \epsilon, i = \overline{0, 2^{t^2-1}-1}\}$. Then, to calculate the number of «strong» permutations, we subtract the number of «weak» permutations from the total number of permutations. For «weak» permutations, elements of the F_0 set can be moved only to places, corresponding to the F_0 and F_1 sets, the number of possible permutations will depend on k — the number of permutations moving elements between the sets. Then the total number of possible options for choosing places for rearranging elements between the sets F_0 and F_1 will be equal to $\sum_{k=0}^{min\{|F_0|,|F_1|\}} \binom{|F_0|}{k}\binom{|F_1|}{|F_1|-k}$, and the number of possible options for arranging elements from F_0 in selected places is $|F_0|!$. Elements of the set F_ϵ can be rearranged in the places corresponding to the sets F_1 и F_ϵ. The number of options for choosing places for arranging elements of the set F_ϵ also depends on k, since some places of the set F_1 are already occupied by elements of the set F_0, that is, only $\binom{|F_1|+|F_\epsilon|-k}{|F_\epsilon|}$ possible places and $|F_\epsilon|!$ ways to arrange elements in these places. The remaining elements of the set F_1 are arranged in the remaining possible places, which gives as coefficient $|F_1|!$. □

Using the given reasoning it is possible to obtain security estimations for specific instances of biometric systems.

3.3 Implementation aspects

One of possible ways to implement the proposed method can be represented by a password-biometric authentication system based on sequential application of password-based key derivation functions (e.g., PBKDF2 [10]), a key derivation function (e.g., [11]) and a Fisher-Yates random permutation algorithm [9].

Additional use of the key derivation function in this case seems reasonable due to the need to obtain permutations of a certain type. The function $kdf(S, T, L, P \ldots)$ described in [11] allows to get permutations with the required properties by changing additional non-secret parameters: S - salt, P - additional information, for the fixed key S, which is the result of password transformation with the function PBKDF2. For example, we can sequentially change salt until we get a «strong» permutation.

It should be noted that by changing non-specific parameters different permutations can be produced for each block.

LBP algorithm for database	LBP algorithm for input image	Algorithm threshold	FAR, %	FRR, %
Basic	Basic	8	0	68.6
Basic	Basic	9	0	34.3
Basic	Basic	10	0	14.3
Basic	Basic	11	75	5.7
Basic	Modified	8	0	100
Basic	Modified	9	0	100
Basic	Modified	10	0	100
Basic	Modified	11	0	100
Modified	Modified	8	0	68.6
Modified	Modified	9	0	34.3
Modified	Modified	10	0	14.3
Modified	Modified	11	75	5.7
Modified	Basic	8	0	100
Modified	Basic	9	0	100
Modified	Basic	10	0	100
Modified	Basic	11	0	100

Table 1: Identification results table for standard and proposed methods for $t = 3$

4 Experimental results

The experimental evaluation was performed on a database developed by the University of Cambridge computer laboratory (AT&T Database) [6]. The database contains a set of 40 photographs taken between April 1992 and April 1994. For each person 10 photos with different facial expressions, lighting, head rotation are used.

The full database AT&T Database of a set of 40 images is used as a testing base. Each image from the database is submitted to the identification system, which builds a combined histogram for this image and compares it with all histograms from the database using the closest neighbour method and L_1 metric. The images are considered to belong to the nearest neighbour class, if the distance between them by the specified metric is less than the threshold specified by the system.

The study compares type FRR and FAR errors when using standard and modified LBP algorithms. Also the errors of both types have been evaluated for the case when for the considered database only one way of construction of histogram is used. The results for $t = 3$ are presented in Table 1, for $t = 5$ — in Table 2. Examples of LBP images with the standard and modified algorithm are shown in Fig. 1, as well as the metric distances of L_1 between them.

LBP algorithm for database	LBP algorithm for input image	Algorithm threshold	FAR, %	FRR, %
Basic	Basic	15	0	48.6
Basic	Basic	16	0	31.4
Basic	Basic	17	25	15.7
Basic	Basic	18	75	4.3
Basic	Modified	15	0	100
Basic	Modified	16	0	100
Basic	Modified	17	0	100
Basic	Modified	18	0	100
Modified	Modified	15	0	48.6
Modified	Modified	16	0	31.4
Modified	Modified	17	25	15.7
Modified	Modified	18	75	4.3
Modified	Basic	15	0	100
Modified	Basic	16	0	100
Modified	Basic	17	0	100
Modified	Basic	18	0	100

Table 2: Identification results table for standard and proposed methods for $t = 5$

original 20.1 18.7

Figure 1: Examples of basic and modified images for $t = 3$

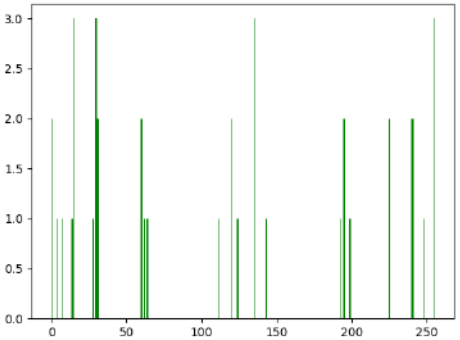


Figure 2: An example of a typical histogram of a large block

ϵ	$\lvert F_0\rvert$	$\lvert F_1\rvert$	$\lvert F_\epsilon\rvert$	Number of «weak» permutations	Number of «strong» permutations
2	179	46	29	$\approx 2^{1576}$	$\approx 2^{1684}$
5	179	61	14	$\approx 2^{1621}$	$\approx 2^{1684}$
7	179	65	9	$\approx 2^{1629}$	$\approx 2^{1684}$
9	179	67	7	$\approx 2^{1636}$	$\approx 2^{1684}$
11	179	69	5	$\approx 2^{1642}$	$\approx 2^{1684}$
13	179	71	4	$\approx 2^{1654}$	$\approx 2^{1684}$
17	179	72	2	$\approx 2^{1653}$	$\approx 2^{1684}$
20	179	73	1	$\approx 2^{1661}$	$\approx 2^{1684}$

Table 3: The table of cardinality of the sets of permutation elements depending on the parameter ϵ and the corresponding number of permutations.

Number of blocks	$\lvert F_0\rvert$	$\lvert F_1\rvert$	$\lvert F_\epsilon\rvert$	Number of «weak» permutations	Number of «strong» permutations
2	76	78	99	$\approx 2^{1544}$	$\approx 2^{1684}$
4	112	73	68	$\approx 2^{1542}$	$\approx 2^{1684}$
8	148	58	47	$\approx 2^{1546}$	$\approx 2^{1684}$
16	179	46	29	$\approx 2^{1575}$	$\approx 2^{1684}$
24	194	39	20	$\approx 2^{1585}$	$\approx 2^{1684}$
36	208	32	13	$\approx 2^{1603}$	$\approx 2^{1684}$

Table 4: Power table of sets of permutation elements depending on the number of partitioning blocks and the corresponding number of permutations, $\epsilon = 2$.

Given that the quality of the identification system depends on the chosen permutation, it is necessary to estimate the number of «strong» and «weak» histogram permutations. To do this, we considered the entire database, and for various ϵ calculated the amount of the «weak» and «strong» permutations. The table 3 shows the average sizes of the sets F_0, F_1, F_ϵ for different values of the parameter ϵ, $t = 3$. For these results, using the formula (1), we calculated the amount of the «strong» histogram permutations for $t = 3, \epsilon = \frac{2T+1}{n_w n_h} = \frac{2*10+1}{4*4} = 2$, which turned out to be approximately equal to 2^{1684}. The total number of permutations has the same power ($\approx 2^{1684}$).

Due to the fact that the number of «weak» permutations is of a lower order compared to the total number of permutations, the order of «strong» permutations is close to the total number for large values of t (see Table 3). In addition, the number of «weak» permutations depends on the number of blocks into which the image (see Table 4) is split. The results show that with increasing number of blocks, the number of «weak» permutations also

increases, but the recognition quality [1] also grows. Thus, the average estimates for the database in question allow us to conclude that the number of «strong» permutations is an order of magnitude greater than the number of «weak» permutations, which means that the system can be resistant to spoofing attacks. The optimization of the system parameters depends on the database used.

References

[1] Ahonen T., Hadid A., Pietikainen M., "Face description with local binary patterns: Application to face recognition", *IEEE Transactions on Pattern Analysis & Machine Intelligence*, **12** (2006), 2037–2041.

[2] Kanade T., "Picture processing system by computer complex and recognition of human faces", 1974.

[3] Kruglova S., Marshalko G., "Investigating the possibility of bypassing biometric facial recognition systems using the LBP algorithm", *Voprosy kiberbrzopasnosti*, **1** (2019), 45–52.

[4] Lavrentyeva G.M., Novoselov S.A., Kozlov A.V., Kudashev O.Yu., Shchemelinin V.L., Matveev Yu.N., De Marsico M., "Audio-replay attacks spoofing detection for speaker recognition systems", *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, **18**:3 (2018).

[5] Matveev Y.N., Volkova S.S.,, "Convolutional neural networks for anti-face spoofing", *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, **17**:4 (2017).

[6] Ojo J. A., Adeniran S. A., "Colour face image database for skin segmentation, face detection, recognition and tracking of black faces under real-life situations", *International Journal of Image Processing (IJIP)*, **4**:6 (2011), 600.

[7] Choi J. Y., Plataniotis K. N., Ro Y. M., "Using colour local binary pattern features for face recognition", 2010 IEEE International Conference on Image Processing, 2010, 4541–4544.

[8] Tabassi E., Burns K., Hadjimichael M., Molina-Markham A., Sexton J., "A Taxonomy and Terminology of Adversarial Machine Learning", 2019.

[9] Fisher R. A., Yates F., "Statistical tables: For biological, agricultural and medical research", 1938.

[10] "Recommendations for standardisation R 50.1.111-2016. Information technology. Cryptographic data protection. Password protection of key information.", 2016.

[11] "Recommendations for standardisation R 1323565.1.022-2018. Information technology. Cryptographic data protection. Key derivation functions.", 2018.

Asymmetric cryptography

On Methods of Shortening ElGamal-type Signatures

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

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

Abstract

Development of signature schemes providing short signatures is a quite relevant non-trivial challenge for cryptographers. Since the late 1980's many short signature schemes have been proposed. The most perspective schemes are multivariate schemes and schemes based on Weil pairing. Unfortunately, the cryptographic tools used in these schemes are still not supported by most cryptographic software that complicates their effortless use in practice.

In the current paper we investigate the opportunity of shortening the standard ElGamal-type signatures. We propose three methods of shortening signatures (for any ElGamal-type schemes such as ECDSA, GOST and SM2) and analyze how applying these methods affects the security. Applying all three methods to the GOST signature scheme with elliptic curve subgroup order q, $2^{255} < q < 2^{256}$, can reduce the signature size from 512 to 320 bits. The modified scheme provides sufficient security and acceptable (for non-interactive protocols) signing and verifying time.

Keywords: short signature scheme, ElGamal-type signature scheme, GOST, provable security.

1 Introduction

A signature scheme is one of the most widely used cryptographic protocol in practice. It is a self-supporting protocol replacing a handwritten signature and is used as a primitive in a huge amount of multiple protocols (e.g. TLS Handshake [1] and IKEv2 [2]). Therefore, the operational characteristics of signature scheme such as sizes of keys and signature, time complexity of signing and verifying, are crucial for applications. Although all parameters are important, in the current paper we focus only on the size of signature values.

One of the applications requiring short signatures is the systems where a human is asked to manually key in the signature. For example, product registration systems often ask users to key in a signature provided on a CD label. Also, the size of signature influences the requirements on a channel capacity which may be essential for low-bandwidth communication environments (e.g. Internet of Things and QR codes).

1.1 Related works

Due to the relevance of the considered issues many short signature schemes have been proposed since the late 1980's. One of these schemes is known as a BLS signature scheme [3] based on Weil pairing and providing 160 bits signatures with sufficient for practice security.

Other type of short signature schemes is multivariate schemes based on Hidden Field Equations (HFE) such as Quartz [5], Gui [10], SFLASH [4], UOV [12], Rainbow [13]. Although several of these schemes have been broken due to newly developed attacks (see [6, 7]), a number of multivariate schemes such as UOV, Rainbow withstood cryptanalysis (for suitable parameter sets) for more than 20 years. Also in 2016 the work [11] proposed technique reducing the signature size of almost every multivariate signature scheme by 10 to 15 % without increasing the key sizes or slowing down the scheme significantly. The authors claim that by applying their technique to the Gui signature scheme they obtain signatures of size only 110 bit, «which are the shortest signatures of all existing digital signature schemes». However, the scheme has relatively large public key (about 100 KByte) and slow verification time for the smallest signatures.

In light of the above, the BLS signature scheme is treated as a most favourable solution. Unfortunately, currently Weil pairing is still a non-typical cryptographic tool requiring generation and consequent deep analysis of so called «pairing-friendly» curves. Hence, not any cryptographic software supports this type of curves (unlike typical well-investigated elliptic curves used in standard signature schemes such as ECDSA and GOST). Therefore, the task to provide shorter signatures using typical cryptographic primitives is relevant.

1.2 Our contribution

In the current paper we consider the standard ElGamal-type signature schemes [14] (particularly GOST [19, 20, 21, 22]) with two-component signatures $\overline{r}\|\overline{s}$, where $\overline{r}$ and $\overline{s}$ are $\lceil \log_2 q \rceil$-bit strings (here q is the prime order of the used elliptic curve subgroup), and propose three methods of shortening this type of signature:

– The first method is to replace an internal function f (a mapping from a random elliptic curve point to an integer $r \in \mathbb{Z}_q$) by the hash function with truncated output that implies the reduction of the $\overline{r}$ component size.

– The main idea behind the second method is to make the certain bits of the $\overline{r}$ signature component to be constant and then to cut out them. This method leads to increasing signing time.

– The last method is to directly truncate signature ($\overline{r}$ and/or $\overline{s}$ component). This methods leads to increasing verification time.

All these methods are independent and can be applied together in any combination.

The idea to use hash function for the ElGamal-type signature shortening is also briefly mentioned in [29]. Unlike our first method, the method presented in [29] modifies the original signature scheme significantly due to hashing the message and the $\overline{r}$-component together. Moreover, from what we understand, the authors propose to use the same hash function as for message processing in the original signature scheme. Therefore, it is not clear by what exact means signature shortening is made since the hash function output is usually as long as the original components. Even if truncation of the hash output is implicitly supposed to be done, authors present only asymptotic security results (in terms of polynomial adversaries and negligible success probabilities). However, such a result cannot be used for choosing hash output length securely for practical application and concrete security bounds should be presented. The paper [11] proposes similar technique as in the last method but for multivariate signature schemes specifically. Unlike our method, signature verification in [11] does not imply signature recovering.

We analyse how applying the methods affects the security by obtaining concrete SUF-CMA-security bounds for modified schemes in the random oracle model. The first method changes the internal structure of the base scheme (function f) and in fact provides a new instance of the ElGamal-type signature scheme. For presentation purposes we obtain security bounds for the modified GOST signature scheme (named GOST-H) only, although we believe that the proof can be easily generalized. Using standard techniques we show that the hardness of elliptic curve discrete logarithm problem (ECDLP) and standard security properties of the hash function implies SUF-CMA-security of the GOST-H signature scheme. The second and the third methods are considered in general: for them we reduce the security of the base unmodified scheme to the security of the corresponding modified signature scheme. Applying all three methods to the GOST signature scheme with EC subgroup order q, $2^{255} < q < 2^{256}$, can reduce the signature size from 512 to 320 bits. The modified scheme provides sufficient security and acceptable for non-interactive protocols signing (≈ 6 seconds) and verifying

(≈ 3 seconds) time.

1.3 Paper organization

The remainder of the paper is organized as follows. In Section 2 basic definitions and notations are introduced. Section 3 introduces ElGamal-type signature schemes and describes the main object of the research — three methods of shortening signatures of such type. In Section 4 we formally define basic security notions for signature schemes and accompanying primitives. Section 5 is devoted to the security analysis of the proposed methods. We draw our conclusions in Section 6. Detailed proofs of our theorems are relegated to the appendices due to space limitations.

2 Basic notations and definitions

By $\{0,1\}^s$ we denote the set of s-component bit strings and by $\{0,1\}^*$ we denote the set of all bit strings of finite length including the empty string. For bit strings a and b we denote by $a\|b$ their concatenation. Let $|a|$ be the bit length of the string a.

For a bit string u and a positive integer $l \leqslant |u|$ let $\mathsf{msb}_l(u)$ ($\mathsf{lsb}_l(u)$) be the string consisting of the l rightmost (leftmost) bits of u. For integer $r \geqslant 0$ let $\mathsf{str}(r)$ (or just $\overline{r}$) be the $(\lfloor \log_2(r) \rfloor + 1)$-bit representation of $r > 0$ with the least significant bit on the left and zero bit if $r = 0$. For a bit string u let $\mathsf{int}(u)$ be the integer r such that $\mathsf{str}(r) = u$.

If p is a prime number then the set $\mathbb{Z}_p$ is a finite field with characteristic p. We assume the canonic representation of the elements in $\mathbb{Z}_p$ as a natural number in the interval $[0 \ldots p-1]$. 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 group generated by P as $\langle P \rangle$ and neutral element in $\mathbb{G}$ as O.

For any set A and B let $Func(A,B)$ be the set of all mappings from A to B. 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 a probabilistic algorithm A we denote $A \xrightarrow{\$} x$ ($x \xleftarrow{\$} A$). If we need to emphasize that A is deterministic than we denote

it by $A \to x$ ($x \leftarrow A$). The event when A returned value val as a result is denoted by $A \to val$.

We define security properties using 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 formal definition of this measure will be given in each specific case.

3 Three methods of shortening

A signature scheme consists of three algorithms $\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf}$ such that: algorithm KGen generates secret signing key sk and public signature verification key pk; algorithm Sig takes as input a signing key sk and message m and generates a signature sgn for message m; deterministic algorithm Vf takes as input verification key pk, message m and candidate signature sgn and outputs 1 (accept) or 0 (reject). It is required that for every $(\mathsf{pk}, \mathsf{sk})$ outputted by KGen and every message m it holds that

$$\mathsf{Vf}(\mathsf{pk}, m, \mathsf{Sig}(\mathsf{sk}, m)) = 1.$$

The GenElGamal framework is introduced in [14].

We follow the notations of [19] and change the definition in [14] in the following way: we represent the signature as the concatenation of vectors $\overline{t}$ and $\overline{s}$ instead of pair of elements in $\mathbb{Z}_q$, we denote r as k, h as e, t as r, x as d, X as Q. Moreover, we denote the GenElGamal signature scheme as GenEG.

The signature generated by the GenEG scheme is represented as $\overline{r}\|\overline{s}$, where $r, s \in \mathbb{Z}_q$, thus, its size is at most $2\lceil \log q \rceil$. The following sections introduce three methods of shortening signature size, we call the schemes obtained by applying these methods $\mathsf{GenEG\text{-}H}$, $\mathsf{GenEG_S}$ and $\mathsf{GenEG^V}$ respectively.

3.1 GenEG-H scheme

The r component in all GenEG schemes is computed as the result of applying function f to point R. The idea behind the first method of shortening

signature is to modify function f by splitting it into two functions. At first we apply the compression function H_2 to the x-coordinate of point R and then we represent the result as an element in $\mathbb{Z}_q^*$. Note that the bit representation of $\overline{r}$ will be shorter due to compression.

The modified function f is defined in the following way.

$$f(R) = \phi(H_2(R.x)),$$

where H_2 maps $\mathbb{Z}_p$ to $\{0,1\}^b$, $b < \lceil \log q \rceil$, and ϕ maps $\{0,1\}^b$ to $\mathbb{Z}_q^*$. The function H_2 can be instantiated by the function $H(1\|\overline{x}) \mod 2^b$, where H is the hash function that maps $\{0,1\}^*$ to $\mathbb{Z}_{2^{256}}$. The function ϕ is defined as follows: it maps $x \in \{0,1\}^b \setminus \{0\}$ to $\mathsf{int}(x)$ and maps zero to 2^b. This definition of the ϕ function is always correct due to the fact that $b < \lceil \log q \rceil$.

In order to separate domains of the hash function used for different cases, we redefine the hash function used for hashing messages as $H_1(x) = H(0\|\overline{x}) \mod q$.

We illustrate our method applying it to the GOST scheme which is a special case of the GenEG scheme. The formal definition of the GOST scheme is relegated to Appendix A. Note that function f in the GOST scheme (as well as in the ECDSA scheme) is defined as

$$f(R) = R.x \mod q.$$

We define the GOST-H scheme as follows.

$\mathsf{KGen}(\,)$	$\mathsf{Sig}(d,m)$	$\mathsf{Vf}(Q,m,\overline{r}\|\overline{s})$
1: $d \xleftarrow{\mathcal{U}} \mathbb{Z}_q^*$	1: $e \leftarrow H_1(m)$	1: **if** $s = 0$: **return** 0
2: $Q \leftarrow dP$	2: **if** $e = 0 : e \leftarrow 1$	2: $e \leftarrow H_1(m)$
3: **return** (d, Q)	3: $k \xleftarrow{\mathcal{U}} \mathbb{Z}_q$	3: **if** $e = 0 : e \leftarrow 1$
	4: **if** $k = 0$: **return** $\perp$	4: $R \leftarrow e^{-1}sP - e^{-1}rQ$
	5: $R \leftarrow kP$	5: **if** $\phi(H_2(R.x)) \neq r$: **return** 0
	6: $r \leftarrow \phi(H_2(R.x))$	6: **return** 1
	7: $s \leftarrow ke + dr$	
	8: **if** $s = 0$: **return** $\perp$	
	9: **return** $\overline{r}\|\overline{s}$	

This method allows us to shorten the size of the signature from $(2\lceil \log q \rceil)$ bits to at most $(\lceil \log q \rceil + b + 1)$ bits. For instance, for $b = \lceil \log q \rceil / 2$ we can cut out one quarter of the size.

Note that we do not check the condition $r = 0$ in the GOST-H scheme, because the function ϕ is defined in a such way that it does not map any argument to zero.

3.2 $\mathsf{GenEG_S}$ scheme

The idea behind the second method is to «mine» during the signature generation procedure. We generate signature until it meets certain additional conditions: the first l bits of $\overline{r}$ should match the constant vector. Thus, we can exclude these bits from the signature and reduce the signature size by l bits.

The $\mathsf{GenEG_S.KGen}$ algorithm is similar to the $\mathsf{GenEG.KGen}$ algorithm. The $\mathsf{GenEG_S.Sig}$ and $\mathsf{GenEG_S.Vf}$ procedures are defined as follows.

$\mathsf{Sig}(d, m)$

1 : $cnt \leftarrow 0$
2 : **if** $cnt > thr$: **return** $\perp$
3 : $cnt \leftarrow cnt + 1$
4 : $\overline{r}\|\overline{s} \leftarrow \mathsf{GenEG.Sig}(d, m)$
5 : **if** $\overline{r}\|\overline{s} = \perp$: **goto** 2
6 : **if** $\mathsf{lsb}_l(\overline{r}) \neq const$: **goto** 2
7 : $\overline{r}^* = \mathsf{msb}_{|\overline{r}|-l}(\overline{r})$
8 : **return** $\overline{r}^*\|\overline{s}$

$\mathsf{Vf}(Q, m, \overline{r}^*\|\overline{s})$

1 : $\overline{r} \leftarrow \overline{r}^*\|const$
2 : $res \leftarrow \mathsf{GenEG.Vf}(Q, m, \overline{r}\|\overline{s})$
3 : **return** res

The scheme defined above has two new parameters: l – number of fixed bits in $\overline{r}$ and thr – number of attempts to generate valid signature. These parameters are not independent from each other and they are strictly related to the generation time and probability of outputting the valid signature by the $\mathsf{GenEG_S.Sig}$ procedure. Thus they should be chosen in accordance with the generating mechanism computing power. We will discuss the appropriate values for these parameters in Section 5. The constant vector is an additional scheme parameter, it can be set to l zero bits for simplicity.

Note that the number of loop iterations needed to generate the valid signature depends on the probability of finding $\overline{r}$ satisfying the condition $\mathsf{lsb}_l(\overline{r}) = const$. In case of applying the method to the $\mathsf{GenEG\text{-}H}$ scheme, we can estimate this probability as 2^{-l} since the distribution of hash function output is close to uniform. We claim that the situation will not change in case of applying the method to schemes with function $f(R)$ equal to $R.x \mod q$ since function lsb_l is proven to be good entropy extractor for x-coordinate of point R (see [18] for more details).

3.3 $\mathsf{GenEG^V}$ scheme

The idea behind the third method is to truncate the signature (either $\overline{r}$ or $\overline{s}$ component) by t bits and search them during verification procedure.

The $\mathsf{GenEG}^{\mathsf{V}}$.KGen algorithm is similar to the GenEG.KGen algorithm. The $\mathsf{GenEG}^{\mathsf{V}}$.Sig and $\mathsf{GenEG}^{\mathsf{V}}$.Vf procedures are defined as follows.

$\mathsf{Sig}(d, m)$

1 : $\overline{r}\|\overline{s} \leftarrow \mathsf{GenEG.Sig}(d, m)$
2 : **if** $\overline{r}\|\overline{s} = \perp$: **return** $\perp$
3 : $\overline{s}^* \leftarrow \mathsf{msb}_{|\overline{s}|-t}(\overline{s})$
4 : **return** $\overline{r}\|\overline{s}^*$

$\mathsf{Vf}(Q, m, \overline{r}\|\overline{s}^*)$

1 : $i \leftarrow 0$
2 : **if** $i \geq 2^t$: **return** 0
3 : $\overline{s} \leftarrow \overline{s}^*\|\mathsf{str}_t(i)$
4 : $i \leftarrow i + 1$
5 : $res \leftarrow \mathsf{GenEG.Vf}(Q, m, \overline{r}\|\overline{s})$
6 : **if** $res = 0$: **goto** 2
7 : **return** 1

The construction defined above assumes truncating the $\overline{s}$ component of the signature, however we can define this scheme similarly up to truncating the $\overline{r}$ component. The decision which part of the signature should be truncated could depend on the possible optimization of verification process.

This method allows us to reduce the signature size by t bits. The value of parameter t is strictly related to the signature verification time and should be chosen in accordance with the verifier's computing power.

Note that the proposed method is general and can be applied not only to the GenEG signature scheme but also to any scheme with signature represented as a concatenation of two bit vectors. In particular, it can be applied to the $\mathsf{GenEG}_{\mathsf{S}}$ scheme by replacing the GenEG.Sig and GenEG.Vf calls with the corresponding $\mathsf{GenEG}_{\mathsf{S}}$ procedure calls.

4 Security notions

In this section we formally define basic security models used for signature schemes and the assumptions on primitives.

Definition 1. *For a signature scheme* SS

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

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

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

1 : $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{SS.KGen}(\,)$
2 : $\mathcal{L} \leftarrow \emptyset$
3 : $(m, sgn) \xleftarrow{\$} \mathcal{A}^{Sign}(\mathsf{pk})$
4 : **if** $(m, sgn) \in \mathcal{L}$: **return** 0
5 : $res \leftarrow \mathsf{SS.Vf}(\mathsf{pk}, m, sgn)$
6 : **return** res

Oracle $Sign(m)$

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

We analyse the security of the GOST-H scheme assuming the function H_2 to be a random oracle. The random oracle model was introduced in [27] and is an idealized model that assumes the existence of a public random function H such that all parties can obtain $H(x)$ (for any desired input value x) only by interacting with an oracle computing H; parties cannot compute H (for any input) on their own. Using a random oracle is a common way to ease the cryptographic analysis by making it modular. However, one should always keep in mind that a random oracle cannot be instantiated by any real hash function and, therefore, one should use it very carefully, trying to interpret the obtained security results. We discuss the meaning of random oracle model for our proof in the next section.

Definition 2 (ECDLP problem)**.**

$$\mathsf{Adv}_{\mathbb{G}}^{\mathrm{ECDLP}}(\mathcal{A}) = \Pr\left[Q \xleftarrow{\mathcal{U}} \langle P\rangle\,;\ d \xleftarrow{\$} \mathcal{A}(Q,P) : dP = Q\right]$$

Similar to [16] for the family H_1 of hash fuctions we define signum-relative collision resistance property (see Definition 3) and signum-relative division resistance property (see Definition 4). Throughout the paper we consider implicitly keyed hash functions $H_1\colon \{0,1\}^* \mapsto \mathbb{Z}_q$ 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 3 (SCR property)**.** *For the family of hash functions* H_1

$$\mathsf{Adv}_{\mathsf{H}_1}^{\mathrm{SCR}}(\mathcal{A}) = \Pr\left[(m_1,m_2) \xleftarrow{\$} \mathcal{A} : H_1(m_1) = \pm H_1(m_2) \wedge m_1 \neq m_2\right]$$

Definition 4 (SDR property)**.** *For the family of hash functions* H_1

$$\mathsf{Adv}_{\mathsf{H}_1}^{\mathrm{SDR}}(\mathcal{A}) =$$
$$\Pr\left[\beta_1,\beta_2 \xleftarrow{\mathcal{U}} \{0,1\}^b; (m_1,\Gamma) \xleftarrow{\$} \mathcal{A}_1(\beta_1), m_2 \xleftarrow{\$} \mathcal{A}_2(\Gamma,\beta_2) : \frac{H_1(m_1)}{\phi(\beta_1)} = \pm\frac{H_1(m_2)}{\phi(\beta_2)}\right]$$

The SDR property is implied by the standard assumptions: zero resistance and signum-relative preimage resistance properties of H_1 (see Appendix B.4 for formal proof and definitions of these properties).

We estimate the advantages defined above based on the best known methods of solving the corresponding security tasks. For the ECDLP problem it is the Pollard's ρ-algorithm (see [28]), for the SCR notion it is the attack based on birthday paradox and for the SDR notion (implied by the preimage

resistance) it is the exhaustive search. So for the group $\mathbb{G}$ and for the family of hash functions H_1 we assume that for any adversary $\mathcal{A}$ with the time complexity at most T

$$\mathsf{Adv}^{\mathrm{ECDLP}}_{\mathbb{G}}(\mathcal{A}) \approx \frac{T^2}{q}; \quad \mathsf{Adv}^{\mathrm{SCR}}_{\mathsf{H}_1}(\mathcal{A}) \approx \frac{T^2}{q}; \quad \mathsf{Adv}^{\mathrm{SDR}}_{\mathsf{H}_1}(\mathcal{A}) \approx \frac{T}{q}.$$

5 Security bounds

In this section we provide the security bounds for the schemes defined in Section 3.

For the GOST-H scheme we provide the reduction from the ECDLP problem. We claim that the proof for the other GenEG-H schemes can be obtained using the same technique.

Theorem 1. *Let $\mathcal{A}$ be an adversary with time complexity at most T in the* SUF-CMA *model for the* GOST-H *scheme, making at most Q_S queries to the Sign oracle and at most Q_O queries to the H_2 oracle. Then there exists an adversary $\mathcal{D}$ that solves the* ECDLP *problem for the used elliptic curve group $\mathbb{G}$, an adversary $\mathcal{C}$ that breaks the signum-relative collision resistant property of H_1 and an adversary $\mathcal{M}$ that breaks the signum-relative division resistant property of H_1, such that:*

$$\mathsf{Adv}^{\text{SUF-CMA}}_{\text{GOST-H}}(\mathcal{A}) \leqslant \sqrt{(Q_O+2)\left(\mathsf{Adv}^{\mathrm{SDR}}_{\mathsf{H}_1}(\mathcal{M})\cdot(Q_O+2)+\mathsf{Adv}^{\mathrm{ECDLP}}_{\mathbb{G}}(\mathcal{D})\right)}+ \\ + \frac{Q_O+3}{2^b} + \mathsf{Adv}^{\mathrm{SCR}}_{\mathsf{H}_1}(\mathcal{C}) + \frac{(2Q_O+Q_S+1)Q_S}{q-1}.$$

Furthermore, the time complexity of $\mathcal{C}$ is at most $T+c((Q_S+3)T^V_{\text{GOST-H}}+Q_O)$, the time complexities of $\mathcal{D}$ and $\mathcal{M}$ are at most $2T+2c((Q_S+4)T^V_{\text{GOST-H}}+2Q_O+4)$, where $T^V_{\text{GOST-H}}$ is computational resources needed to verify one signature by the GOST-H.Vf *procedure, c is a constant that depends only on a model of computation and a method of encoding.*

Here Q_S and Q_0 should be interpreted as a maximum number of signatures known to the adversary and as a maximum number of the hash function calls made by the adversary with the computational resources T respectively. The Q_S value is set in accordance with application requirements and the Q_O value depends on computational model and resources T. Usually we set Q_O to $\left\lceil \frac{T}{T_H} \right\rceil$, where T_H is the resources needed to compute one hash value for one-block message in the chosen computational model. It is correct as soon

as we assume sequential computational model, however we note that any parallel model of computations is equivalent to the corresponding sequential computational model with more resources [26].

Proof sketch. The idea behind the proof is similar to the idea used in [15]. The proof consists of two steps. During the first one we show that the notions of existential unforgeability under chosen message attack and under key-only attack are nearly equivalent, assuming H_1 is signum-relative collision resistant. Next, using the forking lemma we show that the hardness of the ECDLP in the group $\mathbb{G}$ and the signum-relative division resistance of H_1 imply unforgeability under key-only attack. The full proof can be found in Appendix B.

Note that for several steps we provide more accurate reductions than article [15] does (there are several unclarified places which seem to be incorrect, for details see Appendix B). Note that providing accurate reductions is quite important since potential mistakes can lead to practical vulnerabilities (see e.g. [23, 24]).

The interpretation of the random oracle model in our case is as follows. If the signature scheme turns out to be insecure, then, due to the proof sketch, the ECDLP problem is solved or the used hash function does not sufficiently disrupt the link between the domain and the range.

Remark 1. *Note that the obtained reduction is not tight: there are no known cryptanalytic attacks breaking the signature scheme with the specified probability and computational resources. This is the common problem of reductions obtained using forking lemma. Moreover, several negative results are known. Paillier and Vergnaud [25] show that the forgeability of several discrete log based signatures cannot be equivalent to solving the discrete log problem in the standard model, assuming the so-called one-more discrete log assumption and algebraic reductions.*

The only term depening on b is $\dfrac{Q_O + 3}{2^b}$. Applying the assumed bounds for $\mathsf{Adv}^{\mathrm{SDR}}_{\mathsf{H}_1}$, $\mathsf{Adv}^{\mathrm{SCR}}_{\mathsf{H}_1}$ and $\mathsf{Adv}^{\mathrm{ECDLP}}_{\mathbb{G}}$ we have the biggest term in the bound of order $\dfrac{\sqrt{Q_O} \cdot T}{\sqrt{q}}$. Thus, assuming $T > \sqrt{Q_O}$ (that is reasonable due to $Q_O \leqslant T$) we obtain the following surprising result: reducing b up to $\dfrac{\lceil \log_2 q \rceil}{2}$ does not significantly change the final bound. Thus, this method allows to shorten the size of the signature from $2\lceil \log_2 q \rceil$ bits to $\dfrac{3}{2}\lceil \log_2 q \rceil$ bits without harming security.

Theorem 2. *Let $\mathcal{A}$ be an adversary with time complexity at most T in the* SUF-CMA *model for the* $\mathsf{GenEG_S}$ *scheme, making at most Q_S queries to the Sign oracle. Then there exists an adversary $\mathcal{B}$ for the* GenEG *scheme in the* SUF-CMA *model that makes at most $Q_S \cdot thr$ queries to the Sign oracle, such that:*

$$\mathsf{Adv}^{\text{SUF-CMA}}_{\mathsf{GenEG_S}}(\mathcal{A}) = \mathsf{Adv}^{\text{SUF-CMA}}_{\mathsf{GenEG}}(\mathcal{B}).$$

Furthermore, the time complexity of $\mathcal{B}$ is at most $T + cQ_S thr$, where c is a constant that depends only on a model of computation and a method of encoding.

The proof can be found in Appendix C.

Consider the Q_S parameter in detail. Unlike the previous theorem, here Q_S cannot be interpreted as a number of signatures known to the adversary, since the scheme can return the failure indicator very often (depending on the parameters l and thr). Therefore, if N is a required for application number of signatures, then Q_S should be set to $\dfrac{N}{pr}$, where pr is the probability to return valid signature for one signing call. We assume that $pr \approx 1 - (1 - 2^{-l})^{thr}$.

Note that thr parameter should be chosen in such a way that the error probability is small enough for practice. The optimal way is to choose $thr = 2^l$ (in this case the error probability is less than e^{-1} for all l).

Theorem 3. *Let $\mathcal{A}$ be an adversary with time complexity at most T in the* SUF-CMA *model for the* $\mathsf{GenEG^V}$ *scheme, making at most Q_S queries to the Sign oracle. Then there exists an adversary $\mathcal{B}$ for the* GenEG *scheme in the* SUF-CMA *model that makes at most Q_S queries to the Sign oracle, such that:*

$$\mathsf{Adv}^{\text{SUF-CMA}}_{\mathsf{GenEG^V}}(\mathcal{A}) = \mathsf{Adv}^{\text{SUF-CMA}}_{\mathsf{GenEG}}(\mathcal{B}).$$

Furthermore, the time complexity of $\mathcal{B}$ is at most $T + c \cdot 2^t \cdot T^V_{\mathsf{GenEG}}$, where T^V_{GenEG} is computational resources needed to verify one signature by the GenEG.Vf *procedure, c is a constant that depends only on a model of computation and a method of encoding.*

The proof can be found in Appendix D.

Note that parameter t affects the security bound via the time complexity of the adversary $\mathcal{B}$. If we consider this parameter to be very large then the computational resources of $\mathcal{B}$ become too large, the GenEG scheme breaks and, as a result, the security bound degenerates.

The Theorems 2 and 3 are presented not in the random oracle model. However, if the security bound for the GenEG scheme is provided in the random oracle model we can change the theorems accordingly. If the adversary

$\mathcal{A}$ makes at most Q_O queries to H_2 oracle, then in case of the Theorem 2 the adversary $\mathcal{B}$ makes the same number of queries to its own H_2 oracle and in case of the Theorem 3 the adversary $\mathcal{B}$ makes at most $Q_O + 2^t$ queries to its own H_2 oracle.

The $\mathsf{GOST\text{-}H}^{\mathsf{V}}_{\mathsf{S}}$ scheme. Let us introduce the $\mathsf{GOST\text{-}H}^{\mathsf{V}}_{\mathsf{S}}$ scheme – the result of applying all three methods to the GOST scheme which is the special case of the GenEG scheme. The $\mathsf{GOST\text{-}H}^{\mathsf{V}}_{\mathsf{S}}.\mathsf{KGen}$ algorithm is similar to the $\mathsf{GOST.KGen}$ algorithm. The $\mathsf{GOST\text{-}H}^{\mathsf{V}}_{\mathsf{S}}.\mathsf{Sig}$ and $\mathsf{GOST\text{-}H}^{\mathsf{V}}_{\mathsf{S}}.\mathsf{Vf}$ procedures are defined as follows.

$\mathsf{Sig}(d, m)$

1: $cnt \leftarrow 0$
2: **if** $cnt > thr$: **return** $\perp$
3: $cnt \leftarrow cnt + 1$
4: $\overline{r}\|\overline{s} \leftarrow \mathsf{GOST\text{-}H.Sig}(d, m)$
5: **if** $\overline{r}\|\overline{s} = \perp$: **goto** 2
6: **if** $\mathsf{lsb}_l(\overline{r}) \neq const$: **goto** 2
7: $\overline{r}^* \leftarrow \mathsf{msb}_{|\overline{r}|-l}(\overline{r})$
8: $\overline{s}^* \leftarrow \mathsf{msb}_{|\overline{s}|-t}(\overline{s})$
9: **return** $\overline{r}^*\|\overline{s}^*$

$\mathsf{Vf}(Q, m, \overline{r}^*\|\overline{s}^*)$

1: $\overline{r} \leftarrow \overline{r}^*\|const$
2: $i \leftarrow 0$
3: **if** $i \geq 2^t$: **return** 0
4: $i \leftarrow i + 1$
5: $\overline{s} \leftarrow \overline{s}^*\|\mathsf{str}_t(i)$
6: $res \leftarrow \mathsf{GOST\text{-}H.Vf}(Q, m, \overline{r}\|\overline{s})$
7: **if** $res = 0$: **goto** 2
8: **return** 1

Summarizing the results of Theorems 1, 2, 3 and the bounds presented in Section 4 we obtain the following security bound for the $\mathsf{GOST\text{-}H}^{\mathsf{V}}_{\mathsf{S}}$ scheme:

$$\mathsf{Adv}^{\text{SUF-CMA}}_{\mathsf{GOST\text{-}H}^{\mathsf{V}}_{\mathsf{S}}}(\mathcal{A}) \leqslant \sqrt{2(Q_O + 2^t + 2) \cdot \frac{(Q_O + 2^t + 2)T_1 + 2T_1^2}{q}} + \\ + \frac{Q_O + 2^t + 3}{2^b} + \frac{(2Q_O + 2^{t+1} + Q_S \cdot thr + 1)Q_S \cdot thr + T_1^2}{q - 1},$$

where

$$T_1 \leqslant T + 2T^V_{\mathsf{GOST\text{-}H}}(Q_S \cdot thr + 2^t + 2) + 2Q_O + 4,$$

$T^V_{\mathsf{GOST\text{-}H}}$ is computational resources needed to verify one signature by the $\mathsf{GOST\text{-}H.Vf}$ procedure.

The size of the short signature generated by the $\mathsf{GOST\text{-}H}^{\mathsf{V}}_{\mathsf{S}}$ scheme is equal to at most $(\lceil \log q \rceil + b + 1 - l - t)$, where parameters b, l and t characterize three methods of shortening respectively. We provide the bounds for the $\mathsf{GOST\text{-}H}^{\mathsf{V}}_{\mathsf{S}}$ scheme for particular values of N, q, thr, T, Q_O: we set N to 10^6 as it is reasonable number of signatures for our application, we use curve with prime subgroup order q such that $2^{255} < q < 2^{256}$, we set thr to 2^l by reasons

Fixed parameters	Variable parameter and corresponding security bound					
$l = 18,\ t = 18$	b	128	100	80	70	60
	$\mathsf{Adv}^{\text{SUF-CMA}}_{\mathsf{GOST\text{-}H^V_S}}(\mathcal{A})$	2^{-35}	2^{-35}	2^{-19}	2^{-9}	1
$b = 100,\ t = 18$	l	10	18	35	50	77
	$\mathsf{Adv}^{\text{SUF-CMA}}_{\mathsf{GOST\text{-}H^V_S}}(\mathcal{A})$	2^{-35}	2^{-35}	2^{-34}	2^{-19}	1
$b = 100,\ l = 18$	t	10	18	30	64	80
	$\mathsf{Adv}^{\text{SUF-CMA}}_{\mathsf{GOST\text{-}H^V_S}}(\mathcal{A})$	2^{-35}	2^{-35}	2^{-35}	2^{-27}	1

Table 1: Security bounds for the $\mathsf{GOST\text{-}H^V_S}$ scheme

discussed above and we set T to 2^{60} assuming such computational power of potential adversary for our application. Moreover, for simplicity we estimate Q_O as T. The GOST-H.Vf procedure assumes two hash and two multiple point calculation, we estimate $T^V_{\mathsf{GOST\text{-}H}}$ as 32 assuming the computational resources measured in hash calculations.

Table 1 presents the evolution of security bound with changing one of the scheme parameter as long as other two parameters are fixed. We choose l and t equal to 18 in the first step based on the appropriate signing (≈ 6 seconds) and verifying (≈ 3 seconds) time, we set b to 100 based on the security bound obtained in the first step.

By choosing the optimal values of methods parameters ($b = 100$, $l = 18$ and $t = 18$) we reduce the signature size from 512 to 320 bits providing the sufficient security for our application.

6 Conclusion

This paper introduces three methods of shortening ElGamal-type signatures. The proposed methods do not imply increasing the key sizes and can be applied together in any combination. Applying second and/or third method leads to increasing signing and/or verifying time. The implementation of these methods do not require any special cryptographic tools.

We apply these methods to ElGamal-type signature schemes and obtain security bounds in the random oracle model. The presented theorems allow us to estimate the security of the modified scheme by the security of the used cryptographic primitives (elliptic curve group and hash function family) in case of the first method and by the security of the original scheme in case of the second and the third methods. The paper presents the security bounds

for the $\mathsf{GOST\text{-}H}^{\mathsf{V}}_{\mathsf{S}}$ scheme with elliptic curve subgroup order q, $2^{255} < q < 2^{256}$ with different parameter values (see Table 1). Choosing the optimal parameter values for our application allows to reduce the signature size from 512 to 320 bits.

References

[1] Rescorla, E., *The Transport Layer Security (TLS) Protocol Version 1.3*, RFC 8446, DOI 10.17487/RFC8446, 2018, https://www.rfc-editor.org/info/rfc8446.

[2] Kaufman, C., Hoffman P., Nir Y., Eronen P., Kivinen T., *Internet Key Exchange Protocol Version 2 (IKEv2)*, RFC 7296, DOI 10.17487/RFC7296, 2014, https://www.rfc-editor.org/info/rfc7296.

[3] Boneh D., Lynn B., Shacham H., "Short Signatures from the Weil Pairing", *LNCS*, Advances in Cryptology — ASIACRYPT 2001, **2248**, ed. Boyd C., Springer, Berlin, Heidelberg, 2001.

[4] Patarin J., Courtois N., Goubin L., "FLASH, a Fast Multivariate Signature Algorithm", *LNCS*, Topics in Cryptology — CT-RSA 2001, **2020**, ed. Naccache D., Springer, Berlin, Heidelberg, 2001.

[5] Patarin J., Courtois N., Goubin L., "QUARTZ, 128-Bit Long Digital Signatures", *LNCS*, Topics in Cryptology — CT-RSA 2001, **2020**, ed. Naccache D., Springer, Berlin, Heidelberg, 2001.

[6] Dubois V., Fouque PA., Shamir A., Stern J., "Practical Cryptanalysis of SFLASH", *LNCS*, Advances in Cryptology - CRYPTO 2007, **4622**, ed. Menezes A., Springer, Berlin, Heidelberg, 2007.

[7] Courtois N.T., Daum M., Felke P., "On the Security of HFE, HFEv- and Quartz", *LNCS*, Public Key Cryptography — PKC 2003, **2567**, ed. Desmedt Y.G., Springer, Berlin, Heidelberg, 2003.

[8] Courtois N.T., Finiasz M., Sendrier N., "How to Achieve a McEliece-Based Digital Signature Scheme", *LNCS*, Advances in Cryptology — ASIACRYPT 2001, **2248**, ed. Boyd C., Springer, Berlin, Heidelberg, 2001.

[9] Koblitz N., "Hidden Monomial Cryptosystems", *Algebraic Aspects of Cryptography, Algorithms and Computation in Mathematics*, **3**, Springer, Berlin, Heidelberg, 1998, 80–102.

[10] Petzoldt A., Chen MS., Yang BY., Tao C., Ding J., "Design Principles for HFEv- Based Multivariate Signature Schemes", *LNCS*, Advances in Cryptology – ASIACRYPT 2015, **9452**, ed. Iwata T., Cheon J., Springer, Berlin, Heidelberg, 2015.

[11] Mohamed M.S.E., Petzoldt A., "The Shortest Signatures Ever", *LNCS*, Progress in Cryptology – INDOCRYPT 2016, **10095**, ed. Dunkelman O., Sanadhya S., Springer, Cham, 2016.

[12] Kipnis A., Patarin J., Goubin L., "Unbalanced Oil and Vinegar Signature Schemes", *LNCS*, Advances in Cryptology — EUROCRYPT'99, **1592**, ed. Stern J., Springer, Berlin, Heidelberg, 1999.

[13] Ding J., Schmidt D., "Rainbow, a New Multivariable Polynomial Signature Scheme", *LNCS*, Applied Cryptography and Network Security. ACNS 2005, **3531**, ed. Ioannidis J., Keromytis A., Yung M., Springer, Berlin, Heidelberg, 2005.

[14] Fersch, M., Kiltz, E., Poettering, B., "On the One-Per-Message Unforgeability of (EC)DSA and Its Variants", *LNCS*, Theory of Cryptography. TCC 2017, **10678**, ed. Kalai Y., Reyzin L., Springer, Cham, 2017.

[15] Fersch M., Kiltz E., Poettering B., "On the provable security of (EC) DSA signatures", Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, 2016, 1651–1662.

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

[17] 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.

[18] Chevalier C., Fouque PA., Pointcheval D., Zimmer S., "Optimal Randomness Extraction from a Diffie-Hellman Element", *LNCS*, Advances in Cryptology - EUROCRYPT 2009, **5479**, ed. Joux A., Springer, Berlin, Heidelberg, 2009.

[19] *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.

[20] *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.

[21] *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.

[22] 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.

[23] Inoue A., Iwata T., Minematsu K., Poettering B., "Cryptanalysis of OCB2: Attacks on Authenticity and Confidentiality", *LNCS*, Advances in Cryptology – CRYPTO 2019, **11692**, ed. Boldyreva A., Micciancio D., Springer, Berlin, Heidelberg, 2019.

[24] Koblitz N., Menezes A., *Critical Perspectives on Provable Security: Fifteen Years of "Another Look" Papers*, Cryptology ePrint Archive: Report 2019/1336, 2019.

[25] Paillier P., Vergnaud D., "Discrete-Log-Based Signatures May Not Be Equivalent to Discrete Log", *LNCS*, Advances in Cryptology - ASIACRYPT 2005, **3788**, ed. Roy B., Springer, Berlin, Heidelberg, 2005.

[26] Savage J.E., *Models of Computation: Exploring the Power of Computing*, Addison-Wesley Longman Publishing Co, Boston, 1998.

[27] Bellare M., Rogaway P., "Random oracles are practical: A paradigm for designing efficient protocols", *Proceedings of the 1st ACM conference on Computer and communications security*, 1993, 62–73.

[28] Pollard, J.M., "A monte carlo method for factorization", *BIT*, **15** (1975), 331–334.

[29] Zheng Y., "Digital signcryption or how to achieve cost(signature & encryption) $\ll$ cost(signature) + cost(encryption)", *LNCS*, Advances in Cryptology — CRYPTO'97, **1294**, ed. Kaliski B.S., Springer, Berlin, Heidelberg, 1997.

A GOST definition

The GOST signature scheme is a special case of the GenEG scheme. We define it relative to functions H, f and group $\mathbb{G}$.

KGen( )	Sig(d, m)	Vf$(Q, m, \overline{r}\|\overline{s})$
1 : $d \xleftarrow{\mathcal{U}} \mathbb{Z}_q^*$	1 : $e \leftarrow H(m) \mod q$	1 : **if** $(r = 0 \vee s = 0)$: **return** 0
2 : $Q \leftarrow dP$	2 : **if** $e = 0 : e \leftarrow 1$	2 : $e \leftarrow H(m)$
3 : **return** (d, Q)	3 : $k \xleftarrow{\mathcal{U}} \mathbb{Z}_q$	3 : **if** $e = 0 : e \leftarrow 1$
	4 : **if** $k = 0$: **return** $\perp$	4 : $R \leftarrow e^{-1}sP - e^{-1}rQ$
	5 : $R \leftarrow kP$	5 : **if** $f(R) \neq r$: **return** 0
	6 : $r \leftarrow f(R)$	6 : **return** 1
	7 : **if** $r = 0$: **return** $\perp$	
	8 : $s \leftarrow ke + dr$	
	9 : **if** $s = 0$: **return** $\perp$	
	10 : **return** $\overline{r}\|\overline{s}$	

The function f maps $\mathbb{G}^*$ to $\mathbb{Z}_q$ and for the GOST scheme is defined as follows:

$$f(R) = R.x \mod q.$$

There are some differences between the scheme defined above and the standardized scheme defined in [19]. First, our version of the scheme can output failure indicator $\perp$. In contrast, function Sig in standardized scheme always output a valid signature, going to line 3 in case of all «bad» events. The second difference is that k is chosen randomly from the set $\mathbb{Z}_q^*$ in the standardized scheme, but in our scheme it is chosen from the set $\mathbb{Z}_q$, and the procedure Sig outputs $\perp$ in case of $k = 0$. We claim that these two differences do not affect the security and correctness of the scheme.

B GOST-H security

B.1 Proof details

We provide the proof in the random oracle model, i.e. we replace function H_2 with the random oracle. Family of hash functions H_1 is required to be signum-relative collision resistant and signum-relative division resistant in the sense of Definitions 3, 4.

Let us introduce SUF-KO security model for the signature scheme since we use it during the proof.

Definition 5. *For a signature scheme* SS

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

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

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

1: $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{SS.KGen}(\,)$
2: $(m, sgn) \xleftarrow{\$} \mathcal{A}(\mathsf{pk})$
3: $res \leftarrow \mathsf{SS.Vf}(\mathsf{pk}, m, sgn)$
4: **return** res

The idea behind the proof is similar to the idea used in [15], however several steps in [15] are unclear and seem to be incorrect. We provide more accurate reduction than [15] does and point out the differences from [15] throughout the proof. We split the proof into two parts. In Section B.2 we show that if the adversary can forge the $\mathsf{GOST\text{-}H}$ scheme using some valid pairs message-signature (i.e. in the SUF-CMA model), then we can construct two adversaries: one of them breaks the signum-relative collision resistance property of H_1 and the other one makes forgery without any valid pairs message-signature (i.e. in the SUF-KO model). In Section B.3 we construct the adversary that breaks the signum-relative division resistance property of H_1 and the adversary that solves the ECDLP problem using the key-only adversary constructed at the first step. The forking lemma (see [15]) is our key tool on the second step. Both steps of the proof are organized as follows: at first, we construct the sequence of experiments for the adversary and estimate the difference between them (in some cases by constructing the adversaries for H_1 properties), after that we build another adversary who uses the first adversary as a black box and implements the last experiment for him. We highlight the changes in the experiment 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. We use lemma 2 from [17] to estimate the difference between two experiments $\mathbf{Exp}^i$ and $\mathbf{Exp}^j$ that are «identical-until-bad», i.e. one experiment is derived from the other by adding the abort condition. According to this lemma

$$\Pr\left[\mathbf{Exp}^i \Rightarrow 1\right] - \Pr\left[\mathbf{Exp}^j \Rightarrow 1\right] \leqslant \Pr[\text{abort condition is met}].$$

For presentation purposes we introduce the internal result of function f and denote it as r'. More particularly, we denote $\mathsf{H}_2(R.x)$ as r' and thus $r = \phi(r')$. We assume that r' is the element in $\{0,1\}^b$. Moreover, we assume

throughout the proof that the GenEG signature is represented not like the vector's concatenation but as the pair of corresponding elements in $\mathbb{Z}_q$.

B.2 SUF-KO to SUF-CMA reduction

Theorem 4. *Let $\mathcal{A}$ be an adversary with time complexity at most T in the* SUF-CMA *model for the* GOST-H *scheme, making at most Q_S queries to the Sign oracle and at most Q_O queries to the H_2 oracle. Then there exists an adversary $\mathcal{B}$ in the* SUF-KO *model for the* GOST-H *scheme making at most $(Q_O + 2)$ queries to the H_2 oracle and exists an adversary $\mathcal{C}$ that breaks the signum-relative collision resistant property of H_1, such that:*

$$\mathsf{Adv}^{\text{SUF-CMA}}_{\text{GOST-H}}(\mathcal{A}) \leqslant \mathsf{Adv}^{\text{SUF-KO}}_{\text{GOST-H}}(\mathcal{B}) + \mathsf{Adv}^{\text{SCR}}_{\mathsf{H}_1}(\mathcal{C}) + \frac{(2Q_O + Q_S + 1)Q_S}{q - 1}.$$

Furthermore, the time complexities of $\mathcal{B}$ and $\mathcal{C}$ are at most $T + c((Q_S + 3)T^V_{\text{GOST-H}} + Q_O)$, where $T^V_{\text{GOST-H}}$ is computational resources needed to verify one signature by the GOST-H.Vf *procedure, c is a constant that depends only on a model of computation and a method of encoding.*

Construction of adversary $\mathcal{C}$. Let $\mathbf{Exp}^0$ denote the original security experiment as defined in the SUF-CMA security model definition (see Figure 1). We fix $\mathcal{A}$ – the adversary that makes forgery for the GOST-H scheme in the SUF-CMA model. The adversary has the access to the random oracle H_2 and to the signing oracle *Sign*. We assume that adversary can make at most Q_O queries to the oracle H_2 and Q_S queries to the oracle *Sign*. Our goal is to upper-bound $\Pr\left[\mathbf{Exp}^{\text{SUF-CMA}}_{\text{GOST-H}}(\mathcal{A}) \Rightarrow 1\right] = \Pr\left[\mathbf{Exp}^0(\mathcal{A}) \Rightarrow 1\right]$.

Note that we change the check for k being equal to zero (see line 4 in the GOST-H.Sig procedure, Section 3.1) to the check for R being equal to zero point (see line 5 in the *Sign* oracle). This change does not affect the scheme but simplifies the proof.

$\mathbf{Exp}^1$ is the modification of the $\mathbf{Exp}^0$ obtained by implementing H_2 using «lazy sampling» (see Figure 2). The idea is to «open» new pairs $(x, H_2(x))$ as soon as the adversary asks for it. We introduce the set Π – the subset of $(\mathbb{Z}_p, \{0,1\}^b)$, which is defined by the union of two sets Π^S and Π^O. We store the pairs obtained from queries to the H_2 oracle in Π^O set and the pairs obtained from queries to the *Sign* oracle in Π^S set. If $(\alpha, \beta) \in \Pi$, we denote β as $\Pi(\alpha)$. We write $(\alpha, \cdot) \in \Pi$ shorthand for the condition that there exists β such that $(\alpha, \beta) \in \Pi$.

This modification does not affect the distribution of H_2 and *Sign* outputs. Thus $\Pr\left[\mathbf{Exp}^0(\mathcal{A}) \Rightarrow 1\right] = \Pr\left[\mathbf{Exp}^1(\mathcal{A}) \Rightarrow 1\right]$.

Proof. $\mathbf{Exp}^0(\mathcal{A}) = \mathbf{Exp}^{\text{SUF-CMA}}_{\mathsf{GOST\text{-}H}}(\mathcal{A})$

1 : $d \xleftarrow{\mathcal{U}} \mathbb{Z}_q^*$
2 : $Q \leftarrow dP$
3 : $H_2 \xleftarrow{\mathcal{U}} Func(\mathbb{Z}_p, \{0,1\}^b)$
4 : $\mathcal{L} \leftarrow \emptyset$

.....Setup completed.....

5 : $(m, \langle r, s\rangle) \xleftarrow{\$} \mathcal{A}^{Sign, H_2}(Q)$
6 : **if** $(m, \langle r, s\rangle) \in \mathcal{L}$: **abort**
7 : **if** $s = 0$: **abort**
8 : $e \leftarrow H_1(m)$
9 : **if** $e = 0 : e \leftarrow 1$
10 : $R \leftarrow e^{-1}sP - e^{-1}rQ$
11 : **if** $\phi(H_2(R.x)) \neq r$: **abort**
12 : **return** 1

Oracle $Sign(m)$

1 : $e \leftarrow H_1(m)$
2 : **if** $e = 0 : e \leftarrow 1$
3 : $k \xleftarrow{\mathcal{U}} \mathbb{Z}_q$
4 : $R \leftarrow kP$
5 : **if** $R = 0$: **return** $\perp$
6 : $r' \leftarrow H_2(R.x)$
7 : $r \leftarrow \phi(r')$
8 : $s \leftarrow ke + dr$
9 : **if** $s = 0$: **return** $\perp$
10 : $\mathcal{L} \leftarrow \mathcal{L} \cup \{(m, \langle r, s\rangle)\}$
11 : **return** $\langle r, s\rangle$

Oracle $H_2(\alpha)$

1 : **return** $H_2(\alpha)$

Figure 1: The $\mathbf{Exp}^0$ for the adversary $\mathcal{A}$ for the $\mathsf{GOST\text{-}H}$ scheme in the SUF-CMA 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 2, lines 7, 8, 9 are added). The $Sign$ and H_2 oracles do not change from the $\mathbf{Exp}^1$.

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 9.

Let construct an adversary $\mathcal{C}$ that breaks the signum-relative collision resistant property of H_1. 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 H_2 implementation with lazy sampling. Otherwise, the polynomial-time bounded adversary could not choose function H_2 randomly from the set $Func(\mathbb{Z}_p, \{0,1\}^b)$ cause the density of this set is exponential. $\mathcal{A}$ delivers a forgery to $\mathcal{C}$, and $\mathcal{C}$ finds the signum-relative collision iff the condition in lines 7-8 is met.

Thus we obtain the following bound:

$$\Pr\left[\mathbf{Exp}^1(\mathcal{A}) \Rightarrow 1\right] - \Pr\left[\mathbf{Exp}^2(\mathcal{A}) \Rightarrow 1\right] \leqslant \mathsf{Adv}^{\text{SCR}}_{\mathsf{H}_1}(\mathcal{C}).$$

The adversary $\mathcal{C}$ implements $\mathbf{Exp}^2$ and thus processes at most Q_S queries to $Sign$ oracle and at most Q_O queries to H_2 oracle, checks the collision condition and verifies the forgery obtained from $\mathcal{A}$. Taking into account that signature generation procedure and hash computation are faster than verification procedure, $\mathcal{C}$ uses at most $c((Q_S+2)T^V_{\mathsf{GOST\text{-}H}} + Q_O)$ additional computational resources, where $T^V_{\mathsf{GOST\text{-}H}}$ is computational resources needed to verify

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

$d \xleftarrow{\mathcal{U}} \mathbb{Z}_q^*$
$Q \leftarrow dP$
$(\Pi^O, \Pi^S) \leftarrow (\emptyset, \emptyset)$
$\Pi \leftarrow \Pi^O \cup \Pi^S$
$\mathcal{L} \leftarrow \emptyset$

.....Setup completed.....

$(m, \langle r, s\rangle) \xleftarrow{\$} \mathcal{A}^{Sign, H_2}(Q)$
if $(m, \langle r, s\rangle) \in \mathcal{L}$: **abort**
if $s = 0$: **abort**
$e \leftarrow H_1(m)$
if $e = 0 : e \leftarrow 1$
$R \leftarrow e^{-1}sP - e^{-1}rQ$
if $\phi(H_2(R.x)) \neq r$: **abort**
return 1

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

$d \xleftarrow{\mathcal{U}} \mathbb{Z}_q^*$
$Q \leftarrow dP$
$(\Pi^O, \Pi^S) \leftarrow (\emptyset, \emptyset)$
$\Pi \leftarrow \Pi^O \cup \Pi^S$
$\mathcal{L} \leftarrow \emptyset$

.....Setup completed.....

$(m, \langle r, s\rangle) \xleftarrow{\$} \mathcal{A}^{Sign, H_2}(Q)$
$\forall (m^*, \cdot) \in \mathcal{L}, m^* \neq m$:
  if $H_1(m^*) = \pm H_1(m)$:
    abort
if $(m, \langle r, s\rangle) \in \mathcal{L}$: **abort**
if $s = 0$: **abort**
$e \leftarrow H_1(m)$
if $e = 0 : e \leftarrow 1$
$R \leftarrow e^{-1}sP - e^{-1}rQ$
if $\phi(H_2(R.x)) \neq r$: **abort**
return 1

Oracle $Sign(m)$

$e \leftarrow H_1(m)$
if $e = 0 : e \leftarrow 1$
$k \xleftarrow{\mathcal{U}} \mathbb{Z}_q$
$R \leftarrow kP$
if $R = 0$: **return** $\perp$
if $(R.x, \cdot) \in \Pi$:
  $r' \leftarrow \Pi(R.x)$
else :
  $r' \xleftarrow{\mathcal{U}} \{0,1\}^b$
  $\Pi^S \leftarrow \Pi^S \cup \{(R.x, r')\}$
  $\Pi \leftarrow \Pi^O \cup \Pi^S$
$r \leftarrow \phi(r')$
$s \leftarrow ke + dr$
if $s = 0$: **return** $\perp$
$\mathcal{L} \leftarrow \mathcal{L} \cup \{(m, \langle r, s\rangle)\}$
return $\langle r, s\rangle$

Oracle $H_2(\alpha)$

if $(\alpha, \cdot) \in \Pi$:
  return $\Pi(\alpha)$
$\beta \xleftarrow{\mathcal{U}} \{0,1\}^b$
$\Pi^O \leftarrow \Pi^O \cup \{(\alpha, \beta)\}$
$\Pi \leftarrow \Pi^O \cup \Pi^S$
return β

Figure 2: The **Exp**^{1} and **Exp**^{2} for the adversary $\mathcal{A}$ for the GOST-H scheme in the SUF-CMA model. The $Sign$ and H_2 oracles are the same in the **Exp**^{1} and **Exp**^{2}

one signature by the GOST-H.Vf procedure, c is a constant that depends only on a model of computation and a method of encoding.

Note that if we find signum-relative collision then we immediately construct a forgery for the GOST-H scheme (it is also true for the GOST scheme) in the SUF-CMA model. It is the interesting property of the GOST signature scheme implied by its construction, namely the equation for s component computation. The probability of such collision event is part of the resulting security bound.

Construction of adversary $\mathcal{B}$. In the further experiments we change the $Sign$ oracle behaviour only (see Figure 3).

Oracle $Sign(m)$ ($\mathbf{Exp}^3$)	Oracle $Sign(m)$ ($\mathbf{Exp}^4$)	Oracle $Sign(m)$ ($\mathbf{Exp}^5$)
1 : $e \leftarrow H_1(m)$	1 : $e \leftarrow H_1(m)$	1 : $e \leftarrow H_1(m)$
2 : **if** $e = 0 : e \leftarrow 1$	2 : **if** $e = 0 : e \leftarrow 1$	2 : **if** $e = 0 : e \leftarrow 1$
3 : $k \xleftarrow{\mathcal{U}} \mathbb{Z}_q$	3 : $r' \xleftarrow{\mathcal{U}} \{0,1\}^b$	3 : $r' \xleftarrow{\mathcal{U}} \{0,1\}^b$
4 : $R \leftarrow kP$	4 : $r \leftarrow \phi(r')$	4 : $r \leftarrow \phi(r')$
5 : **if** $R = 0 :$ **return** $\perp$	5 : $s \xleftarrow{\mathcal{U}} \mathbb{Z}_q$	5 : $s \xleftarrow{\mathcal{U}} \mathbb{Z}_q$
6 : **if** $(R.x, \cdot) \in \Pi :$	6 : $R \leftarrow e^{-1}sP - e^{-1}rQ$	6 : **if** s = 0: **abort**
7 : **abort**	7 : **if** $R = 0 :$ **return** $\perp$	7 : $R \leftarrow e^{-1}sP - e^{-1}rQ$
8 : $r' \xleftarrow{\mathcal{U}} \{0,1\}^b$	8 : **if** $(R.x, \cdot) \in \Pi :$	8 : **if** $R = 0 :$ **return** $\perp$
9 : $\Pi^S \leftarrow \Pi^S \cup \{(R.x, r')\}$	9 : **abort**	9 : **if** $(R.x, \cdot) \in \Pi :$
10 : $\Pi \leftarrow \Pi^O \cup \Pi^S$	10 : $\Pi^S \leftarrow \Pi^S \cup \{(R.x, r')\}$	10 : **abort**
11 : $r \leftarrow \phi(r')$	11 : $\Pi \leftarrow \Pi^O \cup \Pi^S$	11 : $\Pi^S \leftarrow \Pi^S \cup \{(R.x, r')\}$
12 : $s \leftarrow ke + dr$	12 : **if** $s = 0 :$ **return** $\perp$	12 : $\Pi \leftarrow \Pi^O \cup \Pi^S$
13 : **if** $s = 0 :$ **return** $\perp$	13 : $\mathcal{L} \leftarrow \mathcal{L} \cup \{(m, \langle r, s \rangle)\}$	13 : **if** $s = 0 :$ **return** $\perp$
14 : $\mathcal{L} \leftarrow \mathcal{L} \cup \{(m, \langle r, s \rangle)\}$	14 : **return** $\langle r, s \rangle$	14 : $\mathcal{L} \leftarrow \mathcal{L} \cup \{(m, \langle r, s \rangle)\}$
15 : **return** $\langle r, s \rangle$		15 : **return** $\langle r, s \rangle$

Figure 3: The $Sign$ oracles in the $\mathbf{Exp}^3$, $\mathbf{Exp}^4$, $\mathbf{Exp}^5$

The $Sign$ oracle in the $\mathbf{Exp}^3$ is the modification of the $Sign$ oracle in the $\mathbf{Exp}^2$ by adding the abort condition in case of choosing $R.x$ that already belongs to set Π (lines 6-7). We should estimate the probability of this event to estimate the difference between the $\mathbf{Exp}^2$ and $\mathbf{Exp}^3$.

The value k is uniformly distributed in a set $\mathbb{Z}_q^*$ of cardinality $(q-1)$. Thus $R.x$ is uniformly distributed in a set of cardinality $(q-1)/2$. In the worst case the adversary $\mathcal{A}$ has already made all queries to the H_2 oracle and thus Π contains at least Q_O elements. The abort condition is met if the value $R.x$ hits one of elements in Π. We can estimate this probability as

$\frac{Q_O + Q_S/2}{(q-1)/2} = \frac{2Q_O + Q_S}{q-1}$. As these lines are executed at most Q_S times, the overall probability can be bounded by $\frac{(2Q_O + Q_S)Q_S}{q-1}$.

We obtain the following bound:

$$\Pr\left[\mathbf{Exp}^2(\mathcal{A}) \Rightarrow 1\right] - \Pr\left[\mathbf{Exp}^3(\mathcal{A}) \Rightarrow 1\right] \leqslant \frac{(2Q_O + Q_S)Q_S}{q-1}.$$

The signature oracle in the $\mathbf{Exp}^4$ gets along with only public information. Values r' and s are randomly chosen from the relevant sets and then point R is constructed. We define the corresponding pair in H_2 implementation by saving this pair in the Π^S set. Note that if we couldn't do so (i.e., $R.x$ already belongs to the Π), the abort condition is met like in the $\mathbf{Exp}^3$. This step differs from [15] in the order of $\perp$ outputs and **abort** conditions.

Consider the distribution on r', s and $\perp$. Note that if the distributions on r' are identical in both experiments then the distributions on r are identical too. In the $\mathbf{Exp}^3$ r' is distributed uniformly in $\{0,1\}^b$, k is distributed uniformly in $\mathbb{Z}_q^*$ except of the values that lead to $R.x$ that already belongs to Π. k and r are independent from each over in the equation for s, therefore s is distributed uniformly in $\mathbb{Z}_q$ except of the values corresponding to «bad» values of k. In the $\mathbf{Exp}^4$ r' and s are also distributed uniformly on the corresponding sets and s values that lead to the same «bad» events as in the $\mathbf{Exp}^3$ are excluded. The probability of returning $\perp$ is also the same in these experiments.

The probability of abort in the $\mathbf{Exp}^3$ and $\mathbf{Exp}^4$ is the same because R is uniformly distributed in the set of cardinality q in both experiments and zero point is excluded.

Thus we conclude that

$$\Pr\left[\mathbf{Exp}^3(\mathcal{A}) \Rightarrow 1\right] = \Pr\left[\mathbf{Exp}^4(\mathcal{A}) \Rightarrow 1\right].$$

Finally we are moving to the $\mathbf{Exp}^5$. The abort condition in line 6 is added to the signing oracle. Note that line 13 is redundant now, however we keep it for clarity. The qualitative significance of this modification is following: the set Π^S contains only those pairs $(R.x, r')$ that lead to valid signatures now, because the condition in line 13 can never be met. Note that there is no such step in [15].

We estimate the difference between the $\mathbf{Exp}^4$ and $\mathbf{Exp}^5$ by estimating the probability of abort condition in line 6. Per each execution it is equal to $1/q$, so the overall probability can be bounded by Q_S/q.

We obtain the following bound:

$$\Pr\left[\mathbf{Exp}^4(\mathcal{A}) \Rightarrow 1\right] - \Pr\left[\mathbf{Exp}^5(\mathcal{A}) \Rightarrow 1\right] \leqslant \frac{Q_S}{q}.$$

Before constructing the adversary $\mathcal{B}$ we summarize the obtained bounds:

$$\begin{aligned}
&\Pr\left[\mathbf{Exp}^0(\mathcal{A}) \Rightarrow 1\right] - \Pr\left[\mathbf{Exp}^5(\mathcal{A}) \Rightarrow 1\right] = \left(\Pr\left[\mathbf{Exp}^0(\mathcal{A}) \Rightarrow 1\right] - \right.\\
&\left. - \Pr\left[\mathbf{Exp}^1(\mathcal{A}) \Rightarrow 1\right]\right) + \left(\Pr\left[\mathbf{Exp}^1(\mathcal{A}) \Rightarrow 1\right] - \Pr\left[\mathbf{Exp}^2(\mathcal{A}) \Rightarrow 1\right]\right) +\\
&+ \left(\Pr\left[\mathbf{Exp}^2(\mathcal{A}) \Rightarrow 1\right] - \Pr\left[\mathbf{Exp}^3(\mathcal{A}) \Rightarrow 1\right]\right) + \left(\Pr\left[\mathbf{Exp}^3(\mathcal{A}) \Rightarrow 1\right] - \right.\\
&\left. - \Pr\left[\mathbf{Exp}^4(\mathcal{A}) \Rightarrow 1\right]\right) + \left(\Pr\left[\mathbf{Exp}^4(\mathcal{A}) \Rightarrow 1\right] - \Pr\left[\mathbf{Exp}^5(\mathcal{A}) \Rightarrow 1\right]\right) \leqslant\\
&\leqslant \mathsf{Adv}^{\mathrm{SCR}}_{\mathsf{H}_1}(\mathcal{C}) + \frac{(2Q_O + Q_S)Q_S}{q-1} + \frac{Q_S}{q} \leqslant \mathsf{Adv}^{\mathrm{SCR}}_{\mathsf{H}_1}(\mathcal{C}) + \frac{(2Q_O + Q_S + 1)Q_S}{q-1}.
\end{aligned}$$

Let construct the adversary $\mathcal{B}$ for the GOST-H scheme in the SUF-KO model that uses $\mathcal{A}$ as the black box (see Figure 4).

$\mathcal{B}^{H_2^*}(Q)$

$(\Pi^O, \Pi^S) \leftarrow (\emptyset, \emptyset)$
$\Pi \leftarrow \Pi^O \cup \Pi^S$
$\mathcal{L} \leftarrow \emptyset$
$(m, \langle r, s\rangle) \xleftarrow{\$} \mathcal{A}^{SimSign, SimH_2}(Q)$
$\forall (m^*, \cdot) \in \mathcal{L}, m^* \neq m$:
  if $H_1(m^*) = \pm H_1(m)$:
    abort
if $(m, \langle r, s\rangle) \in \mathcal{L}$: **abort**
if $s = 0$: **abort**
$e \leftarrow H_1(m)$
if $e = 0 : e \leftarrow 1$
$R \leftarrow e^{-1}sP - e^{-1}rQ$
if $\phi(SimH_2(R.x)) \neq r$: **abort**
$r' \leftarrow \phi^{-1}(r)$
if $(R.x, r') \in \Pi^S$:
  Find corresponding $(m_1, \langle r_1, s_1\rangle) \in \mathcal{L}$
  Compute d
  $(m, \langle r, s\rangle) \xleftarrow{\$} Sign^{\mathcal{B}}(d, m)$
return $(m, \langle r, s\rangle)$

$SimH_2(\alpha)$

if $(\alpha, \cdot) \in \Pi$:
  return $\Pi(\alpha)$
$\beta \leftarrow H_2^*(\alpha)$
$\Pi^O \leftarrow \Pi^O \cup \{(\alpha, \beta)\}$
$\Pi \leftarrow \Pi^O \cup \Pi^S$
return β

$Sign^{\mathcal{B}}(d, m)$

$e \leftarrow H_1(m)$
if $e = 0 : e \leftarrow 1$
$k \xleftarrow{\mathcal{U}} \mathbb{Z}_q^*$
$R \leftarrow kP$
$r \leftarrow \phi(H_2^*(R.x))$
$s \leftarrow ke + dr$
if $s = 0$:
  $e_1 \leftarrow H_1(m_1)$
  if $e_1 = 0 : e_1 \leftarrow 1$
  $s_1 \leftarrow ke_1 + dr$
return $m_1, \langle r, s_1\rangle$

Figure 4: The adversary $\mathcal{B}$ for the GOST-H scheme in the SUF-KO model that uses the adversary $\mathcal{A}$ for the GOST-H scheme in the SUF-CMA model

Adversary $\mathcal{B}$ simulates the $Sign$ and H_2 oracles using $SimSign$ and $SimH_2$ to answer the $\mathcal{A}$ queries. The $SimSign$ algorithm is similar to the oracle $Sign$ in the $\mathbf{Exp}^5$.

After receiving the forgery from $\mathcal{A}$, $\mathcal{B}$ verifies this forgery by itself. Assume that $\mathcal{A}$ delivers a valid forgery $(m, \langle r, s\rangle)$ (we denote it as $(\widetilde{m}, \langle \widetilde{r}, \widetilde{s}\rangle)$), i.e. the line 15 is reached. This means that the set Π contains the pair $(\widetilde{R}.x, \widetilde{r}')$: either this pair was already in the Π before verification check in line 13 or it was saved after $SimH_2$ call during this check. There are two possible cases. If $(\widetilde{R}.x, \widetilde{r}') \in \Pi^O$, the forgery is already valid with respect to the oracle H_2^* and $\mathcal{B}$ can simply forward it to its own challenger. If $(\widetilde{R}.x, \widetilde{r}') \in \Pi^S$, $\mathcal{B}$ can recover the signing key d as described below and construct the new forgery with the $Sign^{\mathcal{B}}$ algorithm.

Note that the set Π^S contains only such pairs $(R.x, r')$ that result in the valid signatures $\langle r, s\rangle$. This is provided by the $\mathbf{Exp}^5$ modification of the $Sign$ oracle. Thus if $(R.x, r') \in \Pi^S$ the adversary $\mathcal{B}$ can search through $\mathcal{L}$ and find element $(m_1, \langle r_1, s_1\rangle)$, which was established during $\mathcal{A}$ signing queries, meanwhile r_1' and R_1 corresponding to $(m_1, \langle r_1, s_1\rangle)$ satisfy: $r_1' = \widetilde{r}'$, $R_1.x = \widetilde{R}.x$. This search can be realized since it's possible to find all elements in $\mathcal{L}$ with $r_1 = \phi(\widetilde{r}')$, compute $e_1 = H_1(m_1)$ and $R_1 = e_1^{-1}s_1P - e_1^{-1}r_1Q$ and check whether $R_1.x = \widetilde{R}.x$.

The $R_1.x = \widetilde{R}.x$ implies $R_1 = \pm\widetilde{R}$ and thus $k_1 = \pm\widetilde{k}$. So the following linear equation system holds:

$$\begin{cases} \widetilde{s} & = \widetilde{k}\widetilde{e} + d\phi(\widetilde{r}'); \\ s_1 & = \pm\widetilde{k}e_1 + d\phi(\widetilde{r}'); \end{cases}$$

for $\widetilde{e} = H_1(\widetilde{m}), e_1 = H_1(m_1)$. There are two unknown variables $\widetilde{k}$ and d in the system above. Moreover, $\phi(\widetilde{r}') \neq 0$ due to the definition of ϕ. This system has a unique solution whenever $\widetilde{e} \neq \pm e_1$. Observe that case $\widetilde{e} = \pm e_1$ and thus $H_1(\widetilde{m}) = \pm H_1(m_1)$ is excluded by lines 5, 6, 7 if $\widetilde{m} \neq m_1$. The $\widetilde{m} = m_1$ condition (together with $\widetilde{r} = r_1$ condition) implies $(\widetilde{m}, \langle \widetilde{r}, \widetilde{s}\rangle) = (m_1, \langle r_1, s_1\rangle)$ and thus is excluded by line 8. Summing all, we can always compute d if the pair $(\widetilde{R}.x, \widetilde{r}')$ belongs to Π^S.

The only remaining challenge is constructing valid forgery by $\mathcal{B}$ using signing key d. $\mathcal{B}$ invokes $Sign^{\mathcal{B}}$ procedure for the message $\widetilde{m}$ that merely repeats the GOST-H.Sig procedure except for $s = 0$ case. In this case $\mathcal{B}$ constructs the forgery for message m_1, found on the previous step, using the same value of k (and r consequently). We claim that s_1 is always nonzero. This follows from the fact that $\widetilde{e} \neq \pm e_1$ as discussed above and thus $s_1 =$

$ke_1 + dr = ke_1 + (s - k\widetilde{e}) = k(e_1 - \widetilde{e}) \neq 0$. Note that [15] does not consider $s = 0$ case.

We conclude that if $\mathcal{A}$ delivers a valid forgery $(\widetilde{m}, \langle \widetilde{r}, \widetilde{s} \rangle)$ to $\mathcal{B}$, $\mathcal{B}$ delivers a valid forgery to its own challenger and

$$\Pr\left[\mathbf{Exp}^5(\mathcal{A}) \Rightarrow 1\right] = \Pr\left[\mathbf{Exp}^{\text{SUF-KO}}_{\mathsf{GOST\text{-}H}}(\mathcal{B}) \Rightarrow 1\right].$$

All in all we proved:

$$\begin{aligned}\mathsf{Adv}^{\text{SUF-CMA}}_{\mathsf{GOST\text{-}H}}(\mathcal{A}) - \mathsf{Adv}^{\text{SUF-KO}}_{\mathsf{GOST\text{-}H}}(\mathcal{B}) = \Pr\left[\mathbf{Exp}^{\text{SUF-CMA}}_{\mathsf{GOST\text{-}H}}(\mathcal{A}) \Rightarrow 1\right] - \\ - \Pr\left[\mathbf{Exp}^{\text{SUF-KO}}_{\mathsf{GOST\text{-}H}}(\mathcal{B}) \Rightarrow 1\right] = \left(\Pr\left[\mathbf{Exp}^0(\mathcal{A}) \Rightarrow 1\right] - \Pr\left[\mathbf{Exp}^5(\mathcal{A}) \Rightarrow 1\right]\right) + \\ + \left(\Pr\left[\mathbf{Exp}^5(\mathcal{A}) \Rightarrow 1\right] - \Pr\left[\mathbf{Exp}^{\text{SUF-KO}}_{\mathsf{GOST\text{-}H}}(\mathcal{B}) \Rightarrow 1\right]\right) \leqslant \\ \leqslant \mathsf{Adv}^{\text{SCR}}_{\mathsf{H}_1}(\mathcal{C}) + \frac{(2Q_O + Q_S + 1)Q_S}{q - 1}.\end{aligned}$$

Note that the number of queries made by $\mathcal{B}$ to the H_2^* oracle is at most $Q_O + 2$.

The adversary $\mathcal{B}$ needs the same amount of computational resources as $\mathcal{C}$, but it also generates new signature in some cases. Thus $\mathcal{B}$ uses at most $c((Q_S + 3)T^V_{\mathsf{GOST\text{-}H}} + Q_O)$ additional computational resources.

□

B.3 ECDLP to SUF-KO reduction

Theorem 5. *Let $\mathcal{B}$ be an adversary with time complexity at most T in the* SUF-KO *model for the* GOST-H *scheme, making at most Q_O queries to the H_2 oracle. Then there exists an adversary $\mathcal{D}$ that solves the* ECDLP *problem and exists an adversary $\mathcal{M}$ that breaks the signum-relative division resistant property of H_1, such that:*

$$\mathsf{Adv}^{\text{SUF-KO}}_{\mathsf{GOST\text{-}H}}(\mathcal{B}) \leqslant \sqrt{Q_O\left(Q_O \cdot \mathsf{Adv}^{\text{SDR}}_{\mathsf{H}_1}(\mathcal{M}) + \mathsf{Adv}^{\text{ECDLP}}_{\mathbb{G}}(\mathcal{D})\right)} + \frac{Q_O + 1}{2^b}.$$

Furthermore, the time complexities of $\mathcal{D}$ and $\mathcal{M}$ are at most $2T + 2c(Q_O + T^V_{\mathsf{GOST\text{-}H}})$, where $T^V_{\mathsf{GOST\text{-}H}}$ is computational resources needed to verify one signature by the GOST-H.Vf *procedure, c is a constant that depends only on a model of computation and a method of encoding.*

Proof. Let $\mathbf{Exp}^0$ denote the original security experiment as defined in the SUF-KO security model definition (see Figure 5). We fix $\mathcal{B}$ – the adversary that makes forgery for the GOST-H scheme in the SUF-KO model. The adversary has the access to the random oracle H_2, we assume that adversary can make at most Q_O queries to this oracle.

Using the same trick as in Section B.2, we define $\mathbf{Exp}^1$ similar to the $\mathbf{Exp}^0$ but with the H_2 implemented by «lazy sampling» (see Figure 5). As before,

$$\Pr\left[\mathbf{Exp}^0(\mathcal{B}) \Rightarrow 1\right] = \Pr\left[\mathbf{Exp}^1(\mathcal{B}) \Rightarrow 1\right].$$

The $\mathbf{Exp}^2$ is the modification of the $\mathbf{Exp}^1$ in the following way: values $\beta_j,\ j = 1, \ldots, Q_O + 1$, are sampled during experiment initializing phase and H_2 oracle just translates them one by one responding to the queries (see Figure 5). Note that additional β_{Q_O+1} value is sampled since challenger queries the H_2 oracle on the finalization step and one more β is used if Π doesn't contain $(R.x, \cdot)$ element (see line 13). We introduce flag flg to indicate $(R.x, \cdot) \notin \Pi$ and abort experiment in case when β_{Q_O+1} matches $\phi^{-1}(r)$ (see line 14). We estimate the difference between $\mathbf{Exp}^1$ and $\mathbf{Exp}^2$ by estimating the probability of this event. Note that this step differs from [15].

$$\Pr\left[\beta_{Q_O+1} \xleftarrow{\mathcal{U}} \{0,1\}^b;\ \beta_{Q_O+1} = \phi^{-1}(r)\right] \leqslant \frac{1}{2^b}.$$

Therefore, we obtain the following bound:

$$\Pr\left[\mathbf{Exp}^1(\mathcal{B}) \Rightarrow 1\right] - \Pr\left[\mathbf{Exp}^2(\mathcal{B}) \Rightarrow 1\right] \leqslant \frac{1}{2^b}.$$

Construction of algorithm C**.** We construct a deterministic algorithm C that takes a vector $(Q, \beta_1, \ldots, \beta_{Q_O}; \rho)$ as input, invokes the adversary $\mathcal{B}$ on input Q and a random tape derived from ρ and processes the queries to the H_2 oracle as they are processed in the $\mathbf{Exp}^2$ (see Figure 6). Note that random choice in line 4 is made with randomness derived from ρ. Here and after we write «**abort** » as a shortcut for «**return** $\perp$». If $\mathcal{B}$ aborts also C aborts. As $\mathcal{B}$ returns a forgery to C, C validates it and finds the index $j \in \{1, \ldots, Q_O\}$ of the corresponding query to the H_2 oracle. Note that this index always exists cause C aborts if $(R.x, \cdot) \notin \Pi$ before the verification check in line 11. Based on the above,

$$acc = \Pr\left[Q \xleftarrow{\mathcal{U}} \mathbb{G}^*, \beta_1, \ldots, \beta_{Q_O} \xleftarrow{\mathcal{U}} \{0,1\}^b;\ \mathsf{C}(Q, \beta_1, \ldots, \beta_{Q_O}) \not\Rightarrow \perp\right] = \\ = \Pr\left[\mathbf{Exp}^2(\mathcal{B}) \Rightarrow 1\right].$$

Therefore,

$$\Pr\left[\mathbf{Exp}^{\text{SUF-KO}}_{\mathsf{GOST-H}}(\mathcal{B}) \Rightarrow 1\right] - acc = \left(\Pr\left[\mathbf{Exp}^0(\mathcal{B}) \Rightarrow 1\right] - \Pr\left[\mathbf{Exp}^1(\mathcal{B}) \Rightarrow 1\right]\right) + \\ + \left(\Pr\left[\mathbf{Exp}^1(\mathcal{B}) \Rightarrow 1\right] - \Pr\left[\mathbf{Exp}^2(\mathcal{B}) \Rightarrow 1\right]\right) + \left(\Pr\left[\mathbf{Exp}^2(\mathcal{B}) \Rightarrow 1\right] - acc\right) \leqslant \frac{1}{2^b}.$$

$\mathbf{Exp}^0(\mathcal{B}) = \mathbf{Exp}^{\text{SUF-KO}}_{\mathsf{GOST\text{-}H}}(\mathcal{B})$

1 : $d \xleftarrow{\mathcal{U}} \mathbb{Z}_q^*$
2 : $Q \leftarrow dP$
3 : $H_2 \xleftarrow{\mathcal{U}} Func(\mathbb{Z}_p, \{0,1\}^b)$

.....Setup completed.....

4 : $(m, \langle r, s\rangle) \xleftarrow{\$} \mathcal{B}^{H_2}(Q)$
5 : **if** $s = 0$: **abort**
6 : $e \leftarrow H_1(m)$
7 : **if** $e = 0 : e \leftarrow 1$
8 : $R \leftarrow e^{-1}sP - e^{-1}rQ$
9 : **if** $\phi(H_2(R.x)) \neq r$: **abort**
10 : **return** 1

Oracle $H_2(\alpha)$

1 : **return** $H_2(\alpha)$

$\mathbf{Exp}^1(\mathcal{B})$

1 : $d \xleftarrow{\mathcal{U}} \mathbb{Z}_q^*$
2 : $Q \leftarrow dP$
3 : $\Pi \leftarrow \emptyset$

.....Setup completed.....

4 : $(m, \langle r, s\rangle) \xleftarrow{\$} \mathcal{B}^{H_2}(Q)$
5 : **if** $s = 0$: **abort**
6 : $e \leftarrow H_1(m)$
7 : **if** $e = 0 : e \leftarrow 1$
8 : $R \leftarrow e^{-1}sP - e^{-1}rQ$
9 : **if** $\phi(H_2(R.x)) \neq r$: **abort**
10 : **return** 1

Oracle $H_2(\alpha)$

1 : **if** $(\alpha, \cdot) \in \Pi$:
2 : **return** $\Pi(\alpha)$
3 : $\beta \xleftarrow{\mathcal{U}} \{0,1\}^b$
4 : $\Pi \leftarrow \Pi \cup \{(\alpha, \beta)\}$
5 : **return** β

$\mathbf{Exp}^2(\mathcal{B})$

1 : $d \xleftarrow{\mathcal{U}} \mathbb{Z}_q^*$
2 : $Q \leftarrow dP$
3 : $\Pi \leftarrow \emptyset$
4 : $flg \leftarrow$ false
5 : $i \leftarrow 0$
6 : $\beta_1, \ldots, \beta_{Q_O+1} \xleftarrow{\mathcal{U}} \{0,1\}^b$

.....Setup completed.....

7 : $(m, \langle r, s\rangle) \xleftarrow{\$} \mathcal{B}^{H_2}(Q)$
8 : **if** $s = 0$: **abort**
9 : $e \leftarrow H_1(m)$
10 : **if** $e = 0 : e \leftarrow 1$
11 : $R \leftarrow e^{-1}sP - e^{-1}rQ$
12 : **if** $(R.x, \cdot) \in \Pi : flg \leftarrow$ true
13 : **if** $\phi(H_2(R.x)) \neq r$: **abort**
14 : **if** $flg =$ false : **abort**
15 : **return** 1

Oracle $H_2(\alpha)$

1 : **if** $(\alpha, \cdot) \in \Pi$:
2 : **return** $\Pi(\alpha)$
3 : $i \leftarrow i + 1$
4 : $\beta \leftarrow \beta_i$
5 : $\Pi \leftarrow \Pi \cup \{(\alpha, \beta)\}$
6 : **return** β

Figure 5: The $\mathbf{Exp}^0$, $\mathbf{Exp}^1$ and $\mathbf{Exp}^2$ for the adversary $\mathcal{B}$ for the GOST-H scheme in the SUF-KO model

$\mathsf{C}(Q, \beta_1, \ldots, \beta_{Q_O}; \rho)$

1 : $\Pi \leftarrow \emptyset$
2 : $flg \leftarrow \mathsf{false}$
3 : $i \leftarrow 0$
4 : $\beta_{Q_O+1} \xleftarrow{\mathcal{U}} \{0,1\}^b$
5 : $(m, \langle r, s\rangle) \leftarrow \mathcal{B}^{SimH_2}(Q; \rho)$
6 : **if** $s = 0$: **abort**
7 : $e \leftarrow H_1(m)$
8 : **if** $e = 0 : e \leftarrow 1$
9 : $R \leftarrow e^{-1}sP - e^{-1}rQ$
10 : **if** $(R.x, \cdot) \in \Pi : flg \leftarrow \mathsf{true}$
11 : **if** $\phi(SimH_2(R.x)) \neq r$: **abort**
12 : **if** $flg = \mathsf{false}$: **abort**
13 : $r' \leftarrow \phi^{-1}(r)$
14 : find $j \in \{1, \ldots, Q_O\} : r' = \beta_j$
15 : **return** $(j, m, \langle r, s\rangle)$

$\mathsf{Fork}_{\mathsf{C}}(Q)$

1 : Pick random coins ρ for C
2 : $\beta_1, \ldots, \beta_{Q_O} \xleftarrow{\mathcal{U}} \{0,1\}^b$
3 : $(j, m_1, \langle r_1, s_1\rangle) \leftarrow \mathsf{C}(Q, \beta_1, \ldots, \beta_{Q_O}; \rho)$
4 : $\beta'_j, \ldots, \beta'_{Q_O} \xleftarrow{\mathcal{U}} \{0,1\}^b$
5 : **if** $\beta_j = \beta'_j$: **abort**
6 : $\left(j', m_2, \langle r_2, s_2\rangle\right) \leftarrow \mathsf{C}(Q, \beta_1, \ldots, \beta_{j-1}, \beta'_j, \ldots, \beta'_{Q_O}; \rho)$
7 : **if** $j \neq j'$: **abort**
8 : **return** $(m_1, \langle r_1, s_1\rangle, m_2, \langle r_2, s_2\rangle)$

$\mathcal{D}(Q)$

1 : $(m_1, \langle r_1, s_1\rangle, m_2, \langle r_2, s_2\rangle) \leftarrow \mathsf{Fork}_{\mathsf{C}}(Q)$
2 : **if** $H_1(m_1)/r_1 = \pm H_1(m_1)/r_2$: **abort**
3 : compute d
4 : **return** d

Figure 6: The C algorithm that uses the adversary $\mathcal{B}$ for the GOST-H scheme in the SUF-KO model; the $\mathsf{Fork}_{\mathsf{C}}$ algorithm that uses C algorithm and the adversary $\mathcal{D}$ that solves ECDLP problem using $\mathsf{Fork}_{\mathsf{C}}$ algorithm

The algorithm C invokes the adversary $\mathcal{B}$, processes at most Q_O queries to H_2 oracle and verifies the forgery obtained from $\mathcal{B}$. Thus computational complexity of C is at most $T + c(Q_O + T^V_{\mathsf{GOST\text{-}H}})$, where T is the computational resources of $\mathcal{B}$, $T^V_{\mathsf{GOST\text{-}H}}$ is computational resources needed to verify one signature by the $\mathsf{GOST\text{-}H.Vf}$ procedure, c is a constant that depends only on a model of computation and a method of encoding.

We apply the forking lemma (see [15]) and construct the forking algorithm $\mathsf{Fork_C}$ (see Figure 6). If C aborts also $\mathsf{Fork_C}$ aborts. According to the forking lemma the probability frk that $\mathsf{Fork_C}$ terminates without aborting can be estimated as

$$\Pr\left[Q \xleftarrow{\mathcal{U}} \mathbb{G}^*;\ \mathsf{Fork_C}(Q) \not\Rightarrow \perp\right] = frk \geq acc\left(\frac{acc}{Q_O} - \frac{1}{2^b}\right).$$

Construction of adversary $\mathcal{D}$. Finally we construct the adversary $\mathcal{D}$ that solves the ECDLP problem (see Figure 6). At first $\mathcal{D}$ invokes $\mathsf{Fork_C}$ algorithm with the same input as its own input. If $\mathsf{Fork_C}$ aborts also $\mathcal{D}$ aborts. Obtaining two pairs (message, signature) from $\mathsf{Fork_C}$, $\mathcal{D}$ checks whether the condition in line 2 holds and otherwise computes d with the algorithm described below.

Using the pairs obtained from $\mathsf{Fork_C}$ adversary $\mathcal{D}$ computes $e_1 = H_1(m_1), e_2 = H_1(m_2)$ and constructs the following linear system of equations:

$$\begin{cases} R_1 &= e_1^{-1}s_1P - e_1^{-1}r_1Q; \\ R_2 &= e_2^{-1}s_2P - e_2^{-1}r_2Q; \\ r_1 &\neq r_2. \end{cases}$$

By construction of $\mathsf{Fork_C}$ second execution of C differs only since the j-th query of $\mathcal{B}$ to the H_2 oracle. Therefore the j-th input $\alpha = R.x$ to the H_2 oracle was the same in two executions and we claim that $R_1.x = R_2.x$ and thus $R_1 = \pm R_2$. We transform the system above to the following equation

$$e_1^{-1}s_1P - e_1^{-1}r_1Q = \pm e_2^{-1}s_2P \mp e_2^{-1}r_2Q;$$

and compute d by the following formula:

$$d = \frac{e_1^{-1}s_1 \mp e_2^{-1}s_2}{e_1^{-1}r_1 \mp e_2^{-1}r_2}.$$

Note that condition $e_1^{-1}r_1 \mp e_2^{-1}r_2 \neq 0$ holds due to abort in line 2. Summing all, $\mathcal{D}$ computes d as soon as $\mathsf{Fork_C}$ does not abort and condition in line 2 is not met.

Construction of adversary $\mathcal{M}$. We can estimate the probability of aborting in line 2 by constructing an adversary $\mathcal{M} = (\mathcal{M}_1, \mathcal{M}_2)$ for the signum-relative division resistance property (see Figure 7).

The construction of adversary $\mathcal{M}_1$ is quite similar to lines 1-3 of $\mathsf{Fork}_{\mathsf{C}}$ algorithm up to the following difference: adversary $\mathcal{M}_1$ guesses $j^* \in \{1, \ldots, Q_O\}$ (see line 3 of $\mathcal{M}_1$) and puts needed value β to the j^*-position in the C input. If C aborts, also $\mathcal{M}_1$ aborts. Obtaining $(j_1, m_1, \langle r_1, s_1\rangle)$, adversary $\mathcal{M}_1$ checks whether j^* is guessed correctly, and, if so, returns its internal state Γ and m_1 to its own challenger. Adversary $\mathcal{M}_2$ is invoked on input (Γ, β') and simulates lines 4-7 of $\mathsf{Fork}_{\mathsf{C}}$ algorithm up to the following difference: it puts β' to the j^*-position in the C input. If C aborts, also $\mathcal{M}_2$ aborts. Once $\mathcal{M}_2$ gets $(j_2, m_2, \langle r_2, s_2\rangle)$, it checks whether $j_1 = j_2$ and, if so, returns m_2 to its own challenger. Adversary $\mathcal{M}$ wins if $H_1(m_1)/\phi(\beta) = \pm H_1(m_2)/\phi(\beta')$.

$\mathcal{M}_1(\beta)$

1 : $d \xleftarrow{\mathcal{U}} \mathbb{Z}_q^*$
2 : $Q \leftarrow dP$
3 : $j^* \leftarrow \{1, \ldots, Q_O\}$
4 : Pick random coins ρ for C
5 : $\beta_1, \ldots, \beta_{Q_O} \xleftarrow{\mathcal{U}} \{0,1\}^b$
6 : $\beta_{j^*} \leftarrow \beta$
7 : $(j_1, m_1, \langle r_1, s_1\rangle) \leftarrow \mathsf{C}(Q, \beta_1, \ldots, \beta_{Q_O}; \rho)$
8 : **if** $j_1 \neq j^*$: **abort**
9 : $\Gamma \leftarrow (Q, \beta_1, \ldots, \beta_{j^*}, \rho)$
10 : **return** (m_1, Γ)

$\mathcal{M}_2(\Gamma, \beta')$

1 : $(Q, \beta_1, \ldots, \beta_{j^*}, \rho) \leftarrow \Gamma$
2 : $\beta'_{j^*}, \ldots, \beta'_{Q_O} \xleftarrow{\mathcal{U}} \{0,1\}^b$
3 : $\beta'_{j^*} \leftarrow \beta'$
4 : **if** $\beta'_{j^*} = \beta_{j^*}$: **abort**
5 : $(j_2, m_2, \langle r_2, s_2\rangle) \leftarrow \mathsf{C}(Q, \beta_1, \ldots, \beta_{j^*-1}, \beta'_{j^*}, \ldots, \beta'_{Q_O}; \rho)$
6 : **if** $j_1 \neq j_2$: **abort**
7 : **return** m_2

Figure 7: The adversary $\mathcal{M}$ for the SDR property that uses algorithm C

Let denote $\mathsf{Adv}^{\mathrm{SDR}}_{\mathsf{H}_1}(\mathcal{M})$ as $\epsilon^{\mathrm{SDR}}_{\mathsf{H}_1}$. Note that the adversary $\mathcal{M}$ wins if it guesses j^* correctly and the abort condition (see line 2 in $\mathcal{D}$'s pseudocode) is met. Then we can estimate the probability of abort condition as

$$\Pr[H_1(m_1)/r_1 = \pm H_1(m_1)/r_2] \leqslant Q_O \epsilon^{\mathrm{SDR}}_{\mathsf{H}_1}.$$

Therefore we obtain the following bound:

$$\delta = \Pr[\mathcal{D} \text{ solves ECDLP}] = \Pr\left[(\mathsf{Fork}_{\mathsf{C}}(Q) \not\Rightarrow \perp) \wedge \left(\frac{H_1(m_1)}{r_1} \neq \pm\frac{H_1(m_2)}{r_2}\right)\right] =$$

$$= \Pr[\mathsf{Fork}_{\mathsf{C}}(Q) \not\Rightarrow \perp] - \Pr\left[(\mathsf{Fork}_{\mathsf{C}}(Q) \not\Rightarrow \perp) \wedge \left(\frac{H_1(m_1)}{r_1} = \pm\frac{H_1(m_2)}{r_2}\right)\right] \geq$$

$$\geq frk - Q_O\epsilon^{\mathrm{SDR}}_{\mathsf{H}_1} \geq acc\left(\frac{acc}{Q_O} - \frac{1}{2^b}\right) - Q_O\epsilon^{\mathrm{SDR}}_{\mathsf{H}_1} = \frac{1}{Q_O}acc^2 - \frac{1}{2^b}acc - Q_O\epsilon^{\mathrm{SDR}}_{\mathsf{H}_1}.$$

By decision of the following inequation:

$$\frac{1}{Q_O}acc^2 - \frac{1}{2^b}acc - (Q_O\epsilon^{\mathrm{SDR}}_{\mathsf{H}_1} + \delta) \leqslant 0.$$

we can bound the acc value as:

$$acc \leqslant \frac{Q_O}{2^b} + \sqrt{Q_O\left(Q_O\epsilon^{\mathrm{SDR}}_{\mathsf{H}_1} + \delta\right)}.$$

Summarizing all the results, we obtain the final bound to complete the proof:

$$\mathsf{Adv}^{\text{SUF-KO}}_{\mathsf{GOST\text{-}H}}(\mathcal{B}) = \Pr\left[\mathbf{Exp}^{\text{SUF-KO}}_{\mathsf{GOST\text{-}H}}(\mathcal{B}) \Rightarrow 1\right] \leqslant acc + \frac{1}{2^b} \leqslant$$

$$\leqslant \sqrt{Q_O\left(Q_O \cdot \mathsf{Adv}^{\mathrm{SDR}}_{\mathsf{H}_1}(\mathcal{M}) + \mathsf{Adv}^{\mathrm{ECDLP}}_{\mathbb{G}}(\mathcal{D})\right)} + \frac{Q_O + 1}{2^b}.$$

Both $\mathcal{D}$ and $\mathcal{M}$ invoke the algorithm C twice, thus their computational complexities are at most $2T + 2c(Q_O + T^V_{\mathsf{GOST\text{-}H}})$.

□

B.4 Signum-relative division resistance property

In this section we consider the signum-relative division resistance property of H_1 and show that this notion is implied by the standard assumptions: zero resistance and signum-relative preimage resistance properties of H_1.

Let us formally introduce these two properties.

Definition 6 (Zero-resistance property)**.** *For the family of hash functions* H_1

$$\mathsf{Adv}^{\mathrm{ZR}}_{\mathsf{H}_1}(\mathcal{A}) = \Pr\left[m \xleftarrow{\$} \mathcal{A} : H_1(m) = 0\right]$$

Definition 7 (Signum-relative preimage resistance property)**.** *For the family of hash functions* H_1

$$\mathsf{Adv}^{\mathrm{SPR}}_{\mathsf{H}_1}(\mathcal{A}) = \Pr\left[y \xleftarrow{\mathcal{U}} \mathbb{Z}_q^*; m \xleftarrow{\$} \mathcal{A}(y) : H_1(m) = \pm y\right]$$

We construct the adversary $\mathcal{S}$ that breaks the signum-relative preimage property and uses the adversary $\mathcal{M} = (\mathcal{M}_1, \mathcal{M}_2)$ that breaks the signum-relative division resistance property as a black box (see Figure 8).

$\mathcal{S}(y)$

1 : $\beta_1 \xleftarrow{\mathcal{U}} \{0,1\}^b$
2 : $(m_1, \Gamma) \xleftarrow{\$} \mathcal{M}_1(\beta_1)$
3 : **if** $H_1(m_1) = 0$: **abort**
4 : $\beta_2 \leftarrow \phi^{-1}(y \cdot \phi(\beta_1) \cdot (H_1(m_1))^{-1})$
5 : $m_2 \xleftarrow{\$} \mathcal{M}_2(\Gamma, \beta_2)$
6 : **return** m_2

Figure 8: The adversary $\mathcal{S}$ for the SPR property that uses the adversary $\mathcal{M}$ for the SDR property

Let denote $H_1(m_1) = 0$ condition as *Event*. We can estimate the probability of *Event* by constructing an adversary $\mathcal{P}$ that breaks the zero-resistance property. Adversary $\mathcal{P}$ simply simulates lines 1-2 of $\mathcal{S}$ pseudocode and wins if the *Event* takes place. Thus

$$\Pr[Event] = \mathsf{Adv}^{\mathrm{ZR}}_{\mathsf{H}_1}(\mathcal{P}).$$

Consider the distribution on β_2 values. Let denote $\left(y \cdot \phi(\beta_1) \cdot (H_1(m_1))^{-1}\right)$ as γ. As y is chosen randomly from $\mathbb{Z}_q^*$, we claim that γ is distributed uniformly on $\mathbb{Z}_q^*$. Note that $\mathbb{Z}_q^*$ contains less elements than $\mathbb{Z}_{2^{\lceil \log q \rceil}}$ and thus for different values of β_2 the probability $\Pr\left[\gamma \xleftarrow{\mathcal{U}} \mathbb{Z}_q^*;\ \phi^{-1}(\gamma) = \beta_2\right]$ may not be the same. However, we claim that for different values of β_2 this probability will not differ more than by $1/(q-1)$. We find this difference negligible and consider the distribution on β_2 as close to uniform.

If abort condition in line 3 is not met and β_2 is distributed uniformly on $\{0,1\}^b$, the adversary $\mathcal{S}$ realizes the same experiment for $\mathcal{M}$ as in Definition 4. Thus we can estimate the probability of $\mathcal{S}$ success as

$$\Pr[\mathcal{S} \text{ breaks SPR-property}] = \Pr\left[\overline{Event} \wedge (\mathcal{M} \text{ breaks SDR-property})\right] =$$
$$= \Pr[\mathcal{M} \text{ breaks SDR-property}] - \Pr[Event \wedge (\mathcal{M} \text{ breaks SDR-property})] \geqslant$$
$$\geqslant \Pr[\mathcal{M} \text{ breaks SDR-property}] - \Pr[Event].$$

Consequently,

$$\mathsf{Adv}^{\mathrm{SDR}}_{\mathsf{H}_1}(\mathcal{M}) \leqslant \mathsf{Adv}^{\mathrm{SPR}}_{\mathsf{H}_1}(\mathcal{S}) + \mathsf{Adv}^{\mathrm{ZR}}_{\mathsf{H}_1}(\mathcal{P}).$$

C $\mathsf{GenEG_S}$ security

Let $\mathcal{A}$ be an adversary for the $\mathsf{GenEG_S}$ scheme in the SUF-CMA model. We construct the adversary $\mathcal{B}$ for the GenEG scheme in the SUF-CMA model that uses $\mathcal{A}$ as the black box (see Figure 9). Note that $\mathcal{B}$ has the access to its own signing oracle $Sign^*$.

$\mathcal{B}^{Sign^*}(Q)$

1 : $\mathcal{L} \leftarrow \emptyset$
2 : $(m, \overline{r}^* \| \overline{s}) \xleftarrow{\$} \mathcal{A}^{SimSign}(Q)$
3 : **if** $(m, \overline{r}^* \| \overline{s}) \in \mathcal{L}$: **abort**
4 : $\overline{r} \leftarrow \overline{r}^* \| const$
5 : **return** $(m, \overline{r} \| \overline{s})$

$SimSign(m)$

1 : $cnt \leftarrow 0$
2 : **if** $cnt > thr$: **return** $\perp$
3 : $cnt \leftarrow cnt + 1$
4 : $\overline{r} \| \overline{s} \leftarrow Sign^*(m)$
5 : **if** $\overline{r} \| \overline{s} = \perp$: **goto** 2
6 : **if** $\mathsf{lsb}_l(\overline{r}) \neq const$: **goto** 2
7 : $\overline{r}^* = \mathsf{msb}_{|\overline{r}|-l}(\overline{r})$
8 : $\mathcal{L} \leftarrow \mathcal{L} \cup \{(m, \overline{r}^* \| \overline{s})\}$
9 : **return** $\overline{r}^* \| \overline{s}$

Figure 9: The adversary $\mathcal{B}$ for the GenEG scheme in the SUF-CMA model that uses the adversary $\mathcal{A}$ for the $\mathsf{GenEG_S}$ scheme in the SUF-CMA model

Adversary $\mathcal{B}$ invokes $\mathcal{A}$ as a subroutine. $\mathcal{B}$ simulates the $Sign$ oracle for $\mathcal{A}$ with $SimSign$ procedure. Similarly to the $\mathsf{GenEG_S.Sig}$ procedure, $\mathcal{B}$ generates «full» signatures with its own oracle until the $\overline{r}$ component matches the constant vector and truncates $\overline{r}$ before outputting the signature.

Obtaining the forgery from $\mathcal{A}$, $\mathcal{B}$ recovers $\overline{r}$ component by concatenation it with constant vector and forwards it to its own challenger.

If $\mathcal{A}$ makes a valid forgery, $\mathcal{B}$ also makes it. Thus

$$\Pr\left[\mathbf{Exp}^{\text{SUF-CMA}}_{\mathsf{GenEG_S}}(\mathcal{A}) \Rightarrow 1\right] = \Pr\left[\mathbf{Exp}^{\text{SUF-CMA}}_{\mathsf{GenEG}}(\mathcal{B}) \Rightarrow 1\right].$$

Assume that $\mathcal{A}$ makes at most Q_S queries to the signing oracle. Then $\mathcal{B}$ by construction makes at most $Q_S \cdot thr$ queries to its own signing oracle.

D $\mathsf{GenEG^V}$ security

Let $\mathcal{A}$ be an adversary for the $\mathsf{GenEG^V}$ scheme in the SUF-CMA model. We construct the adversary $\mathcal{B}$ for the GenEG scheme in the SUF-CMA model that uses $\mathcal{A}$ as the black box (see Figure 10). Note that $\mathcal{B}$ has the access to its own signing oracle $Sign^*$.

$\mathcal{B}^{Sign^*}(Q)$

1 : $\mathcal{L} \leftarrow \emptyset$
2 : $(m, \overline{r}\|\overline{s}^*) \xleftarrow{\$} \mathcal{A}^{SimSign}(Q)$
3 : **if** $(m, \overline{r}\|\overline{s}^*) \in \mathcal{L}$: **abort**
4 : $i \leftarrow 0$
5 : **if** $i \geq 2^t$: **abort**
6 : $\overline{s} \leftarrow \overline{s}^*\|\mathsf{str}_t(i)$
7 : $i \leftarrow i + 1$
8 : $res \leftarrow \mathsf{GenEG.Vf}(Q, m, \overline{r}\|\overline{s})$
9 : **if** $res = 0$: **goto** 5
10 : **return** $(m, \overline{r}\|\overline{s})$

$SimSign(m)$

1 : $\overline{r}\|\overline{s} \leftarrow Sign^*(m)$
2 : **if** $\overline{r}\|\overline{s} = \perp$: **return** $\perp$
3 : $\overline{s}^* \leftarrow \mathsf{msb}_{|\overline{s}|-t}(\overline{s})$
4 : $\mathcal{L} \leftarrow \mathcal{L} \cup \{(m, \overline{r}\|\overline{s}^*)\}$
5 : **return** $\overline{r}\|\overline{s}^*$

Figure 10: The adversary $\mathcal{B}$ for the GenEG scheme in the SUF-CMA model that uses the adversary $\mathcal{A}$ for the $\mathsf{GenEG}^{\mathsf{V}}$ scheme in the SUF-CMA model

Adversary $\mathcal{B}$ invokes $\mathcal{A}$ as a subroutine. $\mathcal{B}$ simulates the $Sign$ oracle for $\mathcal{A}$ with $SimSign$ procedure. Similarly to the $\mathsf{GenEG}^{\mathsf{V}}$.Sig procedure, $\mathcal{B}$ truncates $\overline{s}$ component before outputting the signature.

Obtaining the forgery from $\mathcal{A}$, adversary $\mathcal{B}$ iterates through all possible variants of $\overline{s}$ and verifies signature until the verification procedure stops with 1. Note that $\mathcal{B}$ needs at most $2^t \cdot T^V_{\mathsf{GenEG}}$ additional computational resources to recover the $\overline{s}$ component, where T^V_{GenEG} is computational resources needed to verify one signature by the GenEG.Vf procedure.

If $\mathcal{A}$ makes a valid forgery, $\mathcal{B}$ also makes it. Thus

$$\Pr\left[\mathbf{Exp}^{\text{SUF-CMA}}_{\mathsf{GenEG}^{\mathsf{V}}}(\mathcal{A}) \Rightarrow 1\right] = \Pr\left[\mathbf{Exp}^{\text{SUF-CMA}}_{\mathsf{GenEG}}(\mathcal{B}) \Rightarrow 1\right].$$

Adversary $\mathcal{B}$ by construction makes the same as $\mathcal{A}$ number of queries to its own signing oracle.

Double point compression for elliptic curves of j-invariant 0

Dmitrii Koshelev

Versailles Saint-Quentin-en-Yvelines University, France
Infotecs, Russia
Institute for Information Transmission Problems, Russia
dishport@yandex.ru

Abstract

The article provides a new double point compression method (to $2\lceil\log_2(q)\rceil + 4$ bits) for an elliptic curve $E_b\colon y^2 = x^3 + b$ of j-invariant 0 over a finite field $\mathbb{F}_q$ such that $q \equiv 1 \pmod 3$. More precisely, we obtain explicit simple formulas transforming the coordinates x_0, y_0, x_1, y_1 of two points $P_0, P_1 \in E_b(\mathbb{F}_q)$ to some two elements of $\mathbb{F}_q$ with four auxiliary bits. In order to recover (in the decompression stage) the points P_0, P_1 it is proposed to extract a sixth root $\sqrt[6]{Z} \in \mathbb{F}_q$ of some element $Z \in \mathbb{F}_q$. It is known that for $q \equiv 3 \pmod 4$, $q \not\equiv 1 \pmod{27}$ this can be implemented by means of just one exponentiation in $\mathbb{F}_q$. Therefore the new compression method seems to be much faster than the classical one with the coordinates x_0, x_1, whose decompression stage requires two exponentiations in $\mathbb{F}_q$.

Keywords: finite fields, pairing-based cryptography, elliptic curves of j-invariant 0, double point compression.

1 Introduction

In many protocols of elliptic cryptography one needs a *compression method* for points of an elliptic curve E over a finite field $\mathbb{F}_q$ of characteristic p. This is done for quick transmission of the information over a communication channel or for its compact storage in a memory. There exists a classical method, which considers an $\mathbb{F}_q$-point on $E \subset \mathbb{A}^2_{(x,y)}$ as the x-coordinate with one auxiliary bit to uniquely recover the y-coordinate by solving the quadratic equation over $\mathbb{F}_q$.

Consider an elliptic curve $E_b : y^2 = x^3 + b$ for $b \in \mathbb{F}_q^*$, which is of j-invariant 0. Ordinary curves of such the form have become very popular in elliptic cryptography, especially in *pairing-based cryptography* [1]. This is due to the existence of (maximally possible) degree 6 twists for them, leading to faster pairing computation [1, §3.3]. One of the latest reviews of standards, commercial products and libraries for this type of cryptography is given in

[2, §5]. Last time, the most popular choice for the 128-bit security level is the so-called Barreto-Lynn-Scott $\mathbb{F}_p$-*curve BLS12-381* [3], where $p \equiv 3 \pmod 4$, $p \equiv 10 \pmod{27}$, and $\lceil \log_2(p) \rceil = 381$.

The simultaneous compression of two points $(x_0, y_0), (x_1, y_1)$ from $E(\mathbb{F}_q)$ (so-called *double point compression*) also has reason to live. It occurs, for example, in pairing-based protocols of succinct *non-interactive zero-knowledge proof (NIZK)*. One of the most notable recent works in this field is [4].

Double point compression has already been discussed in [5] not only for $j(E) = 0$, but in a slightly different way. In that article authors do not try to compress points as compact as possible. Instead of this they find formulas transforming the coordinates x_0, y_0, x_1, y_1 to some three elements of the field $\mathbb{F}_q$. The advantage of their approach is the speed, because it should not solve any equations in the decompression stage.

By virtue of [6, Example V.4.4] the ordinariness of the curve E_b means that $p \equiv 1 \pmod 3$ or, equivalently, $\omega := \sqrt[3]{1} \in \mathbb{F}_p$, where $\omega \neq 1$. There is on E_b the order 6 automorphism $[-\omega] : (x, y) \mapsto (\omega x, -y)$. Consider the geometric quotient $GK'_b := E_b^2/[-\omega]^{\times 2}$, which is an example of so-called *generalized Kummer surface* [7, §1.3].

Our double compression is based on $\mathbb{F}_q$-*rationality* of GK'_b, which is almost obvious (see §3). This concept of algebraic geometry means that for almost all (in some topological sense) points of GK'_b their compression (and subsequent decompression) can be accomplished by computing some rational functions defined over $\mathbb{F}_q$. To recover the original point belonging to $E_b^2(\mathbb{F}_q)$ from a given $\mathbb{F}_q$-point on GK'_b we find an inverse image of the natural map $E_b^2 \to GK'_b$ of degree 6. Since $\omega \in \mathbb{F}_q$, it is a *Kummer map*, that is the field $\mathbb{F}_q(E_b^2)$ is generated by a sixth root of some rational function from $\mathbb{F}_q(GK'_b)$.

In the article [7] the author solves a similar task (almost in the same way), namely the compression task of points from $E_b(\mathbb{F}_{q^2})$, where $q \equiv 1 \pmod 3$, $q \equiv 3 \pmod 4$, and $b \in \mathbb{F}_{q^2}^*$. Its actuality for pairing-based cryptography is explained in the introduction of [7]. There we use so-called *Weil restriction* (*descent*) R_b of E_b with respect to the extension $\mathbb{F}_{q^2}/\mathbb{F}_q$ (see [7, §1.2.1]). For this $\mathbb{F}_q$-surface we have $R_b(\mathbb{F}_q) = E_b(\mathbb{F}_{q^2})$. Besides, the map $[-\omega]$ is naturally induced to the order 6 automorphism $[-\omega]_2 \colon R_b \xrightarrow{\sim} R_b$.

We next consider the generalized Kummer surface $GK_b := R_b/[\omega]_2$ under the order 3 automorphism $[\omega]_2 := ([-\omega]_2)^2$. In order to prove $\mathbb{F}_q$-rationality of GK_b we use quite complicated algebraic geometry (unlike GK'_b). In accordance with [8, §8] from $\mathbb{F}_q$-rationality of GK_b it follows $\mathbb{F}_q$-rationality of the generalized Kummer surface $R_b/[-\omega]_2 \simeq_{\mathbb{F}_q} GK_b/[-1]$. However,

this fact does not provide explicit formulas of a birational $\mathbb{F}_q$-isomorphism $R_b/[-\omega]_2 \xrightarrow{\sim} \mathbb{A}^2$. Nevertheless, such formulas can be easily derived in the same way as for GK'_b (for details see §4).

2 Double compression

For the sake of generality we will consider any pair of elliptic $\mathbb{F}_q$-curves of j-invariant 0, where $q \equiv 1 \pmod 3$, i.e., $\omega \in \mathbb{F}_q$. Namely, for $i = 0, 1$ let $E_i \colon y_i^2 = x_i^3 + b_i$, that is E_{b_i} in our old notation. These curves are isomorphic at most over $\mathbb{F}_{q^6}$ by the map

$$\varphi \colon E_0 \xrightarrow{\sim} E_1, \qquad (x_0, y_0) \mapsto \big(\sqrt[3]{\beta} x_0, \sqrt{\beta} y_0\big),$$

where $\beta := b_1/b_0$. Also, for $k \in \mathbb{Z}/6$ let $\varphi_k := \varphi \circ [-\omega]^k = [-\omega]^k \circ \varphi$ and

$$S_i := \big\{(x_i, y_i) \in E_i \mid x_i y_i = 0\big\} \cup \big\{(0:1:0)\big\} \quad \subset \quad E_i[2] \cup E_i[3].$$

Using the fractions

$$X := \frac{x_0}{x_1}, \qquad Y := \frac{y_0}{y_1},$$

we obtain the compression map

$$\mathrm{com} \colon (E_0 \times E_1)(\mathbb{F}_q) \setminus S_0 \times S_1 \quad \hookrightarrow \quad \mathbb{F}_q^2 \times \mathbb{Z}/6 \times \mathbb{Z}/2,$$

$$\mathrm{com}(P_0, P_1) := \begin{cases} \big(X, Y, n, 0\big) & \text{if} \quad \forall k \in \mathbb{Z}/6 \colon \varphi_k(P_0) \neq P_1, \\ \big(x_0, y_0, k, 1\big) & \text{if} \quad \exists k \in \mathbb{Z}/6 \colon \varphi_k(P_0) = P_1, \end{cases}$$

where $n \in \mathbb{Z}/6$ is the position number of the element $z := x_1 y_1 \in \mathbb{F}_q^*$ in the set $\big\{(-1)^i \omega^j z\big\}_{i=0,j=0}^{1,2}$ ordered with respect to some order in $\mathbb{F}_q^*$. For example, in the case $q = p$ this can be the usual numerical one. Note that the condition $\varphi_k(P_0) = P_1$ is possible only if the isomorphism φ is defined over $\mathbb{F}_q$, that is $\sqrt[6]{\beta} \in \mathbb{F}_q$. Finally, if it is necessary, points from $(S_0 \times S_1)(\mathbb{F}_q)$ can be separately processed, using few additional bits.

3 Double decompression

Let $u := x_1^3$, $v := y_1^2$, and $Z := u^2 v^3 = z^6$. Since $x_0 = X x_1$, we have $x_0^3 = X^3 u$. Hence

$$Y^2 = \frac{y_0^2}{y_1^2} = \frac{x_0^3 + b_0}{x_1^3 + b_1} = \frac{X^3 u + b_0}{u + b_1}$$

and

$$u = \frac{b_0 - b_1 Y^2}{Y^2 - X^3}, \qquad v = u + b_1.$$

Using the number $n \in \mathbb{Z}/6$, we can extract the original sixth root

$$z = x_1 y_1 = \sqrt[3]{u}\sqrt{v} = \sqrt[6]{Z} = \sqrt[3]{\sqrt{Z}}.$$

For $q \equiv 3 \pmod 4$, $q \not\equiv 1 \pmod{27}$ according to [1, §5.1.7], [9, §4]

$$a := \sqrt{Z} = \pm Z^{\frac{q+1}{4}}, \qquad \sqrt[3]{a} = \theta a^e, \qquad \text{hence} \qquad z = \pm\theta Z^{e\frac{q+1}{4}}$$

for some $\theta \in \mathbb{F}_q^*$, $\theta^9 = 1$ and $e \in \mathbb{Z}/(q-1)$. Moreover, e has an explicit simple expression depending only on q. We eventually obtain the equalities

$$x_1 = f_n(X, Y) := \frac{uv}{z^2}, \qquad y_1 = g_n(X, Y) := \frac{z}{x_1}.$$

If $Y^2 = X^3$, then

$$\frac{x_0^3 + b_0}{x_1^3 + b_1} = \frac{x_0^3}{x_1^3} \quad \Leftrightarrow \quad b_0 x_1^3 = b_1 x_0^3 \quad \Leftrightarrow \quad \exists j \in \mathbb{Z}/3 \colon x_1 = \omega^j \sqrt[3]{\beta} x_0.$$

This means that $\varphi_k(P_0) = P_1$ for $k \in \{j, j+3\}$. Thus the decompression map has the form

$$\mathrm{com}^{-1} \colon \mathrm{Im}(\mathrm{com}) \quad \rightrightarrows \quad (E_0 \times E_1)(\mathbb{F}_q) \setminus S_0 \times S_1,$$

$$\mathrm{com}^{-1}\big(t, s, m, bit\big) = \begin{cases} (t f_m, s g_m, f_m, g_m) & \text{if} \quad bit = 0, \\ \big((t, s), \varphi_m(t, s)\big) & \text{if} \quad bit = 1, \end{cases}$$

where $f_m := f_m(t, s)$, $g_m := g_m(t, s)$.

Remark 1. *Although the new point compression-decompression method contains a lot of inversion operations in the field $\mathbb{F}_q$, this is often harmless in regard to timing attacks* [1, §8.2.2, §12.1.1]. *The point is that this type of conversion is mainly applied to public data.*

4 Extension of the compression technique

Our approach still works well for compressing $\mathbb{F}_{q^2}$-points on the curve $E_b \colon y^2 = x^3 + b$, where $b \in \mathbb{F}_{q^2}^*$. For simplicity we take $q \equiv 3 \pmod 4$, i.e., $i := \sqrt{-1} \notin \mathbb{F}_q$. Let $b = b_0 + b_1 i$ (such that $b_0, b_1 \in \mathbb{F}_q$) and

$$x = x_0 + x_1 i, \qquad y = y_0 + y_1 i, \qquad X := \frac{x_0}{x_1}, \qquad Y := \frac{y_0}{y_1}.$$

Building on the equations of the Weil restriction $R_b = \mathrm{R}_{\mathbb{F}_{q^2}/\mathbb{F}_q}(E_b)$ (see [7, §1.2.1]), we obtain

$$u := x_1^3 = \frac{2b_0Y - b_1\gamma(Y)}{\alpha(X)\gamma(Y) - 2\beta(X)Y}, \qquad v := y_1^2 = \frac{\beta(X)u + b_0}{\gamma(Y)},$$

where

$$\alpha(X) := 3X^2 - 1, \qquad \beta(X) := X(X^2 - 3), \qquad \gamma(Y) := Y^2 - 1.$$

As above, the degenerate cases (whenever the denominator of X, Y, u, or v equals 0) can be easily handled independently.

Finally, consider an elliptic $\mathbb{F}_{q^2}$-curve $E_a\colon y^2 = x^3 + ax$ of j-invariant 1728, where $q \equiv 1 \pmod 4$. According to [1, Example 2.28] the latter condition is necessary for the ordinariness of E_a. Our technique also remains to be valid for compressing $\mathbb{F}_q$-points of E_a^2 (if $a \in \mathbb{F}_q^*$) and $\mathbb{F}_{q^2}$-points of E_a, because there is on E_a the $\mathbb{F}_q$-automorphism $[i]\colon (x, y) \mapsto (-x, iy)$ of order 4. However in the second case one needs to take another basis of the extension $\mathbb{F}_{q^2}/\mathbb{F}_q$.

References

[1] El Mrabet N., Joye M., *Guide to Pairing-Based Cryptography*, Cryptography and Network Security Series, Chapman and Hall/CRC, New York, 2016.

[2] Sakemi Y., Kobayashi T., Saito T., Wahby R., *Pairing-friendly curves*, 2020, IETF draft.

[3] Bowe S., *BLS12-381: New zk-SNARK elliptic curve construction*, Zcash Company blog: https://z.cash/blog/new-snark-curve/, 2017.

[4] Groth J., "On the size of pairing-based non-interactive arguments", *LNCS*, Eurocrypt 2016, **9665**, ed. Fischlin M., Coron J.-S., Springer, Berlin, 2016, 305–326.

[5] Khabbazian M., Gulliver T., Bhargava V., "Double point compression with applications to speeding up random point multiplication", *IEEE Transactions on Computers*, **56(3)** (2007), 305–313.

[6] Silverman J., *The Arithmetic of Elliptic Curves*, Graduate Texts in Mathematics, **106**, Springer, New York, 2009.

[7] Koshelev D., *A new elliptic curve point compression method based on $\mathbb{F}_p$-rationality of some generalized Kummer surfaces*, 2019, IACR Cryptology ePrint Archive.

[8] Liedtke C., "Algebraic surfaces in positive characteristic", *Birational Geometry, Rational Curves, and Arithmetic*, Simons Symposia, ed. Bogomolov F., Hassett B., Tschinkel Y., Springer, New York, 2013, 229–292.

[9] Cho G. at al., "New cube root algorithm based on the third order linear recurrence relations in finite fields", *Designs, Codes and Cryptography*, **75(3)** (2015), 483–495.

Quantum cryptography and cryptanalysis

Quantum Differential and Linear Cryptanalysis

Denis Denisenko

Bauman Moscow State Technical University (BMSTU), Russia
denisenkodv@bmstu.ru

Abstract

The work is devoted to the study quantum versions of the differential and linear cryptanalysis based on using a combination of the quantum minimum/maximum search algorithm and the quantum counting algorithm.

We have estimated the complexity and the required resources for applying the quantum differential and quantum linear cryptanalysis to searching round keys of block ciphers $E : V_n \times V_m \to V_m$, $E(key, P) = C$, which is a composition from R round, $E^1 : V_k \times V_m \to V_m$, i.e. $E \equiv (E^1)^R = E^R$. It is shown that the implementation of the quantum linear method requires less logical qubits than for the implementation of the quantum differential method.

The complexity of the quantum differential and linear methods can be less than the complexity of the quantum brute-force key searching by Grover's algorithm, when the encryption $E : V_n \times V_m \to V_m$ and $m < n/2$.

The acceleration of calculations due to "quantum parallelism" in the quantum differential and linear cryptanalysis, based on a combination of Grover's quantum algorithms and quantum counting algorithm, is apparently absent, because the using of quantum counting as "subprogram" in the Grover algorithm eliminates quantum acceleration, as far as $O(\sqrt{K}) \cdot O(\sqrt{K}) \approx O(K)$.

Keywords: Symmetric cryptography, quantum attacks, differential and linear cryptanalysis, block ciphers, Grover's algorithm, quantum counting.

1 Introduction

Quantum computing is one of the areas of quantum technology that has been developing since the end of the 20th century. Attempts to accelerate calculations by using quantum parallelism are used in cryptanalysis, including quantum differential and quantum linear cryptanalysis.

It was shown in [1, 2, 3] that using the Bernstein-Vazirani and Simon's quantum algorithms, we could search for quasilinear structures of Boolean functions and difference relations with significant characteristics with polynomial complexity.

The works [4, 5] are devoted to quantum differential and linear cryptanalysis based on a combination of Grover's algorithm [6] and quantum counting algorithm [7, 8]. Unfortunately, there are some inaccuracies and gaps.

In this regard, the main goal of this work is to describe in detail the quantum differential and linear cryptanalisis methods, to obtain estimates of subkeys searching complexity, and also to display the main problems that may arise during the application of these methods, provided that an ideal universal quantum computer comes into the world and when the difference and linear relationships, with corresponding characteristics, are already known.

2 Quantum differential cryptanalysis

Following [5], we describe a method for recovering the first round key with using quantum algorithms.

Suppose that we have difference $(a,b)_{R-1}$ with the difference characteristic $p_{(a,b)_{R-1}}$ and all $N = 2^m$ possible pair blocks of plain text and cipher text (P_i, C_i), $P_i, C_i \in V_m$, $i \in \overline{0, 2^m - 1}$ received on the same secret key.

Since the difference characteristic is $p_{(a,b)_{R-1}} = \max_{x,y \in V_m \setminus \{0\}} p_{(x,y)_{R-1}}$, it is required to find such a candidate for the first round key K_1 that the probability of the difference relation "If $P'_i \oplus P'_j = a$, then $C_i \oplus C_j = b$" on $R-1$ rounds will be the maximum, $P'_i = E^1(K_1, P_i)$ is the result of applying one iteration of the encryption algorithm to the known block P_i on the round key K_1.

Let $Z(K_1)$ is the number of pairs of plaintext blocks (P'_i, P'_j), $i, j \in \overline{0, 2^m - 1}$ on which the property "If $P'_i \oplus P'_j = a$, then $C_i \oplus C_j = b$" is performed,

$$Z(K_1) = \sum_{i,j=0}^{2^m-1} Ind(C_i \oplus C_j = b | P'_i \oplus P'_j = a),$$

or, if we denote remaining $R-1$ rounds on the true secret key $C_i = E^{R-1}(key, P'_i)$, then

$$Z(K_1) = \sum_{i=0}^{2^m-1} Ind(E^{R-1}(key, P'_i) \oplus E^{R-1}(key, P'_i \oplus a) = b),$$

where $Ind(x) \in \{0,1\}$ is an indicator that the logical expression x is performed.

If on the round key K_1 the probability of the difference (a,b) implementation is maximum, then on the round key K_1 the value of $Z(K_1)$ is also maximum.

Thus the task of finding probable candidates for K_1 is equivalent to optimizing the value of $Z(K_1)$, i.e. finding such round key K_1 on which $Z(K_1) \to max$.

The quantum differential method of cryptanalysis is based on a combination of the quantum algorithm for finding the minimum [9] (in our case finding the maximum) and the quantum counting algorithm (see [7, 8]), due to which computation acceleration is expected compared to the classical case.

In the classical case, in order to find $Z(K_1)$ it is necessary to check all known possible pairs (P_i, C_i), $P_i, C_i \in V_m$, $i \in \overline{0, 2^m - 1}$, i.e. for each pair of blocks (P_i, C_i), calculate $P_i' = E^1(K_1, P_i)$ and check if the property "If $P_i' \oplus P_j' = a$, then $C_i \oplus C_j = b$ " is performed.

In the quantum case, we want to use quantum parallelism, i.e. get a superposition of all possible iterative keys K_1 and the corresponding values $Z(K_1)$. Initial quantum state - state of uniform distributed superposition

$$\sum_{K_1} \frac{1}{\sqrt{2^k}} |K_1\rangle |Z(K_1)\rangle ,$$

to which we will apply the amplitude amplification procedure to search for candidates for the K_1.

First of all, from the known $N = 2^m$ pairs of blocks (P_i, C_i), $i \in \overline{0, N-1}$, it is necessary to prepare the state

$$\sum_{i=0}^{2^m-1} \frac{1}{\sqrt{2^m}} |P_i\rangle |C_i\rangle .$$

This state can be obtained by applying generalized $CNOT(C|t)$ gates, in which qubit t is controlled by the set of qubits C. Generalized gates $CNOT(C|t)$ can be implemented without using ancilla qubits (see [10]), therefore, we will consider generalized gates $CNOT(C|t)$ as one logic gate.

As an example, let's consider the case when blocks of plaintext and ciphertext are three-bit vectors defined in the table 1.

The pair number i	P_i	C_i
0	000	001
1	001	010
2	010	011
3	011	100
4	100	101
5	101	110
6	110	111
7	111	000

Table 1: The table of values (P_i, C_i) for an example of constructing a quantum circuit that prepares the quantum state $\sum_{i=0}^{7} \frac{1}{\sqrt{2^3}} |P_i\rangle |C_i\rangle$.

Quantum circuit for preparing $\sum_{i=0}^{7} \frac{1}{\sqrt{2^3}} |P_i\rangle |C_i\rangle$ represented in figure 1.

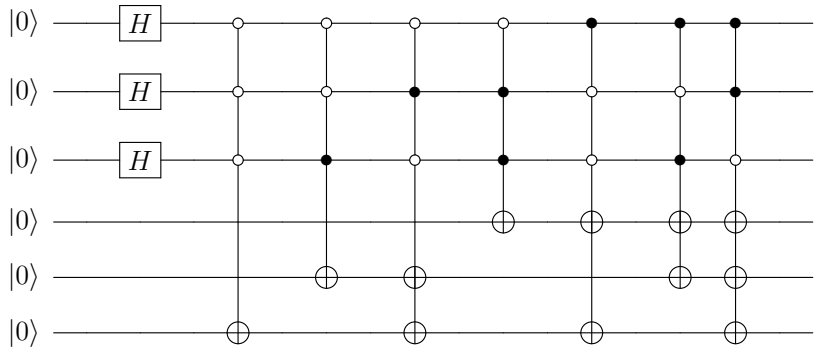


Figure 1: Quantum circuit for preparing $\sum_{i=0}^{7} \frac{1}{\sqrt{2^3}} |P_i\rangle |C_i\rangle$. The upper three qubits correspond to the block P_i, the bottom three qubits correspond to the block C_i, specified in table 1. The highest bit orders of P_i and C_i are on the top, the lower orders are on the bottom. There are no operations required to initialize $|111\rangle |000\rangle$ since the bottom three qubits are already initialized in the states $|0\rangle$.

The figure 1 uses the generalized elements of $CNOT(C|t)$ with several controlled qubits (see [10], section 4.3), which means the composition of several $CNOT(C|t)$ by analogy with the diagram in the figure 2.

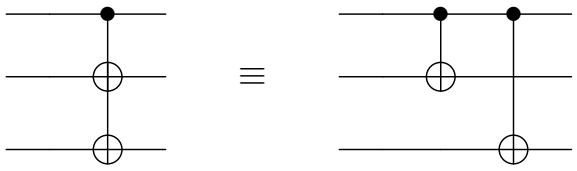


Figure 2: Element $CNOT$ with several controlled qubits.

Let's consider a way to prepare the state $|K_1\rangle |Z(K_1)\rangle$ for an arbitrary first round key $K_1 \in V_k$. We estimate $Z(K_1)$ using the quantum counting algorithm, for which it is necessary to set the corresponding Boolean function f_{K_1} and evaluate $|f_{K_1}^{-1}(1)|$:

Number of a pair P_i, C_i	$x_1\ x_2\ \ldots\ x_m$	$f(x_1, x_2, \ldots, x_m)$
$i = 0$	$00\ldots00$	$Ind(E^{R-1}(key, P'_0) \oplus E^{R-1}(key, P'_0 \oplus a) = b)$
$i = 1$	$00\ldots01$	$Ind(E^{R-1}(key, P'_1) \oplus E^{R-1}(key, P'_1 \oplus a) = b)$
$i = 2$	$00\ldots10$	$Ind(E^{R-1}(key, P'_2) \oplus E^{R-1}(key, P'_2 \oplus a) = b)$
$i = 3$	$00\ldots11$	$Ind(E^{R-1}(key, P'_3) \oplus E^{R-1}(key, P'_3 \oplus a) = b)$
$\vdots$	$\vdots$	$\vdots$
$i = N-1$	$11\ldots11$	$Ind(E^{R-1}(key, P'_{N-1}) \oplus E^{R-1}(key, P'_{N-1} \oplus a) = b)$

Table 2: The table of values of the Boolean function f_{K_1}, by which we evaluate the value of $Z(K_1)$. $P'_i = E^1(K_1, P_i)$ - the result of applying one iteration of the encryption algorithm to the well-known plaintext block P_i on the iteration key K_1.

For preparing the quantum state $\sum_{i=0}^{N-1} \frac{1}{\sqrt{N}} |P_i\rangle |C_i\rangle$, $O(N)$ quan-

tum operations are required. To implement f_{K_1}, we need to prepare $\sum_{i=0}^{N-1} \frac{1}{\sqrt{N}} |P_i\rangle |C_i\rangle$ twice. Figure 3 shows a quantum circuit that implements f_{K_1}.

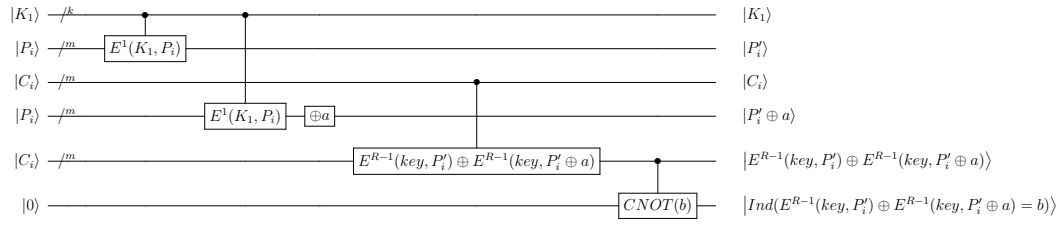


Figure 3: Implementation of f_{K_1} as a quantum circuit. $E^1(K_1, P_i)$ - one iteration of the encryption function on the key K_1 and the plaintext block P_i, which is theoretically possible without the using of ancilla qubits (see [11], [12]) The operation $\oplus a$ is the application of one-qubit gates X on those qubits whose numbers correspond to "1" bits of the difference $a \in V_m$. The operation $E^{R-1}(key, P_i') \oplus E^{R-1}(key, P_i' \oplus a)$ is performed using m standard two-qubit operations $CNOT$, and $CNOT(b)$ - generalized $CNOT$, inverting the lower qubit, in which the difference $b \in V_m$ acts as a control vector.

Note that [5] does not contain a detailed description of the implementation of the function f_{K_1} (see [5], section 3.2), as a result of which understated estimates of the complexity and necessary quantum resources (qubits and gates) are obtained.

In order to get the state $\sum_{K_1} \frac{1}{\sqrt{2^k}} |K_1\rangle |Z(K_1)\rangle$ it is enough to initialize k qubits in the state $|0\rangle$, apply the operator $H^{\otimes k}$ (which consists of k standard Hadamard gates H) to these qubits and perform the quantum counting procedure in the figure 4 (except qubit measurement operations)

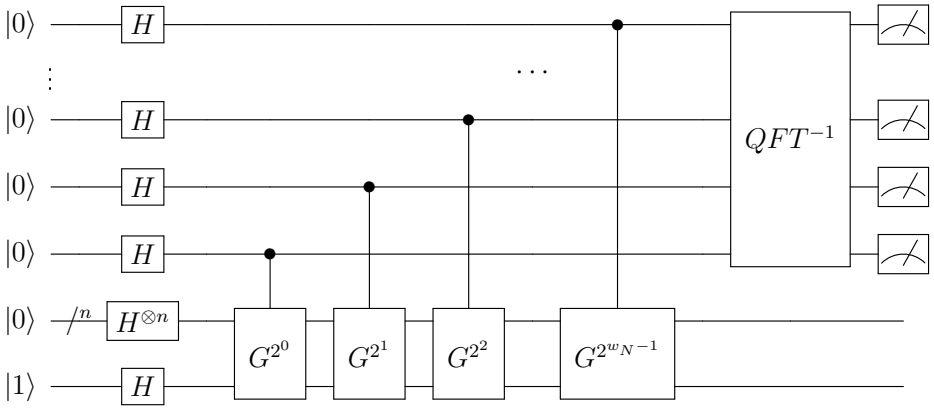


Figure 4: Quantum circuit for determining the angle θ of Grover's iteration G. The control register contains $w_N = \lceil m/2 \rceil + 3$ qubits.

relatively to the boolean function f_{K_1}, the implementation circuit of which is shown in the figure 3. After quantum counting procedure, we obtain a

superposition of iterative keys and corresponding estimates

$$\sum_{K_1} \frac{1}{\sqrt{2^k}} |K_1\rangle |\theta_{K_1}\rangle .$$

The complexity of the quantum counting procedure is $O(2^{w_N})$ Grover iterations with respect to the Boolean function f_{K_1}. The probability of success of the quantum counting procedure is at least $4/\pi^2$.

When we use the Grover algorithm (see [6, 10]) for searching one of the M correct solutions among N possible, $\sqrt{\frac{M}{N}} = \sin\left(\frac{\theta}{2}\right)$, in order to obtain the number $Z(K_1)$ from the estimates θ_{K_1} after applying the quantum counting procedure, it is necessary to calculate $F(\theta) = N \sin^2\left(\frac{\theta}{2}\right)$. However, to search for candidates for the first iteration key K_1, it is not necessary to calculate $F(\theta) = N \sin^2\left(\frac{\theta}{2}\right)$, since $\theta \in [0, 1)$ and if $\theta_1 > \theta_2$, then $N \sin^2\left(\frac{\theta_1}{2}\right) > N \sin^2\left(\frac{\theta_2}{2}\right)$.

To searching for candidates for the first round key K_1, it is enough to get the state $\sum_{K_1} \frac{1}{\sqrt{2^k}} |K_1\rangle |\theta_{K_1}\rangle$. Next, the amplitude amplification algorithm (Grover's algorithm) is used for searching round keys with a maximum value of $Z(K_1)$.

Quantum Differential Cryptanalysis

Input. Block cipher with round encryption $E : V_k \times V_m \to V_m$, $N = 2^m$ pairs of plain text and cipher text blocks (P_i, C_i), $P_i, C_i \in V_m$, $i \in \overline{0, 2^m - 1}$ received on the same secret key, difference relation $(a, b)_{R-1}$ with $p_{(a,b)_{R-1}}$, such that two simple hypotheses can be distinguished by classical statistical methods

$$H_0 : P(\text{"If } P'_i \oplus P'_j = a, \text{ then } C_i \oplus C_j = b\text{"}) \sim Be(1/2^{m-1});$$

$$H_1 : P(\text{"If } P'_i \oplus P'_j = a, \text{ then } C_i \oplus C_j = b\text{"}) \sim Be(p_{(a,b)_{R-1}}),$$

provided $1/2^{m-1} < p_{(a,b)_{R-1}}$, with acceptable error probabilities. The case when $p_{(a,b)_r} = \min_{x,y \in V_m \setminus \{0\}} p_{(x,y)_r}$ and $p_{(a,b)_{R-1}} = 0$ is known as the impossible differential method.

Output. Round key K_1 with a maximum value of $Z(K_1)$.

Preparation step

1: Prepare two registers with quantum states $\sum_{i=0}^{N-1} \frac{1}{\sqrt{N}} |P_i\rangle |C_i\rangle$. The complexity of this stage of $O(N)$ quantum operations.

2: Select a random round key K, calculate the value of $Z(K)$. At this step, we can get the exact value of $Z(K)$ by analysing all N pairs of blocks of open and encrypted text on a classic computer, then calculate the initial value $\theta_{current} := \theta_K = 2 \arcsin \sqrt{\frac{Z(K)}{N}}$.

This step can be performed with the complexity of $O(2^{\lceil \frac{m}{2} \rceil + 3})$ Grover iterations relatively to the Boolean function f_K, with a probability of at least $4/\pi^2$ estimate $\widetilde{\theta}_{current}$, which has a predicted error level $\Delta\widetilde{\theta}_{current} \leq 2^{-3}$ (see [7, 8, 10]).

Search procedure

1: Initialize k qubits in state $|0\rangle$, apply the $H^{\otimes k}$ operator to these qubits, consisting of k standard Hadamard gates H. Thus, we obtain an uniform distributed superposition of keys $\sum_{K_1} \frac{1}{\sqrt{2^k}} |K_1\rangle$.

2: Execute the quantum counting as in the figure 4 with using $w_N = \lceil m/2 \rceil + 3$ control qubits to determine $||f_{K_1}||$ without applying measurements of qubits.

$$f_{K_1} : V_k \times V_m \to V_1,$$

$$f_{K_1}(P_i) = Ind(E^{R-1}(key, P_i') \oplus E^{R-1}(key, P_i' \oplus a) = b).$$

A quantum circuit implementing f_{K_1} is shown in the figure 3.

The complexity of the quantum counting is $O(2^{w_N})$ Grover iterations relatively to the Boolean function f_K, with a probability of at least $4/\pi^2$ we get a superposition of the estimates $\widetilde{\theta}_{K_1}$ which has a predicted error level $\Delta\widetilde{\theta}_{K_1} \leq 2^{-3}$.

Thus, after quantum counting we obtain a superposition of round keys and the corresponding estimates $\sum_{K_1} \frac{1}{\sqrt{2^k}} |K_1\rangle \left|\widetilde{\theta}_{K_1}\right\rangle$ (other registers are omitted for clarity, because they are not involved in the further search procedure), at this step we need $k + 4m + 1 + w_N$ logical qubits.

3: The value $\theta_{current}$ is used as a threshold value, the set of all possible round keys $K \in V_k$ is divided into two classes: the first class of "bad"

keys $\theta_K \leq \theta_{current}$ and the second class of "good" keys $\theta_K > \theta_{current}$. This classification can be described by the Boolean function

$$g(\widetilde{\theta}_{K_1}, \theta_{current}) = Ind(\widetilde{\theta}_{K_1} > \theta_{current}),$$

$g : V_{w_N} \times V_{w_N} \to V_1$.

The Boolean function $g(\widetilde{\theta}_{K_1}, \theta_{current})$ can be implemented as follows. We could initialize one ancilla qubit $|q\rangle = |1\rangle$, and consider it as high order bit of binary decomposition $\theta_{current}$. If after subtraction in the binary number system $\theta_{current} - \widetilde{\theta}_{K_1}$, the high order bit is changed and will be "zero" (we get $|q\rangle = |0\rangle$), then $\widetilde{\theta}_{K_1} > \theta_{current}$. This method of comparing two integers was proposed by [16] and could be implemented by inverting a quantum circuit that implements the operation of adding two integers (see [17], [18], [19]).

Perform the quantum counting procedure as in the figure 4 with using $w_K = \lceil k/2 \rceil + 3$ control qubits – with a probability of at least $4/\pi^2$ and complexity $O(2^{w_K})$ Grover iterations relatively to the Boolean function g, we obtain the estimate θ_g, by which we find

$$||g|| \approx \widetilde{M}_g = 2^k \cdot \sin^2\left(\frac{\theta_g}{2}\right).$$

In order to guarantee the condition $||g|| < 2^k/2$ needed for the quantum counting procedure, we can increase the search space by adding one ancilla qubit (see [10]). We need at least $k + 4m + 1 + w_N + (w_K + 3)$ logical qubits, where in $(w_K + 3)$ we mean 1 ancilla qubit is taken into account for the implementation of the subtraction when implementing g, 1 ancilla qubit to increase the search space and 1 ancilla qubit to implement the Grover iteration relatively g.

4: Apply the Grover algorithm to search for the first round key relative to the Boolean function g. After $\left[\frac{\pi}{4}\sqrt{\frac{2^k}{\widetilde{M}_g}}\right]$ Grover iterations and measurement of qubits with probability $\sin^2(\frac{2\left[\frac{\pi}{4}\sqrt{\frac{2^k}{\widetilde{M}_g}}\right]+1}{2}\theta_g)$ we get one from $\widetilde{M}_g$ possible solutions, i.e. such a key is K_1' on which $\theta_{K_1'} > \theta_{current}$. Set $\theta_{current} = \theta_{K_1'}$ and repeat the search procedure again. To implement the Grover algorithm relatively g we need $k + 4m + 1 + w_N + 2$ logical qubits, one ancilla qubit needs to implement the subtraction with implementation of g and another one qubit needs to implement the Grover iteration.

If each quantum counting procedure execute correctly, then to search for the key K with the maximum value θ_K, i.e. with the maximum value of $Z(K)$, it is required no more than $\widetilde{M}_g$ iterations of the search procedure (for $\widetilde{M}_g$ obtained at the first start of the search procedure).

The lower bound of search procedure success probability can be estimated by $\frac{4}{\pi^2} \cdot \frac{4}{\pi^2} \cdot 0.5 = 0.0821279$ (this estimate is for the case when the quantum counting procedure performed only twice and the success probability of the Grover's algorithm for searching maximum > 0.5), i.e. on average, it will take about 12 starts to find the new value of $\theta_{current}$.

The complexity of each searching procedure is at least

$$O\left(2^{w_N} + 2^{w_K} + \left[\frac{\pi}{4}\sqrt{\frac{2^k}{\widetilde{M}_g}}\right]\right)$$

quantum operations.

Remark 1. *1. The work [15] presents the results of a successful simulation of SDES key searching by using Grover's algorithm in the Quipper quantum simulator. The correct scheme for implementing one Grover iteration is shown in the figure 5.*

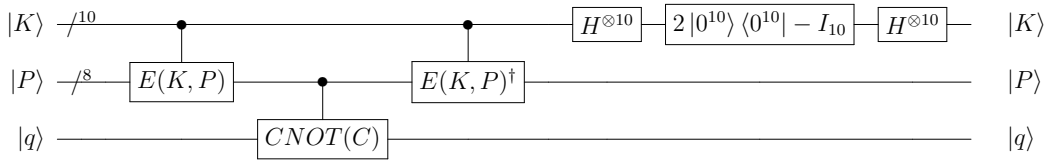


Figure 5: One iteration of the Grover's algorithm for SDES key searching [15].

2. It might seems that quantum circuit in the figure 6 also work correctly, where the encryption function $E(K, P)$ in figure 6 is implemented only once. However, the results of experiments in Quipper quantum simulator indicate that quantum circuit in figure 6 does not lead to success, i.e. "inversion about mean" doesn't increase the amplitude of target secret key. Consequently, we have to implement the encryption function $E(K, P)$ as quantum circuit and the inverse quantum circuit $E(K, P)^\dagger$ (after inverting the flag qubit) at each iteration of the Grover's algorithm.

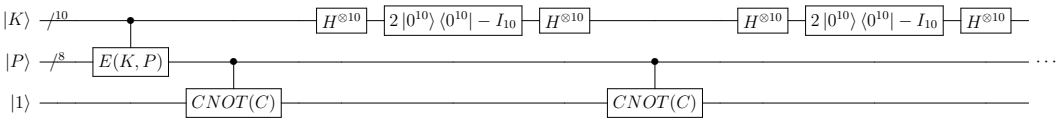


Figure 6: Wrong iteration of the Grover's algorithm for SDES key searching [15].

It follows from the fact that the two-qubit CNOT operation swaps the corresponding amplitudes (see [10]):

$$|q_1q_2\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle \xrightarrow{CNOT(q_1|q_2)}$$

$$\xrightarrow{CNOT(q_1|q_2)} \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|11\rangle + \alpha_{11}|10\rangle,$$

therefore, in order for "inversion about mean" increases amplitude of the target secret key, we have to implement $E(K,P)^\dagger$.

3. *Therefore, at step 4 of presented search procedure,* **quantum counting relatively to the boolean function** f_{K_1} **for preparing** $\sum_{K_1}\frac{1}{\sqrt{2^k}}|K_1\rangle\left|\widetilde{\theta}_{K_1}\right\rangle$ **should be performed at each Grover iteration!** *The general view of the correct quantum circuit that implements one Grover iteration at step 4 is shown in the figure 7.*

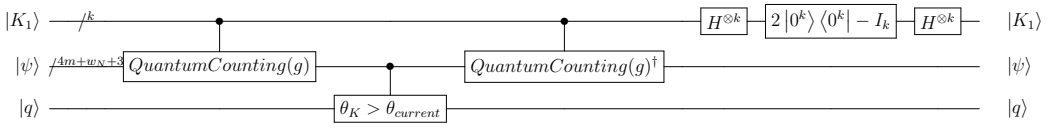


Figure 7: The correct implementation of one Grover's algorithm iteration with using quantum counting as a "subprogram".

4. *In case when the quantum countings in step 4 are performed at each Grover's iteration, the inversion of the flag qubit in Grover's algorithm that correspond to the indicator of the event* $\widetilde{\theta}_{K_1} > \theta_{current}$ *is performed with a probability at least* $\frac{4}{\pi^2}$.

   Then, if the correct executions of quantum countings are independent events, the lower bound of the success probability of the entire algorithm should be multiplied by $\left(\frac{4}{\pi^2}\right)^{\left\lceil\frac{\pi}{4}\sqrt{\frac{2^k}{\widetilde{M}_g}}\right\rceil}$. *For simplicity, we will not take this into account, as if the success probability of quantum counting at each Grover's iteration is equal to one.*

5. *For completeness, we present estimates of complexity in two cases. In the first case, in the author's opinion, the wrong case, the quantum counting at step 4 is performed once, i.e. get an understated estimates of the complexity. In the second case, at step 4, the quantum counting procedure and its inversion is performed at each Grover iteration.*

Conditions	The complexity of the quantum differential cryptanalysis (number of quantum operations)
1.At step 4, quantum counting is performed only once (false)	$O(N + 2^{w_N} + 2^{w_K} + 2^{w_K} \cdot 1 + \left[\frac{\pi}{4}\sqrt{\frac{2^k}{\widetilde{M}_g}}\right])$
2.At step 4, quantum counting and its inversion are performed at each Grover iteration (true).	$O(N + 2^{w_N} + 2^{w_K} + 2^{w_K+1} \cdot \left[\frac{\pi}{4}\sqrt{\frac{2^k}{\widetilde{M}_g}}\right])$

Table 3: Estimates of the complexity of the quantum differential cryptanalysis:
1) $O(N + 2^{w_N} + 2^{w_K} + 2^{w_K} \cdot 1 + \left[\frac{\pi}{4}\sqrt{\frac{2^k}{\widetilde{M}_g}}\right])$ - an understated estimates of the complexity, optimistic from the point of view of a quantum cryptanalyst.
2) $O(N + 2^{w_N} + 2^{w_K} + 2^{w_K+1} \cdot \left[\frac{\pi}{4}\sqrt{\frac{2^k}{\widetilde{M}_g}}\right])$ - the correct estimate.
3) Required at least $k + 4m + 1 + w_N + (w_K + 3)$ logical qubits, where $w_N = \lceil m/2 \rceil + 3$, $w_K = \lceil k/2 \rceil + 3$.

Obtained estimates of the complexity of quantum differential cryptanalysis are more accurate than estimate $O(2^{k/2} + 2^{m/2})$ from [4] and [5]. If the number of known pairs $N < 2^m$, the scheme for applying the quantum differential cryptanalysis does not change, it is possible to reduce the complexity at the preparation step.

3 Quantum linear cryptanalysis

By analogy with the section 2, we describe a quantum linear cryptanalysis for recovering the first round key with using quantum algorithms.

Suppose we have a linear relationship $(a, b)_{R-1}$ with characteristic $p_{(a,b)_{R-1}}$ and $N = 2^m$ of all possible pairs (P_i, C_i), $P_i, C_i \in V_m$, $i \in \overline{0, 2^m - 1}$ received on the same secret key.

We consider the case when $p_{(a,b)_{R-1}} = \max_{x,y \in V_m \setminus \{0\}} p_{(x,y)_{R-1}}$ and we need to find such a candidate for the first round key K_1, that the probability of the linear relation $\langle P'_i, a\rangle = \langle C_i, b\rangle$ in $R-1$ rounds is maximal ($\langle x, y\rangle$ - the scalar product of binary vectors $x, y \in V_m$ in $GF(2)$, $P'_i = E^1(K_1, P_i)$ - the result of applying one iteration of the encryption algorithm to the known plaintext block P_i on the key K_1).

Let $Z(K_1)$ is the number of pairs (P'_i, C_i), $i \in \overline{0, 2^m - 1}$ on which the linear relation $\langle P'_i, a\rangle = \langle C_i, b\rangle$ is realized,

$$Z(K_1) = \sum_{i=0}^{2^m-1} Ind(\langle P'_i, a\rangle = \langle C_i, b\rangle),$$

where $Ind(x) \in \{0, 1\}$ is an indicator that the logical expression x is satisfied.

If on the round key K_1 the probability of the linear relation (a, b) is maximum, then on the round key K_1 the value of $Z(K_1)$ is also maximum.

Thus the task of finding probable candidates for K_1 is equivalent to optimizing the value of $Z(K_1)$, i.e. finding such round key K_1 on which $Z(K_1) \to max$.

In the classical case, in order to find $Z(K_1)$ using a known material on round key K_1, it is necessary to check all known material, i.e. for each pair of blocks (P_i, C_i), calculate $P'_i = E^1(K_1, P_i)$ and check if the property $\langle P'_i, a\rangle = \langle C_i, b\rangle$ is realized.

In the quantum case, as in quantum differential cryptanalysis, we want to use quantum parallelism, i.e. get a superposition of all possible iterative keys K_1 and the corresponding values $Z(K_1)$. Initial quantum state is the state of uniform distributed superposition

$$\sum_{K_1} \frac{1}{\sqrt{2^k}} |K_1\rangle |Z(K_1)\rangle ,$$

to which we will apply the amplitude amplification procedure to search for candidates for the K_1.

First of all, from the known $N = 2^m$ pairs of blocks $(P_i, C_i),\ i \in \overline{0, N-1}$, it is necessary to prepare the state

$$\sum_{i=0}^{2^m-1} \frac{1}{\sqrt{2^m}} |P_i\rangle |C_i\rangle .$$

This state can be obtained by applying generalized $CNOT(C|t)$ gates, in which qubit t is controlled by the set of qubits C. Generalized gates $CNOT(C|t)$ can be implemented without using ancilla qubits (see [10]), therefore, we will consider generalized gates $CNOT(C|t)$ as one logic gate.

An example of initializing $\sum_{i=0}^{N-1} \frac{1}{\sqrt{N}} |P_i\rangle |C_i\rangle$ is presented in section 2 (see table 1 and figure 1).

Let's consider a way to prepare the state $|K_1\rangle |Z(K_1)\rangle$ for an arbitrary first round key $K_1 \in V_k$. We estimate $Z(K_1)$ using the quantum counting algorithm, for which it is necessary to set the corresponding Boolean function f_{K_1} and evaluate $|f_{K_1}^{-1}(1)|$:

Number of a pair P_i, C_i	$x_1\ x_2\ \ldots\ x_m$	$f(x_1, x_2, \ldots, x_m)$
$i = 0$	$00\ldots00$	$Ind(\langle P'_0, a\rangle = \langle C_0, b\rangle)$
$i = 1$	$00\ldots01$	$Ind(\langle P'_1, a\rangle = \langle C_1, b\rangle)$
$i = 2$	$00\ldots10$	$Ind(\langle P'_2, a\rangle = \langle C_2, b\rangle)$
$i = 3$	$00\ldots11$	$Ind(\langle P'_3, a\rangle = \langle C_3, b\rangle)$
$\vdots$	$\vdots$	$\vdots$
$i = N-1$	$11\ldots11$	$Ind(\langle P'_{N-1}, a\rangle = \langle C_{N-1}, b\rangle)$

Table 4: The table of values of the Boolean function f_{K_1}, by which we evaluate the value of $Z(K_1)$. $P'_i = E^1(K_1, P_i)$ - the result of applying one iteration of the encryption algorithm to the known plaintext block P_i on the iteration key K_1.

To prepare the quantum state $\sum_{i=0}^{N-1} \frac{1}{\sqrt{N}} |P_i\rangle |C_i\rangle$, $O(N)$ we need quantum operations. To implement f_{K_1}, in contrast to section 2, the state $\sum_{i=0}^{N-1} \frac{1}{\sqrt{N}} |P_i\rangle |C_i\rangle$ have to be prepared once. Figure 8 shows a quantum circuit that implements f_{K_1} .

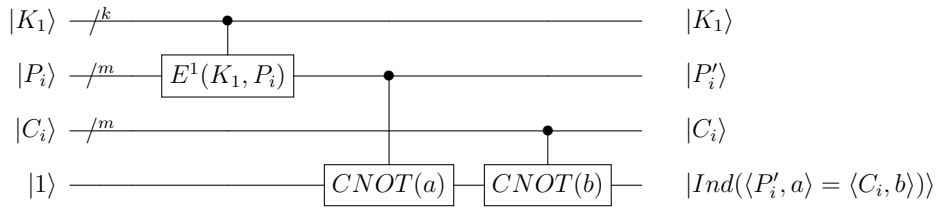


Figure 8: Implementation of f_{K_1} as a quantum circuit. $E^1(K_1, P_i)$ - one iteration of the encryption function on the key K_1 and the plaintext block P_i, which is theoretically possible without the use of ancilla qubits (see [11], [12]) Here the operations $CNOT(a)$ and $CNOT(b)$ are sets of $||a||$ and $||b||$ of standard two-qubit gates $CNOT$, for which control bits correspond to "1" bits in binary representation of $a \in V_m$ and $b \in V_m$.

In order to get the state $\sum_{K_1} \frac{1}{\sqrt{2^k}} |K_1\rangle |Z(K_1)\rangle$ it is enough to initialize k qubits in the state $|0\rangle$, apply the operator $H^{\otimes k}$ (which consists of k standard Hadamard gates H) to these qubits and perform the quantum counting procedure as in the figure 4 (except for measurement operations qubits) relatively to the Boolean function f_{K_1}, the implementation of which is shown in the figure 8. After the quantum counting procedure, we obtain a superposition of round keys and estimates

$$\sum_{K_1} \frac{1}{\sqrt{2^k}} |K_1\rangle |\theta_{K_1}\rangle .$$

The complexity of the quantum counting procedure is $O(2^{w_N})$ Grover iterations relatively the Boolean function f_{K_1}. The success probability of the quantum counting is at least $4/\pi^2$.

Since in the Grover algorithm when we search one of the M correct solutions among N possible $\sqrt{\frac{M}{N}} = \sin\left(\frac{\theta}{2}\right)$, in order to obtain the number $Z(K_1)$ from the estimates θ_{K_1} after applying the quantum counting procedure, it is necessary to calculate $F(\theta) = N\sin^2\left(\frac{\theta}{2}\right)$. However, as in section 2, it is not necessary, since $\theta \in [0,1)$ and if $\theta_1 > \theta_2$, then $N\sin^2\left(\frac{\theta_1}{2}\right) > N\sin^2\left(\frac{\theta_2}{2}\right)$.

To search for candidates for the first round key K_1, it is enough to get the state $\sum_{K_1} \frac{1}{\sqrt{2^k}} |K_1\rangle |\theta_{K_1}\rangle$. Next, the amplification amplification algorithm (Grover's algorithm) is used to search for iterative keys with a maximum value of $Z(K_1)$.

Quantum linear cryptanalysis

Input. Block cipher with round encryption $E : V_k \times V_m \to V_m$, $N = 2^m$ pairs of plain text and cipher text blocks (P_i, C_i), $P_i, C_i \in V_m$, $i \in \overline{0, 2^m - 1}$ received on the same secret key, the linear relation $(a,b)_{R-1}$ with characteristic $p_{(a,b)_{R-1}}$, such that two simple hypotheses can be distinguished by classical statistical methods

$$H_0 : \langle P_i', a\rangle = \langle C_i, b\rangle \sim Be(1/2) \;\; vs \;\; H_1 : \langle P_i', a\rangle = \langle C_i, b\rangle \sim Be(p_{(a,b)_{R-1}}),$$

provided $1/2 < p_{(a,b)_{R-1}}$, with acceptable error probabilities.

The case when $p_{(a,b)_r} = \min_{x,y \in V_m \setminus \{0\}} p_{(x,y)_r}$ and $p_{(a,b)_{R-1}} < 1/2$ is considered similarly, except that it is necessary to find the round key K_1 with the minimum value $Z(K_1)$.

Exit. Round key K_1 with a maximum value of $Z(K_1)$.

Preparation step

1: Prepare two registers with quantum states $\sum_{i=0}^{N-1} \frac{1}{\sqrt{N}} |P_i\rangle |C_i\rangle$. The complexity of this stage of $O(N)$ quantum operations.

2: Select a random round key K, calculate the value of $Z(K)$. At this step, we can get the exact value of $Z(K)$ by analysing all N pairs of blocks of open and encrypted text on a classic computer, then calculate the initial value $\theta_{current} := \theta_K = 2\arcsin\sqrt{\frac{Z(K)}{N}}$.

This step can be performed with the complexity of $O(2^{\lceil \frac{m}{2} \rceil + 3})$ Grover iterations relatively to the Boolean function f_K, with a probability of at least $4/\pi^2$ estimate $\widetilde{\theta}_{current}$, which has a predicted error level $\Delta\widetilde{\theta}_{current} \le 2^{-3}$ (see [7, 8, 10]).

Search procedure

1: Initialize k qubits in state $|0\rangle$, apply the $H^{\otimes k}$ operator to these qubits, consisting of k standard Hadamard gates H. Thus, we obtain an uniform distributed superposition of keys $\sum_{K_1} \frac{1}{\sqrt{2^k}} |K_1\rangle$.

2: Execute the quantum counting as in the figure 4 with using $w_N = \lceil m/2 \rceil + 3$ control qubit to determine $||f_{K_1}||$ without applying measurement of qubits. $f_{K_1} : V_k \times V_m \to V_1$, $f_{K_1}(P_i) = Ind(\langle P_i', a\rangle = \langle C_i, b\rangle)$. A quantum circuit for implementing f_{K_1} is shown in the figure 8.

The complexity of the quantum counting is $O(2^{w_N})$ Grover iterations relatively to the Boolean function f_K, with a probability of at least $4/\pi^2$ we get a superposition of the estimates $\widetilde{\theta}_{K_1}$ which has a predicted error level $\Delta\widetilde{\theta}_{K_1} \leq 2^{-3}$.

Thus, after the quantum counting, we obtain a superposition of round keys and the corresponding estimates $\sum_{K_1} \frac{1}{\sqrt{2^k}} |K_1\rangle \left|\widetilde{\theta}_{K_1}\right\rangle$ (other registers are omitted for clarity, because they are not involved in the further search procedure), at this stage we need $k + 2m + 1 + w_N$ logical qubits.

3: The value $\theta_{current}$ is used as a threshold value, the set of all possible round keys $K \in V_k$ is divided into two classes: the first class of "bad" keys $\theta_K \leq \theta_{current}$ and the second class of "good" keys $\theta_K > \theta_{current}$. This classification can be described by the Boolean function

$$g(\widetilde{\theta}_{K_1}, \theta_{current}) = Ind(\widetilde{\theta}_{K_1} > \theta_{current}),$$

$g : V_{w_N} \times V_{w_N} \to V_1$.

The Boolean function $g(\widetilde{\theta}_{K_1}, \theta_{current})$ can be implemented as follows. We could initialize one ancilla qubit $|q\rangle = |1\rangle$, and consider it as high order bit of binary decomposition $\theta_{current}$. If after subtraction in the binary number system $\theta_{current} - \widetilde{\theta}_{K_1}$, the high order bit is changed and will be "zero" (we get $|q\rangle = |0\rangle$), then $\widetilde{\theta}_{K_1} > \theta_{current}$. This method of comparing two integers was proposed by [16] and could be implemented by inverting a quantum circuit that implements the operation of adding two integers (see [17], [18], [19]).

Perform the quantum counting procedure as in the figure 4 with using $w_K = \lceil k/2 \rceil + 3$ control qubits – with a probability of at least $4/\pi^2$ and complexity $O(2^{w_K})$ Grover iterations relatively the Boolean function g,

we obtain the estimate θ_g, by which we find

$$||g|| \approx \widetilde{M}_g = 2^k \cdot \sin^2\left(\frac{\theta_g}{2}\right).$$

In order to guarantee the condition $||g|| < 2^k/2$ needed for the quantum counting procedure, you can increase the search space by adding one ancilla qubit (see [10]). We need at least $k + 2m + 1 + w_N + (w_K + 3)$ logical qubits, where in $(w_K + 3)$ we mean 1 ancilla qubit is taken into account for the implementation of the subtraction when implementing g, 1 ancilla qubit to increase the search space and 1 ancilla qubit to implement the Grover iteration relatively g.

4: Apply the Grover algorithm to search for the first round key relative to the Boolean function g. After $\left[\frac{\pi}{4}\sqrt{\frac{2^k}{\widetilde{M}_g}}\right]$ Grover iterations and measurement of qubits with probability $\sin^2(\frac{2\left[\frac{\pi}{4}\sqrt{\frac{2^k}{\widetilde{M}_g}}\right]+1}{2}\theta_g)$ we get one from $\widetilde{M}_g$ possible solutions, i.e. such a key is K_1' on which $\theta_{K_1'} > \theta_{current}$. Set $\theta_{current} = \theta_{K_1'}$ and repeat the search procedure again. To implement the Grover algorithm relatively g we need $k + 2m + 1 + w_N + 2$ logical qubits, one ancilla qubit needs to implement the subtraction with implementation of g and another one qubit needs to implement the Grover iteration.

If each quantum counting procedure execute correctly, then to search for the key K with the maximum value θ_K, i.e. with the maximum value of $Z(K)$, it is required no more than $\widetilde{M}_g$ iterations of the search procedure (for $\widetilde{M}_g$ obtained at the first start of the search procedure).

The lower bound of search procedure success probability can be estimated by $\frac{4}{\pi^2} \cdot \frac{4}{\pi^2} \cdot 0.5 = 0.0821279$ (this estimate is for the case when the quantum counting procedure performed only twice and the success probability of the Grover's algorithm for searching maximum > 0.5), i.e. on average, it will take about 12 starts to find the new value of $\theta_{current}$.

The complexity of each searching procedure is at least

$$O\left(2^{w_N} + 2^{w_K} + \left[\frac{\pi}{4}\sqrt{\frac{2^k}{\widetilde{M}_g}}\right]\right)$$

quantum operations.

Note that the same remarks are true (as remarks 1 in section 2) regarding the presented search procedure. Quantum counting relatively to the boolean

function f_{K_1} for preparing $\sum_{K_1} \frac{1}{\sqrt{2^k}} |K_1\rangle \left|\widetilde{\theta}_{K_1}\right\rangle$ and it's inversion have to be performed at each Grover iteration. A general view of the correct quantum circuit that implements one Grover iteration at step 4 is shown in the figure 7.

Conditions	The complexity of the quantum linear cryptanalysis (number of quantum operations)
1.At step 4, quantum counting is performed only once (false).	$O(N + 2^{w_N} + 2^{w_K} + 2^{w_K} \cdot 1 + \left[\frac{\pi}{4}\sqrt{\frac{2^k}{\widetilde{M}_g}}\right])$
2.At step 4, quantum counting and it's inversion are performed at each Grover iteration (true).	$O(N + 2^{w_N} + 2^{w_K} + 2^{w_K+1} \cdot \left[\frac{\pi}{4}\sqrt{\frac{2^k}{\widetilde{M}_g}}\right])$

Table 5: Estimates of the complexity of the quantum linear cryptanalysis:
1) $O(N + 2^{w_N} + 2^{w_K} + 2^{w_K} \cdot 1 + \left[\frac{\pi}{4}\sqrt{\frac{2^k}{\widetilde{M}_g}}\right])$ - an understated estimates of the complexity, optimistic from the point of view of a quantum cryptanalyst.
2) $O(N + 2^{w_N} + 2^{w_K} + 2^{w_K+1} \cdot \left[\frac{\pi}{4}\sqrt{\frac{2^k}{\widetilde{M}_g}}\right])$ - the correct estimate.
3) Required at least $k + 2m + 1 + w_N + (w_K + 3)$ logical qubits, where $w_N = \lceil m/2 \rceil + 3$, $w_K = \lceil k/2 \rceil + 3$.

In case number of pairs $N < 2^m$, the scheme of applying the quantum linear cryptanalysis does not change, it is possible to reduce the complexity at the preparatory stage.

4 Conclusion

1. For a quantum linear cryptanalysis we need fewer logical qubits than to implement a quantum differential crypanalysis.

2. The complexity of the quantum differential and linear crypanalysis may turn out to be less than the complexity of key searching by the Grover's algorithm, if $E : V_n \times V_m \to V_m$ and $m < n/2$.

3. In case number of pairs $N < 2^m$, the schemes for applying the quantum differential and linear crypanalysis do not change, it is possible to reduce the complexity at the preparatory stage. The required number of logical qubits remains the same as in the case of $N = 2^m$.

4. Acceleration of computations due to "quantum parallelism" in the quantum differential and linear cryptanalysis, when we talk about key search, is apparently absent. Using quantum counting as a "subprogram" of the Grover algorithm eliminates quantum acceleration, since the main

part of computation complexity described by $O(\sqrt{K}) \cdot O(\sqrt{K}) \approx O(K)$ quantum operations.

References

[1] Huiqin Xie, Li Yang., "Using Bernstein-Vazirani algorithm to attack block ciphers", 2018, arXiv:1711.00853v3 [quant-ph].

[2] Hong-Wei Li, Li Yang, "A quantum algorithm to approximate the linear structures of Boolean functions", 2014, DOI:10.1017/S0960129516000013, https://arxiv.org/abs/1404.0611.

[3] Hong-Wei Li, Li Yang., "Quantum differential cryptanalysis to the block ciphers", 2015, https://arXiv.org/pdf/1511.08800.pdf.

[4] Kaplan M., Leurent G., Leverrier A., Naya-Plasencia M., "Quantum Differential and Linear Cryptanalysis", *IACR Transactions on Symmetric Cryptology*, 2016, 71-94, https://doi.org/10.13154/tosc.v2016.i1.71-94..

[5] Zhou Q., Lu S., Zhang A., Sun J., "Quantum differential cryptanalysis", 2019, https://arxiv.org/abs/1811.09931.

[6] Grover L.K., "Quantum mechanics helps in searching for a needle in a haystack", *Physical Review Letters*, **79**:2 (1997), 325.

[7] Brassard G., Hoyer P., Tapp A., "Quantum Counting", *Physical Review Letters*, 1998, http://arxiv.org/abs/quant-ph/9805082v1.

[8] Denisenko D.V., "Application of the Quantum counting to Estimation the Weights of Boolean Functions in Quipper", *ZhETF*, 2020, link.

[9] Durr C, Høyer P., "A quantum algorithm for finding the minimum", *Physical Review Letters*, 1996, arXiv.quant-ph/9607014.

[10] Nielsen M.A., Chuang I.L., *Quantum computation and quantum information*, Cambridge Univ. Press, 2010, http://csis.pace.edu/ctappert/cs837-18spring/QC-textbook.pdf.

[11] Denisenko D.V., "Quantum circuits for S-box implementation without ancilla qubits", *ZhETF*, **155**:6 (2019), 999, DOI:10.1134/S004445101906004X.

[12] Denisenko D.V., Nikitenkova M.V., "Optimization of S-boxes GOST R 34.12-2015 "Magma" quantum circuits without ancilla qubits", https://ctcrypt.ru/files/files/2019/materials/12_Denisenko.pdf, CTCrypt 2019..

[13] Bernstein E., Vazirani U., "Quantum complexity theory", *Proceedings of the 25th Annual ACM Symposium on theory of computing*, 1993.

[14] Giuliano Benenti, Giulio Casati and Giuliano Strini., "Principles of quantum computation and information", 2004, https://doi.org/10.1142/5528.

[15] Denisenko D.V., Nikitenkova M.V., "Application of Grover's Quantum Algorithm for SDES Key Searching", *ZhETF*, **128**:1 (2019), DOI:10.1134/S1063776118120142.

[16] Roetteler M., Steinwandt R., "A note on quantum related-key attacks", 2013, arXiv:1306.2301v2 [quant-ph].

[17] Steven A. Cuccaro, Thomas G. Draper, Samuel A. Kutin, David Petrie Moulton., "A new quantum ripple-carry addition circuit", 2004, https://arxiv.org/abs/quant-ph/0410184.

[18] Thomas G. Draper, Samuel A. Kutin, Eric M. Rains, Krysta M. Svore., "A logarithmic-depth quantum carry-lookahead adder", 2004, https://arxiv.org/abs/quant-ph/0406142.

[19] Kaye P., "Reversible addition circuit using one ancillary bit with application to quantum computing.", 2004, https://arxiv.org/abs/quant-ph/0408173.

Postquantum cryptography

Characteristics of Hadamard Square of Reed–Muller Subcodes of Special Type

Victoria Vysotskaya

JSC «NPK Kryptonite», Moscow, Russia
Lomonosov Moscow State University, Russia
vysotskaya.victory@gmail.com

Abstract

The existence of some structure in a code can lead to the decrease of security of the whole system built on it. Often subcodes are used to "disguise" the code as a "general-looking" one. However, the security of subcodes, whose Hadamard square is equal to the square of the base code, is reduced to the security of this code, i.e. this condition is undesirable. The paper finds the limiting conditions on the number of vectors of degree r removing of which retains this weakness for Reed–Muller subcodes and, accordingly, conditions for it to vanish. For $r = 2$ the exact structure of all resistant subcodes was found. For an arbitrary code $RM(r, m)$, the desired number was estimated from both sides. Finally, the ratio of subcodes, whose Hadamard square is not equal to the square of the original code, was proven to tend to zero if additional conditions on the codimension of the subcode and the parameter r are imposed and $m \to \infty$. Thus, the implementation of checks proposed in the paper helps to immediately filter out some insecure subcodes.

Keywords: post-quantum cryptography, code-based cryptography, Reed–Muller subcodes, Reed–Muller codes, Hadamard product, McEliece cryptosystem.

1 Introduction

The security of all standardized cryptographic algorithms used all around the world is based on the complexity of several number-theoretical problems. They usually are the discrete logarithm or factorization problem. However, in 1994 P. Shor showed [1] that quantum computers could break all schemes constructed this way. And in 2001 the Shore's algorithm was implemented on a 7-qubit quantum computer. Since then various companies have been actively developing more powerful quantum computers. Progress in this area poses a real threat to modern public-key cryptography.

There are several approaches to build post-quantum cryptographic schemes. One approach is to use error-correcting codes. No successful quantum-computer attacks on "hard" problems from this area are known.

Classical examples of code-based schemes are the McEliece cryptosystem [2] and the Niederreiter cryptosystem [3], that are equivalent in terms of security.

The interest in code-based schemes as post-quantum ones can be noticed while analyzing the works submitted to the contest for prospective public-key post-quantum algorithms which was announced in 2016 by the US National Institute of Standards and Technology (NIST) [4]. The algorithms that win this contest will be accepted as US national standards. 21 of 69 applications filed (that is, almost a third of all works) were based on coding theory. Despite the fact that some of them were attacked, it seems that this approach looks quite promising and deserves further study and development. This interest is also traced in Russian cryptography. Code-based schemes were chosen by the Technical Committee for Standardization "Cryptographic and Security Mechanisms" (TC 26) as one of directions in developing draft Russian national standards of post-quantum cryptographic algorithms.

When one is facing the challenge to synthesize a new code-based scheme, the first thing to think about is the choice of basic code. Some schemes do not specify the code, thus leaving it to the discretion of the user. Such schemes are usually more reliable since their security is often directly reduced to NP-complete problems. Most often, these problems are decoding and syndrome decoding. However, choosing a special code also has some advantages. For example, such codes provide asymmetric complexity in solving the decoding problem for the legal user and adversary. In addition, due to the structure of the code, the sizes of the public keys can be significantly reduced.

However, the structure can also cause a significant decrease in security of the code, therefore one of the most important tasks is to "disguise" the code as a "general-looking" one. One solution is to use subcodes. This approach allows to "destroy" the structure of the code, retaining the ability to work with the result in mostly the same way as with the original one. Nevertheless, it is worth considering that many of proposed systems based on subcodes turned out to be vulnerable. So in [5] and [6] C. Wieschebrink built efficient attacks on some special cases of the Berger–Loidreauo cryptosystem [7], that is based on subcodes of the Reed–Solomon code. The McEliece cryptosystem based on subcodes of algebraic geometry codes was attacked in [8]. The digital signature based on modified Reed–Muller codes and described in [9] was also attacked during the peer review at the NIST contest.

One of the mechanisms for analyzing codes with a hidden structure is the use of the technique of Hadamard product of two codes. This method was used by M. Borodin and I. Chizhov in [10] to improve Minder–Shokrollahi attack [11] on the McEliece cryptosystem based on Reed–Muller codes. In

another work [12] this technique allowed Chizhov and Borodin to reduce the security of the cryptosystem on subcodes of Reed–Muller codes of codimension one to the security of the scheme on full codes. The paper [13] describes the distinguisher between random codes and Reed–Solomon codes using Hadamard product.

In our paper the mentioned technique will be used to analyze Reed–Muller subcodes in standard basis without restriction on codimension. The main question that we will try to answer is: which Reed–Muller subcodes do not allow Chizhov–Borodin's approach. Since the reduction can be performed to a subcode, which Hadamard square coincides with the square of the original code, we will look for conditions under which this equality ceases to hold. Codes obtaining these conditions will be called *unstable codes*, the others – *stable codes*. In addition we will try to compute the probability that a randomly chosen Reed–Muller subcode is unstable.

In Section 2 the exact structure of all stable subcodes of $RM(2, m)$ is found. Thus, to provide the security it is necessary to choose at least another subcode. To be sure that a subcode of $RM(2, m)$ is unstable it is sufficient to exclude $m+1$ monomials of degree 2 from it's standard basis. For an arbitrary Reed–Muller code $RM(r, m)$ in Section 3 we estimate (both from the above and below) the number of vectors of degree r that must be excluded from the basis of the code in order to distort its square. Finally, in Section 4 we show that the ratio of unstable subcodes tends to zero (as $m \to \infty$) given some additional conditions on the codimension of the subcode and the parameter r. Thus, it is not enough to choose an arbitrary Reed–Muller subcode when synthesizing a real scheme. It is necessary to check the property formulated below as Proposition 4. At the same time subcodes satisfying this property require additional consideration since they may have some special structure.

2 The structure of stable $RM(2, m)$ subcodes

Recall that *Reed–Muller code* $RM(r, m)$ is the set of all Boolean functions f of m variables such that $\deg(f) \leqslant r$. Consider the code $RM(1, m)$. We look for the minimum number of monomials $f_1, \ldots, f_w$ of degree 2 such that the code

$$\operatorname{span}\big((RM(1, m) \cup \{f_1, \ldots, f_w\}) \big)^2 = RM(4, m).$$

Here the squaring operation refers to the squaring of Hadamard. *Hadamard product* of two vectors is a vector obtained as a result of component-wise

product of coordinates of these vectors:

$$(a_1, \ldots, a_n) \circ (b_1, \ldots, b_n) = (a_1 b_1, \ldots, a_n b_n),$$

and Hadamard product of two codes A and B is the span of all pairwise products of form $a \circ b$, where $a \in A, b \in B$.

We will consider Reed–Muller codes spanned by their standard basis. *The standard basis of the Reed–Muller code* $RM(r, m)$ includes all monomials of m variables of degree from 0 to r inclusively, i.e.

$$1, x_1, x_2, \ldots, x_m, x_1x_2, \ldots, x_{m-1}x_m, \ldots, x_1 \cdots x_r, \ldots, x_{m-r-1} \cdots x_m.$$

So we look for minimum number of monomials $f_1, \ldots, f_w$ of degree 2 such that the code

$$\operatorname{span}\big(RM(1, m) \cup \{f_1, \ldots, f_w\}\big) \tag{1}$$

is stable. Obviously, after finding this number, one can also answer another question: what is the maximum number of monomials of degree 2 that can be removed from the basis of the code $RM(2, m)$ so that the code

$$\operatorname{span}\big(RM(2, m) \setminus \{g_1, \ldots, g_q\}\big) \tag{2}$$

is still stable. And so, after removing $(q+1)$ basis vectors, one gets an unstable code.

Now let us proceed to the graph interpretation of this problem. We match a subcode $\mathcal{A} \subset RM(2, m)$ with a graph G with m vertices labeled $x_1, \ldots, x_m$. An edge $\{x_i, x_j\}$ is present if and only if monomial $x_i x_j \in \mathcal{A}$.

We will say that a graph with m vertices *satisfies the property* P if

1. the degree $\deg(v)$ of any vertex v is not less than $(m - 3)$;
2. if $\deg(v) = m - 3$ and edges $\{v, u\}$ and $\{v, w\}$ are missing, then the edge $\{u, w\}$ is present.

The case $\deg(x_1) = m - 3$ is shown in Fig.1, where lines denote present edges.

Theorem 1. *Subcode of the form (1) is stable if and only if the property* P *is satisfied for the corresponding graph.*

Proof. Denote $G = (V, E)$ the graph corresponding to the subcode of form (1). Note that the condition

$$\operatorname{span}\big(\,(RM(1, m) \cup \{f_1, \ldots, f_\ell\})\,\big)^2 = \operatorname{span}\big(\{f'_1, \ldots, f'_{w'}\}\big), \tag{3}$$

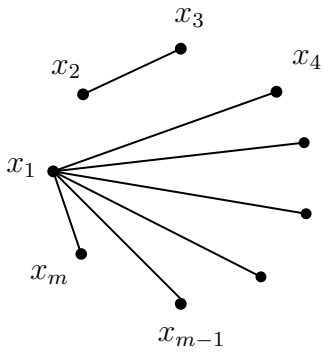


Figure 1:

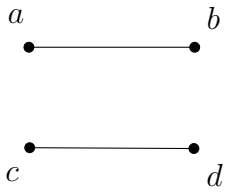


Figure 2: Graph H

where f_i' are monomials of degree 4, is equivalent to the condition that any induced subgraph of G with 4 vertices has a subgraph isomorphic to the graph H shown in Fig.2. The edges $\{a, b\}$ and $\{c, d\}$ correspond to degree-2 monomial used to produce the monomial $abcd$. Also note that from (3) it follows that subcode (1) is stable if we can obtain any monomial of degree 4 as product of some f_i and f_j, then any monomial of degree 3 can be obtained as product of some f_i and some x_j,

Now we can prove the necessity. Fix any vertex v. If any 3 incident edges $\{v, u_j\}$ for $j = 1, 2, 3$ are missing, then the induced subgraph on vertices v, u_1, u_2, u_3 would not have the necessary subgraph. The contradiction proves that $\deg(v) \geqslant m - 3$.

If, however, $\deg(v) = m - 3$ and edges $\{u, v_1\}$ and $\{u, v_2\}$ are missing, the edge $\{v_1, v_2\}$ must be present, as otherwise none of the induced 4-vertex subgraphs containing vertices $\{u, v_1, v_2\}$ will have the necessary subgraph. Thus, the property P is satisfied.

Now to the proof of sufficiency. Fix any induced subgraph with 4 vertices. Note that it satisfies the property P for $m = 4$. If any vertex v has degree 1, i.e. the edge $\{v, w\}$ is present, but $\{v, u_1\}$ and $\{v, u_2\}$ are not, then by P the edge $\{u_1, u_2\}$ must be present. Thus, we have edges $\{v, w\}$ and $\{u_1, u_2\}$ necessary for the H-isomorphic subgraph.

If all 4 vertices have degree at least 2, then we can find a simple cycle in our graph. Obviously, its length is either 3 or 4. If it is 4, the presence of H-isomorphic subgraph is obvious. Otherwise, we have a triangle u, v, w and, moreover, the fourth vertex q has degree at least 2. Assume (without loss of generality) the edge $\{q, u\}$ is present, then for H-isomorphic subgraph we can take the edges $\{q, u\}$ and $\{v, w\}$. □

From Theorem 1 the minimum number of edges is obtained in case if the condition P is true for the graph and the degree of each vertex is $(m - 3)$.

It remains to describe such graphs.

Proposition 1. *If the condition P is satisfied for some graph G and the degree of each vertex is $(m-3)$, then the complementary graph $\overline{G}$ is union of cycles of length at least 4.*

Proof. Graph $\overline{G}$ is triangle-free and all its vertices have degree 2. Choose an arbitrary vertex u_1. It is not isolated, therefore, one can select a vertex adjacent to it, call it u_2, As $\deg(u_2) = 2$, there exists some adjacent vertex $u_3 \neq u_1$. Continue in this way until u_j coincides with one of $u_1, \ldots, u_{j-1}$. Note that u_j cannot coincide with u_i for $i > 1$ as it would mean that $\deg(u_i) \geqslant 3$. Thus $u_1, \ldots, u_{j-1}$ form a simple cycle. Its length is at least 4, as $\overline{G}$ is triangle-free. □

Thus, we have described the structure of the graph corresponding to the minimal stable subcode of form (1). Now let us describe the complete structure of such codes. Let us call a *bamboo graph* a tree which either has one vertex or has two vertices of degree 1 and every other vertices of degree 2.

Proposition 2. *If the condition P is satisfied for some graph G, then the complementary graph $\overline{G}$ is a union of cycles of length at least 4 and bamboo graphs.*

Proof. We proceed as in Proposition 1 and try to find a cycle in $\overline{G}$. But we can stop in a vertex of degree 1, thus obtaining a bamboo graph. Isolated vertices are bamboo graphs by definition. □

Corollary 1. *Assume that $m \geqslant 4$. Then minimum number of monomials of degree 2 needed to get a stable subcode of form (1) is $m(m-3)/2$; maximum number of monomials of degree 2 such that the code of form (2) is stable is m.*

Proof. As follows from Theorem 1, we need to consider the subcodes corresponding to graphs satisfying property P. From Proposition 2 it follows that $\overline{G}$ has no more than m edges (this bound is exact for graph consisting of cycles). Thus G has at least $C_m^2 - m = m(m-3)/2$ edges. Moreover, it means that removing not more than m edges we remain in the stable code. □

Note that removing $m+1$ or more monomials of degree 2 from the code $RM(2, m)$ we get an unstable code.

3 Lower and upper bounds for minimal stable $RM(r, m)$ subcodes sizes

In this section we try to carry out argument for $r > 2$. That is, we will look for the minimum number w, such that the code

$$\operatorname{span}\big(RM(r-1,m) \cup \{f_1, \ldots, f_w\}\big) \tag{4}$$

is stable. Here f_i is a monomial of degree r. We match a subcode $\mathcal{A} \subset RM(r, m)$ with a hypergraph G with m vertices labeled $x_1, \ldots, x_m$. An r-edge $\{x_{i_1}, \ldots, x_{i_r}\}$ is present if and only if monomial $x_{i_1} \ldots x_{i_r} \in \mathcal{A}$. In the general case the condition similar to having an H-isomorphic subgraph in each 4-vertex induced subgraph is equivalent to condition of the code (4) being stable. Namely, each set of $2r$ vertices must be covered by two disjoint r-edges. Let us denote a graph satisfying this condition by *stable graph*. Note about covering monomials of lower degrees is the same as in the case of $r = 2$.

To find the minimum number of monomials of degree r to remove for obtaining an unstable code can be computed by subtracting from the total number of r-edges $w + 1$ one. Therefore, we will not dwell on this issue separately.

In what follows we will use terms "graph" and "hypergraph" interchangeably. Denote $w(r, m)$ the mininal number of degree-r monomials needed to make subcode (4) stable, or, alternatively, minimal number of edges in a stable r-hypergraph with m vertices.

Proposition 3. *For any natural r and $m \geqslant 2r$*

$$w(m, r) \geqslant C_m^{2r} / C_{m-r}^r.$$

Proof. Note that any set of $2r$ vertices in a stable graph contains at least one edge. Moreover, any edge is contained in exactly C_{m-r}^r such sets. Thus total number of edges multiplied by C_{m-r}^r is at least number of all sets of $2r$ vertices, which is C_m^{2r}. This gives the necessary bound. □

Corollary 2. *Any stable graph contains at least $1/C_{2r}^r$ edges of a complete graph.*

Proof. The total possible number of r-edges in a graph with m vertices is C_m^r. Then

$$\frac{C_m^{2r}}{C_{m-r}^r \cdot C_m^r} = \frac{(r!)^2}{(2r)!} = \frac{1}{C_{2r}^r}.$$

□

Let all the vertices of the graph be divided into sets S_i, $i = 1, \ldots, t$ of size $2r$, intersecting each other in some way. Let the size of maximum pairwise intersection be h. Let us denote $\mathcal{S} = \{S_i\}_{i=1}^{t}$.

Lemma 1. *If $h < r/3$, then for any set $Q \notin \mathcal{S}, |Q| = 2r$ there are at most two sets from $\mathcal{S}$ such that their intersection with Q have size at least r.*

Proof. Assume that Q intersects with at least 3 sets such that intersection size is at least r. Without loss of generality we assume that the sets are S_1, S_2 and S_3. Let us denote $Q \cap S_1 = A_1, Q \cap S_2 = A_2, Q \cap S_3 = A_3$. Since $|Q| = 2r$, then it is obvious that $|A_1 \cup A_2 \cup A_2| \leqslant 2r$. On the other hand, according to the inclusion-exclusion formula,

$$|A_1 \cup A_2 \cup A_2| \geqslant |A_1| + |A_2| + |A_3| - |A_1 \cap A_2| - |A_1 \cap A_3| - |A_2 \cap A_3|.$$

Then

$$\sum_{i=1}^{3} |A_i| \leqslant 2r + 3h.$$

By condition $|A_i| \geqslant r,\ i \in \{1, 2, 3\}$, therefore

$$\sum_{i=1}^{3} |A_i| \geqslant 3r.$$

Whence $3r \leqslant 2r + 3h$ and $h \geqslant r/3$, which contradicts the hypothesis of the theorem. □

Let us find the maximum possible number of edges that can be removed from the complete graph using the above arguments, such that the graph remains stable.

Theorem 2. *For any natural $r \geqslant 2$, $m \geqslant 2r$ and $h < r/3$*

$$w(m, r) \leqslant C_m^r - T(r, m, h) \cdot (C_{2r}^r - 2)\,,$$

where

$$T(r, m, h) = \max\big\{t : \exists S_1, \ldots, S_t \left(S_i \subset \{1, \ldots, m\}\ \&\right.$$
$$\left.\&\, |S_i| = 2r\ \&\ (i \neq j \Rightarrow |S_i \cap S_j| \leqslant h),\ i, j \in \{1, \ldots, t\}\right)\big\}.$$

Proof. Note that two disjoint r-edges are sufficient to cover a set of $2r$ vertices. Thus, it is possible to remove $\delta = (C_{2r}^r - 2)$ r-edges from the complete graph on the $2r$ vertices and preserve the stability of it. Obviously, no more edges can be removed.

Suppose that δ edges are removed from each set from $\mathcal{S}$ so that all of them are covered by at least two r-edges. It remains to verify that there exists a similar cover for *any* set of $2r$ vertices. Since by construction we can certainly cover any set S_i, we will prove that we can also cover any set $Q \notin S$, $|Q| = 2r$.

Note that if the cardinality of the intersection with some S_i does not exceed $(r-1)$, then removing edges in it does not affect the number of edges in Q. At the same time, according to Lemma 1, for $h < r/3$ any set of size $2r$ can have intersection of size at least r with no more than two sets from S. If there is only one such set, say, S_1, then we have two cases:

1. $|Q \cap S_1| = 2r - 1$. In this case there exist some edge $e_1 \in (Q \cap S_1)$ not containing vertex v, $\{v\} = S_1 \setminus Q$ (as S_1 must be covered by two disjoint edges). Thus, we can take $e_2 = Q \setminus e_1$ (it must be present as we have removed only edges contained inside sets S_i), and $\{e_1, e_2\}$ form the disjoint cover of Q.

2. $|Q \cup S_1| < 2r - 1$. In this case there are at least two vertices v_1 and v_2 inside $Q \setminus S_1$ and the cover can be formed using any two disjoint edges $e_1, e_2 \subset Q$ such that $v_1 \in e_1$, $v_2 \in e_2$.

Now consider the case when there are exactly two sets S_1 and S_2 intersecting with Q at no less than r vertices. Assume that $|A_1| > r + h$. Then, according to the inclusion-exclusion formula $|A_1 \cap A_2| = |A_1| + |A_2| - |A_1 \cup A_2| > r + h + r - 2r = h$ that contradicts with $|S_1 \cap S_2| \leqslant h$. Thus $r \leqslant |A_i| \leqslant r + h,\ i \in \{1, 2\}$. So there are at most $2 \cdot C^r_{r+h}$ edges removed from Q. Note that

$$\frac{C^r_{2r}}{2 \cdot C^r_{r+h}} = \frac{(2r)!\, r!\, h!}{2r!\, r!\, (r+h)!} = \left(\frac{2r}{r+h}\right)\left(\frac{2r-1}{r+h-1}\right)\cdots\left(\frac{r+1}{2(h+1)}\right). \quad (5)$$

The last multiplier is greater than 1 for $r > 3$. For others holds

$$\frac{2r-i}{r+h-i} < \frac{2r}{r+h} < \frac{6}{4}.$$

Besides for $r > 2$ there at least 3 multipliers in product (5). So for $r > 3$ we can claim

$$\frac{C^r_{2r}}{2 \cdot C^r_{r+h}} > 2.$$

For $r = 2$ and $r = 3$ the inequality can be verified directly.

There are $C^r_{2r}/2$ pairs of disjoint edges inside Q, so there remains at least one such pair after removal of $2 \cdot C^r_{r+h} < C^r_{2r}/2$ edges from Q.

So we obtained a stable graph removing δ edges from a complete graph for each set from $\mathcal{S}$. It remains to note that $|\mathcal{S}|$ is the number of sets of size $2r$ whose intersections are not larger than h. □

Remark 1. *In [14] P. Erdös and J. Spencer introduce the value* $\mathrm{m}(n,k,t)$. *It determines the size of the largest set of* k*-element subsets of* $\{1,\dots,n\}$ *such that any two members of this set intersect in less than* t *elements. Later V. Rödl [15] proves that*

$$\lim_{n\to\infty} \mathrm{m}(n,k,t) = \frac{C_n^t}{C_k^t}.$$

That is, in our case,

$$\lim_{m\to\infty} T(r,m,h) = \lim_{m\to\infty} \mathrm{m}(m,2r,\lfloor r/3\rfloor) = \frac{C_m^{\lfloor r/3\rfloor}}{C_{2r}^{\lfloor r/3\rfloor}}.$$

4 The ratio of unstable $RM(r,m)$ subcodes

We consider subcodes of the standard basis of the Reed –Muller code in which ℓ vectors are missing. This number is called the *codimension* of the subcode. Let us denote the set of subcodes of codimension ℓ by $RM^{\ell}(r,m)$.

For the given parameter s and the set $I = \{i_j\}_{j=1}^{s}$ we will call unordered pairs $\{A,B\}$ *critical partition* if:

$$A \cap B = \varnothing,$$
$$A \cup B = I,$$
$$1 \leqslant |A|, |B| \leqslant r.$$

Then it is impossible to obtain the monomial $x_{i_1}\dots x_{i_s}$ after squaring a subcode if and only if at least one element of each critical partition is removed. This follows from the fact that if this monomial is present in the square of the code, it should be formed of a pair $\{A,B\}$ from the appropriate critical partition. But by the hypothesis either A or B is absent.

Obviously, the following proposition is true.

Proposition 4. *A code is unstable* $RM(r,m)$ *subcode if and only if at least one element from each critical partition for some monomial* $x_{i_1}\dots x_{i_s}$ *is removed.*

Proposition 5. *For the given parameter* s *and the set* I *the number of critical partitions is*

$$w(s) = \sum_{p=\max\{s-r,1\}}^{\min\{r,s-1\}} \frac{1}{2}\, C_s^p.$$

Proof. On the one hand the sizes of the subsets must not exceed r. On the other hand the partition must be non-trivial, that is, partitioning into an empty set and a set, coinciding with I, is unacceptable. Finally, when considering all partitions, each pair is counted twice. □

Let us order in some way (say, lexicographically) the elements of each critical partition and then the critical partitions themselves. Now we consider any set M consisting of elements of critical partitions and having the property that for every critical partition M contains at least one element of this partition. We can encode M with a string $\alpha \in \{1,2,3\}^{w(s)}$, where

$$\alpha_j = \begin{cases} 1 & \Leftrightarrow \text{ the 1st element of the } j\text{-th pair lies in } M, \\ 2 & \Leftrightarrow \text{ the 2nd element of the } j\text{-th pair lies in } M, \\ 3 & \Leftrightarrow \text{ both elements of the } j\text{-th pair lie in } M; \end{cases}$$

We will also write $M(\alpha)$ to denote the set corresponding to a given $\alpha \in \{1,2,3\}^{w(s)}$. It can be easily seen that

$$|M(\alpha)| = \#_\alpha(1) + \#_\alpha(2) + 2 \cdot \#_\alpha(3),$$

where $\#_\alpha(c)$ is the number of symbols c in the string α.

Let us denote $k = \sum_{p=0}^{r} C_m^p$ the dimension of the original code (or the number of vectors in its standard basis). There are exactly two kinds of unstable subcodes: those containing monomial 1 and those not containing it. There are obviously $C_{k-1}^{\ell-1}$ subcodes of the second kind.

Now we fix s, an index set I of size s and a string $\alpha \in \{1,2,3\}^{w(s)}$. Among the subcodes of the first type there are

$$C_{k-1-2w(s)}^{\ell-|M(\alpha)|}$$

ones that satisfy the condition: among the monomials comprising critical partitions for I exactly monomials from $M(\alpha)$ are absent. The reason is that we need to choose $\ell - |M(\alpha)|$ monomials from all monomials of degree more than 0 that do not comprise any critical partition (there are $k-1-2w(s)$ of them).

For a given s there are C_m^s variants of choosing index set I. But some codes may be counted several times. So we can consider the following theorem proved.

Theorem 3. *The number of unstable $RM(r,m)$ subcodes is*

$$\theta \leqslant \sum_{s=2}^{2r} C_m^s \cdot \sum_{\alpha \in \{1,2,3\}^{w(s)}} C_{k-1-2w(s)}^{\ell-|M(\alpha)|} + C_{k-1}^{\ell-1}.$$

Theorem 4. *If $\ell = const$ and $r \geqslant 2\ell + 1$, then the ratio of unstable $RM(r, m)$ subcodes tends to zero as $m \to \infty$.*

Proof. Our goal is the asymptotic estimate of the probability of the event that after removing ℓ vectors from the standard basis of the code $RM(r, m)$, the square of the resulting code will differ from $RM(2r, m)$. The upper bound for it is θ/C_k^ℓ. We divide this bound into two parts and show the tendency to zero for each of them independently. For one of them it follows immediately from the fact that

$$\frac{C_{k-1}^{\ell-1}}{C_k^\ell} = \frac{\ell}{k} \xrightarrow[m\to\infty]{} 0,$$

since $k \to \infty$ as $m \to \infty$.

Now we consider the first part and denote it's numerator by γ. Notice that

$$\#_\alpha(1) + \#_\alpha(2) + 2 \cdot \#_\alpha(3) = |M(\alpha)| \geqslant w(s) = \#_\alpha(1) + \#_\alpha(2) + \#_\alpha(3).$$

Then the number of removed vectors that are elements of critical partitions for s is $|M(\alpha)| \geqslant w(s)$ and the total number of removed vectors is ℓ. That is, $w(s) \leqslant \ell$ and we can consider only parameters s satisfying this condition. Then

$$2w(s) = \sum_{p=\max\{s-r,1\}}^{\min\{r,s-1\}} C_s^p \leqslant 2\ell. \tag{6}$$

We consider separately two cases. If $s \geqslant r+1$, we have $\min\{r, s-1\} = r$ and in the sum (6) there is the element C_s^r. Thus

$$2\ell \geqslant 2w(s) \geqslant C_s^r \geqslant s.$$

The last inequality follows from the fact that

$$C_s^r = \frac{s^{\underline{r}}}{r!} = \frac{(r+1)}{2} \cdot \frac{(r+2)}{3} \cdot \ldots \frac{(s-1)}{r} \cdot \frac{s}{1}.$$

If, on the other hand, $s < r+1$, we have $\max\{s-r, 1\} = 1$ and there is the element C_s^1 in the sum (6). Hence

$$2\ell \geqslant 2w(s) \geqslant C_s^1 = s.$$

So either way the inequality $s \leqslant 2\ell$ is satisfied.

We simplify the upper bound for γ using this inequality and the monotonicity of the binomial coefficient C_n^k with respect to the parameter k, which

guarantees the increase of the value C_n^k with the increase of k:

$$\sum_{s=2}^{2r} C_m^s \cdot \sum_{\alpha\in\{1,2,3\}^{w(s)}} C_{k-1-2w(s)}^{\ell-|M(\alpha)|} \leqslant \sum_{s=2}^{2\ell} C_m^{2\ell} \cdot \sum_{\alpha\in\{1,2,3\}^{w(s)}} C_{k-1-2w(s)}^{\ell-|M(\alpha)|} \leqslant$$
$$\leqslant 2\ell \cdot C_m^{2\ell} \max_{s\in[2,2\ell]} \left\{ C_{k-1-2w(s)}^{\ell-z} \cdot 3^{w(s)} \right\},$$

where $z = \min_{\alpha\in\{1,2,3\}^{w(s)}} \{|M(\alpha)|\}$.

Note that $\ell = const$ and $3^{w(s)} \leqslant const$, since $s \leqslant 2\ell$, and $w(s) < 2^s$. The last is true by virtue of

$$2^s = (1+1)^s = \sum_{p=0}^{s} C_s^k > \frac{1}{2} \sum_{p=\max\{s-r,1\}}^{\min\{r,s-1\}} C_s^p = w(s).$$

These considerations, as well as the monotonicity of the binomial coefficient C_n^k with respect to n and the inequality $|M(\alpha)| \geqslant w(s)$, allow us to obtain the upper bound

$$const \cdot C_m^{2\ell} \cdot C_k^{\ell-w(s)} \leqslant const \cdot C_m^{2\ell} \cdot C_k^{\ell-1} := \psi.$$

We proceed to the ratio estimation.

$$\frac{\gamma}{C_k^\ell} \leqslant \frac{\psi}{C_k^\ell} = \frac{const \cdot C_m^{2\ell} \cdot C_k^{\ell-1}}{C_k^\ell} = \frac{const \cdot C_m^{2\ell} \cdot \ell}{k-\ell+1} = \frac{const \cdot C_m^{2\ell}}{k-\ell+1} \leqslant const \cdot \frac{m^{2\ell}}{2k}.$$

After tending m to infinity we can claim that such $p = 2\ell+1$ exists, that is, summand $C_m^p \geqslant m^p$ is an element of the sum representation of k. Then

$$const \cdot \frac{m^{2\ell}}{2k} \leqslant const \cdot \frac{m^{2\ell}}{m^{2\ell+1}} = const \cdot \frac{1}{m} \xrightarrow[m\to\infty]{} 0.$$

□

Future research

More accurate estimates on the minimal stable code sizes for general case are still required, as are better estimates of the ratio of stable subcodes. In addition, an idea for future research could be to find an analogues of the obtained results for an arbitrary basis of the Reed–Muller code.

References

[1] Shor P. V., "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer", *SIAM Journal on Computing*, **26**:5 (1997), 1484–1509.

[2] McEliece R. J., "A public-key cryptosystem based on algebraic coding theory", *DSN Progress Report*, **4244**, 1978, 114–116.

[3] Niederreiter H., "Knapsack-type cryptosystems and algebraic coding theory", *Problems of Control and Information Theory*, **15**:2 (1986), 159–166.

[4] https://csrc.nist.gov/Projects/post-quantum-cryptography/Post-Quantum-Cryptography-Standardization/Call-for-Proposals.

[5] Wieschebrink C., "An Attack on a Modified Niederreiter Encryption Scheme", *Public Key Cryptography - PKC 2006*, PKC 2006, Lecture Notes in Computer Science, **3958**, eds. Yung M., Dodis Y., Kiayias A., Malkin T., Springer, Berlin, Heidelberg, 2006, 14–26.

[6] Wieschebrink C., "Cryptanalysis of the Niederreiter Public Key Scheme Based on GRS Subcodes", *Post-Quantum Cryptography*, PQCrypto 2010, Lecture Notes in Computer Science, **6061**, eds. Sendrier N., Springer, Berlin, Heidelberg, 2010, 61–72.

[7] Berger T. P., Loidreau P., "How to mask the structure of codes", *Designs, Codes and Cryptography*, **35**:1 (2005), 63–79.

[8] Couvreur A. , Marquez-Corbella I., Pellikaan R., "Cryptanalysis of public-key cryptosystems that use subcodes of algebraic geometry codes", *Coding Theory and Applications*, CIM Series in Mathematical Sciences, **3**, eds. Pinto R., Rocha Malonek P., Vettori P., Springer, Cham, 2015, 133–140.

[9] Lee W., Kim Y.-S., Lee Y.-W., No J.-S., *Post quantum signature scheme based on modified Reed-Muller code pqsigRM*, 2017, 35 pp., NIST proposal.

[10] Borodin M. A., Chizhov I. V., "Effective attack on the McEliece cryptosystem based on Reed-Muller codes", *Discrete Mathematics and Applications*, **24**:5, 273–280.

[11] Minder L., Shokrollahi A., "Cryptanalysis of the Sidelnikov Cryptosystem", *Advances in Cryptology - EUROCRYPT 2007*, EUROCRYPT 2007, Lecture Notes in Computer Science, **4515**, eds. Naor M., Springer, Berlin, Heidelberg, 2007, 347–360.

[12] Chizhov I. V., Borodin M. A., "Hadamard products classification of subcodes of Reed–Muller codes codimension 1", *Discrete Mathematics and Applications*, **32**:1 (2020), 115–134.

[13] Couvreur A., Gaborit P., Gauthier-Umãna V., Otmani A., Tillich J.-P., "Distinguisher-based attacks on public-key cryptosystems using Reed–Solomon codes", *Designs, Codes and Cryptography*, **73**:2 (2014), 641–666.

[14] Erdös P., Spencer J., *Probabilistic Methods in Combinatorics*, Akadèmiai Kiadó, Budapest, 1974, 106 pp.

[15] Rödl V., "On a Packing and Covering Problem", *European Journal of Combinatorics*, **6**:1 (1985), 69–78

IND-CCA2 secure McEliece-type modification in the standard model

Yury Kosolapov and Oleg Turchenko

Southern federal university, Russia
itaim@mail.ru, olegmmcs@gmail.com

Abstract

The main goal of this work is to construct a McEliece-type cryptosystem with IND-CCA2 property in the standard model and an effective data transfer rate. The proposed modification is based on the application of the s-repetition method and uses one common secret permutation. The modification requires to transmit s encrypted blocks for $s/2$ information messages, that makes this modification more effective than most other modifications based on the s-repetition method. The paper also provides additional cryptosystems with the semantic security.

Keywords: McEliece-type cryptosystem, s-repetition method, IND-CCA2-security, IND-CPA-security, standard model

1 Introduction

Active research in the field of code-based asymmetric cryptosystems is related to their possible applications in post-quantum era. The complexity of such cryptosystems is based on the problem of decoding a general linear code that makes them immune to known attacks on quantum computers. The first code-based cryptosystem built on the basis of the Goppa code was proposed by R. McEliece [1]. Replacing the Goppa code with some other codes leads to a weakening of the system. In particular, for generalized Reed-Solomon codes (GRS-codes), Reed-Muller binary codes (RM-codes), direct sums of GRS-codes, direct sums of binary RM-codes, effective algorithms for finding suitable private keys by public keys are found [2]–[4]. Moreover, changing the method of constructing a public key can weaken the system. For instance, for the cryptosystem V.M. Sidelnikov [5] and its generalization [6] in [7] a way was found to create a suitable private key. Note that for the original McEliece system on Goppa code, polynomial key-recovery attacks (structural attacks) are not currently known. At the same time, the original McEliece system, regardless of the code used, is vulnerable to attacks on ciphertexts. One of the most successful attacks is attack based on decoding

by information sets [8]. In [9] a modification of the McEliece cryptosystem was proposed that provides protection against chosen plaintext attack. However, this modification remains vulnerable to the strongest class of attacks, the so-called adaptive chosen ciphertext attacks.

In [10], authors constructed modifications that provide protection against structural attacks and, at the same time, are immune to adaptive chosen ciphertext attacks. However, the proposed design has a low data transfer rate (the ratio of the length of the message to the length of the encrypted text). In this paper, on the basis of the construction from [10], the task is to construct a system with security against adaptive chosen ciphertext and an effective data transfer rate.

In addition to the introduction, the paper contains three sections. The first section introduces a definition of an asymmetric cryptosystem and the necessary concepts of attack models. Original McEliece cryptosystem [1] and the randomized McEliece cryptosystem [9] are also defined in this section. The second section is devoted to the construction of new cryptosystems. Security of the constructed cryptosystems is considered in the third section. The fourth section considers security parameters of constructed cryptosystem and comparison with other IND-CCA2 schemes.

2 Preliminaries

2.1 Security notions

Let n, t be natural, $2t < n$, $[n] = \{1, ..., n\}$, $\beta \subseteq [n]$, $2^{[n]}$ is set of all subsets of $[n]$, $\mathbb{F}_q$ be a Galois field of cardinality q, where q is the degree of a prime number. The support of the vector $\mathbf{m} = (m_1, ..., m_n)(\in \mathbb{F}_q^n)$ is the set $\mathrm{supp}(\mathbf{m}) = \{i : m_i \neq 0\}$ and the Hamming weight of this vector is a number $\mathrm{wt}(\mathbf{m}) = |\mathrm{supp}(\mathbf{m})|$. A function $\gamma : \mathbb{N} \to [0, 1]$ is negligible of k, if

$$\forall c \in \mathbb{N}\ \exists k_c \in \mathbb{N}\ \forall k > k_c : \gamma(k) \leqslant k^{-c}.$$

We will use the notation of the algorithms similarly to the [11]. Notation $y \leftarrow \mathcal{A}(x_1, x_2, ...)$ means that the algorithm $\mathcal{A}$ runs with input parameters $x_1, x_2, ...$ and outputs value y. If the algorithm $\mathcal{A}$ has access to the output of the algorithm (oracle) $\mathcal{O}$ then we write $y \leftarrow \mathcal{A}^{\mathcal{O}}(x_1, x_2, ...)$. Notation $\mathcal{A}^{\varnothing}(x_1, x_2, ...)$ means that $\mathcal{A}$ does not have access to the output of any oracle. If S is a finite set, then $s \in_R S$ denotes the operation of picking an element at random and uniformly from S. Denote by $\mathcal{E}_{n,t,\beta}$ the subset of $\mathbb{F}_2^n$ such that any vector $\mathbf{e} = (e_1, ..., e_n)(\in \mathcal{E}_{n,t,\beta})$ has Hamming weight t and $e_i = 0$ for any

$i \in \beta$. We will write $\mathcal{E}_{n,t}$ when $\beta = \varnothing$. To consider security notions of public key cryptosystems it is convenient to define a cryptosystem as a triplet of algorithms i.e. $\Sigma = (\mathcal{K}, \mathcal{E}, \mathcal{D})$, where:

1) $\mathcal{K}$ is a probabilistic polynomial-time key generation algorithm which takes as input a security parameter $N \in \mathbb{N}$ and outputs a pair of public-key and a secret-key (pk, sk),

2) $\mathcal{E}$ is probabilistic polynomial-time encryption algorithm which takes as input a public-key pk and a message $\mathbf{m}$, and outputs a ciphertext $\mathbf{c}$; we will write $\{\mathbf{m}\}^{\Sigma}_{pk}$ as encryption of the message $\mathbf{m}$ with the key pk,

3) $\mathcal{D}$ is deterministic polynomial-time decryption algorithm which takes as input a secret-key sk and a ciphertext $\mathbf{c}$, and outputs either a message $\mathbf{m}$ or a symbol $\perp$ in the case, when the ciphertext is incorrect; decryption of the ciphertext $\mathbf{c}$ on the secret key sk we will denote $\{\mathbf{c}\}^{\Sigma}_{sk}$.

Now we will define the security of $\Sigma = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ under chosen plaintext attack (CPA) and under adaptive chosen ciphertext attack (CCA2) in the same way as in [11]. Let's consider adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, where $\mathcal{A}_1$ and $\mathcal{A}_2$ are polynomial time algorithms: the first one takes public key as input and outputs the pair of plain texts $(\mathbf{m}_1, \mathbf{m}_2)$ and state information st formed during generation of plaintexts. For $\mathbf{g} \in \{\text{CPA}, \text{CCA2}\}$ the advantage of the adversary $\mathcal{A}$ in randomized game or experiment $\mathbf{g}$ is determined by the value:

$$\mathbf{Adv}^{\mathbf{g}}_{\Sigma,\mathcal{A}}(N) = 2\Pr\left\{ \begin{array}{r} (pk, sk) \leftarrow \mathcal{K}(N) \\ (\mathbf{m}_0, \mathbf{m}_1, st) \leftarrow \mathcal{A}_1(pk) \\ b \leftarrow \{0,1\} \\ \mathbf{c} \leftarrow \{\mathbf{m}_b\}^{\Sigma}_{pk} \end{array} : \mathcal{A}_2(\mathbf{c}, st) = b \right\} - 1, \quad (1)$$

where $\Pr\{x\}$ denotes probability of the event x. Note that if $\mathbf{g} = \text{CPA}$ then $\mathcal{A}_1 = \mathcal{A}_1^{\varnothing}$ and $\mathcal{A}_2 = \mathcal{A}_2^{\varnothing}$; if $\mathbf{g} = \text{CCA2}$ then $\mathcal{A}_1 = \mathcal{A}_1^{\mathcal{O}_1}$ and $\mathcal{A}_2 = \mathcal{A}_2^{\mathcal{O}_2}$, where $\mathcal{O}_1$ and $\mathcal{O}_2$ are decryption oracles. Note that oracle $\mathcal{O}_2$ is not allowed to decrypt the challenge ciphertext $\mathbf{c}$. It is said that the cryptosystem Σ has the property of indistinguishability under chosen plaintext attack (IND-CPA) if for any polynomial algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ the advantage of $\mathbf{Adv}^{\text{CPA}}_{\Sigma,\mathcal{A}}(N)$ is a negligible function in N. Analogically the cryptosystem Σ has the property of indistinguishability under adaptive chosen cyphertext attack (IND-CCA2) if for any polynomial algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ the advantage of $\mathbf{Adv}^{\text{CCA2}}_{\Sigma,\mathcal{A}}(N)$ is a negligible function in N.

To construct the CCA2-modification of McEliece cryptosystem it is necessary to define signature scheme (SS) and one-time strongly unforgeable feature in the same way as [10]. A signature scheme is triplet of algorithms $\text{SS} = (\mathcal{K}, Sign, Check)$, where $\mathcal{K}$ is key generation algorithm which takes

as input a security parameter $N \in \mathbb{N}$ and outputs a signing-key **dsk** and a verification-key **vk**, $Sign$ is signing algorithm which takes as input a signing-key **dsk** and a message **m**, and outputs a signature σ, $Check$ is checking algorithm which takes as input a verification-key **vk** a message **m** and a signature σ, and outputs 1 if σ is valid for **m** and 0 otherwise.

Let $\mathrm{SS} = (\mathcal{K}_{\mathrm{SS}}, Sign, Chk)$ be a signature scheme and $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ is adversary, where $\mathcal{A}_1$ and $\mathcal{A}_2$ are polynomial time algorithms: the first generates message by verification key and the second tries to form new message with correct signature. Then the probability of forging signature scheme SS by the adversary $\mathcal{A}$ is determined by the value:

$$\mathrm{P}^{fal}_{\mathrm{SS},\mathcal{A}}(N) = \Pr\left\{ \begin{array}{c} (\mathbf{vk}, \mathbf{dsk}) \leftarrow \mathcal{K}(N) \\ (\mathbf{m}, st) \leftarrow \mathcal{A}_1(\mathbf{vk}) \\ \sigma \leftarrow Sign(\mathbf{dsc}, \mathbf{m}) \\ (\mathbf{m}^*, \sigma^*) \leftarrow \mathcal{A}_2(\mathbf{m}, \sigma, st) \end{array} : Chk(\mathbf{vk}, \mathbf{m}^*, \sigma^*) = 1 \right\},$$

where $(\mathbf{m}^*, \sigma^*) \neq (\mathbf{m}, \sigma)$. It is said that a signature scheme SS is one-time strongly unforgeable if for all polynomial time algorithms $\mathcal{A}$ the value $\mathrm{P}^{fal}_{\mathrm{SS},\mathcal{A}}(N)$ is negligible function of N. It is important to note, that one-time strongly unforgeable signature scheme can be constructed using one-way functions (see [12], [13]).

2.2 Original McEliece cryptosystem McE

Consider the McEliece cryptosystem $\mathrm{McE} = (\mathcal{K}_{\mathrm{McE}}, \mathcal{E}_{\mathrm{McE}}, \mathcal{D}_{\mathrm{McE}})$ on the linear $[n, k, d]$-code $C(\subseteq \mathbb{F}_q^n)$, where n is the length, k is the code dimension, and d is the minimum code distance. Let G be the generator matrix of the code C, $t = \lfloor \frac{d-1}{2} \rfloor$. A secret key sk is a pair (S, P), where S is a non-singular $(k \times k)$-matrix over the field $\mathbb{F}_q$, and P is a permutation $(n \times n)$-matrix. A public key pk is a pair $(\widetilde{G} = SGP, t)$. Encryption of a message $\mathbf{m} \in \mathbb{F}_q^k$ is performed according to the rule:

$$\{\mathbf{m}\}^{\mathrm{McE}}_{pk} = \mathbf{m}\widetilde{G} + \mathbf{e} = \mathbf{c},\ \mathbf{e} \in_R \mathcal{E}_{n,t}. \tag{2}$$

To decrypt the ciphertext **c** one should use an effective decoder $\mathrm{Dec}_C : \mathbb{F}_q^n \to \mathbb{F}_q^k$ of the code C and the secret key sk:

$$\{\mathbf{c}\}^{\mathrm{McE}}_{sk} = \mathrm{Dec}_C(\mathbf{c}P^{-1})S^{-1}. \tag{3}$$

Note that, the ciphertext **c** from (2) can be decoded by information sets. Recall that for $[n, k]_q$-code C the set $\tau = \{i_1, ..., i_k\} \subseteq [n]$ is called information set if any generator matrix G_C of C has submatix of full rank,

consisted of columns on positions from τ. For $M \in \mathbb{N}$, the vector $\mathbf{m}(\in \mathbb{F}_q^M)$ and the ordered set $\omega = \{\omega_1, ..., \omega_l\} \subseteq [M]$, where $\omega_1 < ... < \omega_l$, we consider the projection operator $\Pi_\omega : \mathbb{F}_q^M \to \mathbb{F}_q^{|\omega|}$ acting according to the rule: $\Pi_\omega(\mathbf{m}) = (m_{\omega_1}, ..., m_{\omega_l})$. The M parameter for this operator will be clear from the context. The submatrix of G_C consisted of columns on positions from τ we denote by $\Pi_\tau(G_C)$. If $\Pi_\tau(\tilde{G})$ has full rank and vector $\Pi_\tau(\mathbf{c})$ is error-free (or $\mathrm{supp}(\tau) \cap \mathrm{supp}(\mathbf{e}) = \varnothing$) then

$$\mathbf{m} = \Pi_\tau(\mathbf{c}) \cdot (\Pi_\tau(G_C))^{-1}.$$

In general, the probability of the event that the vector $\Pi_\tau(\mathbf{c})$ has no errors is negligible. But in some cases this probability can be increased, for example, when message is encrypted two times on the same public key [14]. To resist such attacks, one can use the randomization method.

2.3 Randomized McEliece McE_l

Now we consider randomized McEliece cryptosystem $\mathrm{McE}_l = (\mathcal{K}_{\mathrm{McE}_l}, \mathcal{E}_{\mathrm{McE}_l}, \mathcal{D}_{\mathrm{McE}_l})$ first proposed in [9]. Let $\mathbf{x} \in \mathbb{F}_q^{n_1}$, $\mathbf{y} \in \mathbb{F}_q^{n_2}$, $\mathbf{z} \in \mathbb{F}_q^n$, $n_1 + n_2 = n$, then $\mathbf{z} = \mathbf{x} \parallel \mathbf{y}$ will be a concatenation of the vectors $\mathbf{x}$ and $\mathbf{y}$. For the code C the encryption rule for $\mathbf{m}(\in \mathbb{F}_q^l)$ in randomized McEliece McE_l has the form:

$$\{\mathbf{m}\}_{pk}^{\mathrm{McE}_l} = \{\mathbf{m} \parallel \mathbf{r}\}_{pk}^{\mathrm{McE}} = \mathbf{c},\ \mathbf{r} \in \mathbb{F}_q^{k-l}.$$

To decrypt the ciphertext $\mathbf{c}$, it is enough to apply the rule (3) and discard the last $k - l$ symbols:

$$\{\mathbf{c}\}_{sk}^{\mathrm{McE}_l} = \Pi_{[l]}(\{\mathbf{c}\}_{sk}^{\mathrm{McE}}).$$

3 S-concatenation construction

3.1 Basic cryptosystem bMcE_l

On the basis of the cryptosystem McE_l we construct a new randomized cryptosystem $\mathrm{bMcE}_l = (\mathcal{K}_{\mathrm{bMcE}_l}, \mathcal{E}_{\mathrm{bMcE}_l}, \mathcal{D}_{\mathrm{bMcE}_l})$ and call it the basic cryptosystem. For ω consider a subset $\mathcal{G}(\omega)$ of permutations group $\mathcal{S}_k$ acting on the elements of the set $[k]$:

$$\mathcal{G}(\omega) = \{\pi \in \mathcal{S}_k : \pi(1) = \omega_1, ..., \pi(l) = \omega_l\}.$$

With every permutation π from $\mathcal{G}(\omega)$ we associate a permutation $(k \times k)$-matrix R_π. The encryption rule of basic McEliece bMcE_l has the form:

$$\{\mathbf{m}\}_{pk,\omega}^{\mathrm{bMcE}_l} = \{(\mathbf{m} \parallel \mathbf{r}_1)R_\pi\}_{pk}^{\mathrm{McE}} \parallel \{(\mathbf{m} \parallel \mathbf{r}_2)R_\pi\}_{pk}^{\mathrm{McE}} = \mathbf{c}_1 \parallel \mathbf{c}_2 = \mathbf{c},$$

where $\mathbf{m} \in \mathbb{F}_q^l$, $\omega \subset_R [k]$, $|\omega| = l$, $\mathbf{r}_1 \in_R \mathbb{F}_q^{k-l}$, $\mathbf{r}_2$ is formed in accordance with the restriction $\mathrm{supp}(\mathbf{r}_1 - \mathbf{r}_2) = [k] \setminus \omega$, $\pi \in_R \mathcal{G}(\omega)$, and error vectors $\mathbf{e}_1$ and $\mathbf{e}_2$ in McE-encryption are chosen so that $\mathbf{e}_1 \in_R \mathcal{E}_{n,t}$, $\mathbf{e}_2 \in_R \mathcal{E}_{n,t,\mathrm{supp}(\mathbf{e}_1)}$. It is obvious that

$$\mathrm{wt}(\mathbf{e}_1) + \mathrm{wt}(\mathbf{e}_2) = 2t. \tag{4}$$

To decrypt the ciphertext $\mathbf{c}$ one should calculate

$$\{\mathbf{c}\}_{sk}^{\mathrm{bMcE}_l} = \Pi_\eta(\{\mathbf{c}_1\}_{sk}^{\mathrm{McE}}), \eta = [k] \setminus \mathrm{supp}(\{\mathbf{c}_1\}_{sk}^{\mathrm{McE}} - \{\mathbf{c}_2\}_{sk}^{\mathrm{McE}}). \tag{5}$$

Note that for random $\mathbf{r}$, random ω and any $\mathbf{m}$ it is computationally hard to find ω from a vector $(\mathbf{m} \parallel \mathbf{r})R_\pi$. In average, the adversary has to enumerate

$$\binom{l + \frac{k-l}{2}}{l}$$

variants as average weight of $\mathbf{r}$ is $\frac{k-l}{2}$.

3.2 Auxiliary S-concatenation McEliece $\widehat{\mathrm{bMcE}_l^s}$

Now we construct auxiliary modification of McEliece cryptosystem $\widehat{\mathrm{bMcE}_l^s} = (\mathcal{K}_{\widehat{\mathrm{bMcE}_l^s}}, \mathcal{E}_{\widehat{\mathrm{bMcE}_l^s}}, \mathcal{D}_{\widehat{\mathrm{bMcE}_l^s}})$ based on bMcE_l. Key generation algorithm $\mathcal{K}_{\widehat{\mathrm{bMcE}_l^s}}$ takes as input a security parameter $N \in \mathbb{N}$ and outputs a public and a secret keys of the form:

$$pk = (pk_1, ..., pk_s), sk = (sk_1, ..., sk_s),$$
$$(pk_i, sk_i) = \mathcal{K}_{\mathrm{bMcE}_l}(N), i \in [s];$$

encryption algorithm $\mathcal{E}_{\widehat{\mathrm{bMcE}_l^s}}$ takes as input a public-key pk and a message $\mathbf{m} = (\mathbf{m}_1 \parallel ... \parallel \mathbf{m}_s)$ where $\mathbf{m}_i \in \mathbb{F}_q^l$, and outputs a ciphertext $\mathbf{c}$:

$$\{\mathbf{m}\}_{pk}^{\widehat{\mathrm{bMcE}_l^s}} = \mathbf{c}' = [\mathbf{c}'_{1,1} \parallel \mathbf{c}'_{1,2}] \parallel ... \parallel [\mathbf{c}'_{s,1} \parallel \mathbf{c}'_{s,2}], \tag{6}$$

where $[\mathbf{c}'_{j,1} \parallel \mathbf{c}'_{j,2}] = \{\mathbf{m}_j\}_{pk_j,\omega}^{\mathrm{bMcE}_l}$ for $j \in [s]$ and ω is chosen randomly once for all $j = 1, ..., s$. Decryption of the ciphertext $\mathbf{c}'$ is perfomed as follows. For each $\mathbf{c}'_i$ from $\mathbf{c}' = \mathbf{c}'_1 \parallel ... \parallel \mathbf{c}'_s$ it finds $\mathbf{m}'_i = \{\mathbf{c}'_i\}_{sk_i}^{\mathrm{bMcE}_l}$ and η_i according to (5) and outputs

$$\mathbf{m}' = \begin{cases} \mathbf{m}'_1 \parallel ... \parallel \mathbf{m}'_s, \text{ if } \eta_1 = ... = \eta_s \\ \perp, \text{ otherwise} \end{cases}$$

3.3 S-concatenation McEliece bMcE_l^s

Let us construct a CCA2-modification $\mathrm{bMcE}_l^s = (\mathcal{K}_{\mathrm{bMcE}_l^s}, \mathcal{E}_{\mathrm{bMcE}_l^s}, \mathcal{D}_{\mathrm{bMcE}_l^s})$ using the one-time strongly unforgeable signature scheme $\mathrm{SS} = (\mathcal{K}_{\mathrm{SS}}, Sign, Check)$. Key generation algorithm of our modification $\mathcal{K}_{\mathrm{bMcE}_l^s}$ takes as input a security parameter $N \in \mathbb{N}$ and outputs a public-key pk and a secret key sk of the form

$$pk = ((pk_i^0, pk_i^1))_{i=1}^s, sk = ((sk_i^0, sk_i^1))_{i=1}^s, \tag{7}$$

where $(pk_i^b, sk_i^b) = \mathcal{K}_{\mathrm{bMcE}_l}(N)$, $b \in \{0,1\}$, $i \in [s]$. Encryption algorithm $\mathcal{E}_{\mathrm{bMcE}_l^s}$ takes as input a public-key pk and a message $\mathbf{m} = (\mathbf{m}_1 \parallel ... \parallel \mathbf{m}_s)$ where $\mathbf{m}_i \in \mathbb{F}_q^l$, and outputs a ciphertext $\mathbf{c}$:

$$\mathbf{c} = \{\mathbf{m}\}_{pk}^{\mathrm{bMcE}_l^s} = \mathbf{c}' \parallel \mathbf{vk} \parallel \sigma,$$

where $(\mathbf{dsk}, \mathbf{vk}) = \mathcal{K}_{\mathrm{SS}}(N)$, $\mathbf{vk} = (vk_1, ..., vk_s) \in \{0,1\}^s$, $\mathbf{c}' = \{\mathbf{m}\}_{pk^{\mathbf{vk}}}^{\widehat{\mathrm{bMcE}_l^s}}$, $\sigma = Sign(\mathbf{dsk}, \mathbf{c}')$ and

$$pk^{\mathbf{vk}} = (pk_1^{vk_1}, ..., pk_s^{vk_s}). \tag{8}$$

Decryption algorithm $\mathcal{D}_{\mathrm{bMcE}_l^s}$ takes as input a secret-key sk and a ciphertext $\mathbf{c}$, and outputs either a message $\mathbf{m} \in \mathbb{F}_q^{sl}$ or a symbol $\perp$. On the first step, $\mathcal{D}_{\mathrm{bMcE}_l^s}$ checks signature of the message. If $Check(\mathbf{c}', \mathbf{vk}, \sigma) = 0$ then $\mathcal{D}_{\mathrm{bMcE}_l^s}$ outputs $\perp$, otherwise it computes and outputs $\mathbf{m}' = \{\mathbf{c}'\}_{sk^{\mathbf{vk}}}^{\widehat{\mathrm{bMcE}_l^s}}$ where $sk^{\mathbf{vk}} = (sk_1^{vk_1}, ..., sk_s^{vk_s})$.

4 Security of S-concatenation

In this section we will consider the case $q = 2$.

4.1 Security assumptions

Let McE be the McEliece cryptosystem with security parameter N. The security of McE is based on two following standard assumptions.

Assumption 1. *There is no polynomial algorithm capable of distinguishing the $(k \times n)$-matrix of the public key of the* McE *cryptosystem from a random $(k \times n)$-matrix with non-negligible probability in N.*

Assumption 2. *There is no polynomial algorithm that solves the problem of decoding a general linear code.*

According to [15], the problem of decoding a general linear code is NP-hard. Since $P \neq NP$ has not been proved, we formulate this only as an assumption.

Note that, if these assumptions hold, then one can say that McE is one way trapdoor function (or OW-CPA secure) [16]. The hardness of most McE-type cryptosystems is based on the above assumptions (for example, [9], [10], [17]). To formulate the following theorems we should introduce auxiliary assumption.

Assumption 3. *There is no polynomial algorithm that takes as input ciphertext* $\mathbf{c}$ *of the* McE *and the number* $L \in \mathbb{N}$*, and outputs* 0 *if* $\mathbf{c}$ *corresponds to an information message of a weight less than* L *and outputs* 1 *if* $\mathbf{c}$ *corresponds to an information message of weight* L *with non-negligible distinguishing advantage in the* N*.*

4.2 Semantic security of bMcE_l

Let B_t be the random variable such that $\Pr\{B_t = \beta\} = (C_n^t)^{-1}$ for any $\beta \subseteq [n]$, $|\beta| = t$. Denote by $E_{n,t,\beta}$ random variable with uniform distribution over the set $\mathcal{E}_{n,t,\beta}$.

Lemma 1. *Random variables* $E_{n,t,\varnothing}$ *and* E_{n,t,B_t} *have the same distribution.*

The random variables $E_{n,t,\varnothing}$ and E_{n,t,B_t} take values from the set $\mathcal{E}_{n,t}$. Let $\mathbf{e}$ is arbitrary vector from $\mathcal{E}_{n,t}$. By definition $\Pr\{E_{n,t,\varnothing} = \mathbf{e}\} = \frac{1}{C_n^t}$. Let us find the probability of event $E_{n,t,B_t} = \mathbf{e}$:

$$\begin{aligned}\Pr\{E_{n,t,B_t} = \mathbf{e}\} &= \Pr\{(B_t = \beta \wedge \mathrm{supp}(e) \cap \beta = \varnothing), E_{n,t,\beta} = \mathbf{e}\} \\ &= \Pr\{(B_t = \beta \wedge \mathrm{supp}(e) \cap \beta = \varnothing)\} \times \\ &\times \Pr\{E_{n,t,\beta} = \mathbf{e} | (B_t = \beta \wedge \mathrm{supp}(e) \cap \beta = \varnothing)\} \\ &= C_{n-t}^t (C_n^t)^{-1} (C_{n-t}^t)^{-1} = (C_n^t)^{-1}.\end{aligned}$$

For public matrix $\widetilde{G}$ and secret permutation $(k \times k)$-matrix R_π, $\pi \in \mathcal{G}(\omega)$, define matrix

$$\begin{pmatrix} \widetilde{G}_\omega^1 \\ \hline \widetilde{G}_\omega^2 \end{pmatrix} = R_\pi \widetilde{G},$$

where there are l rows in G_ω^1 and $k - l$ rows in G_ω^2.

Theorem 1. bMcE_l *is* IND-CPA *secure if assumptions 1-3 hold.*

Represent the ciphertext of bMcE_l as a system of the form $\{\mathbf{m}\}_{pk}^{\mathrm{bMcE}_l} = \mathbf{c} = X \parallel Y$, where

$$X = \mathbf{m}\widetilde{G}_\omega^1 \oplus \mathbf{r}_1\widetilde{G}_\omega^2 \oplus \mathbf{e}_1,$$
$$Y = \mathbf{m}\widetilde{G}_\omega^1 \oplus (\mathbf{1} \oplus \mathbf{r}_1)\widetilde{G}_\omega^2 \oplus \mathbf{e}_2,$$

where $\mathbf{1}(\in \mathbb{F}_2^{k-l})$ is a vector of ones.

Let us consider X and Y independently of each other. The part X is a ciphertext of the McE_l with the public key $\widetilde{G}' = R_\pi\widetilde{G}$. Note that the system McE_l is IND-CPA-secure [9]. Since we consider X and Y independent, then the vector $\mathbf{1} \oplus \mathbf{r}_1$ is a random vector. From here, Y differs from a ciphertext of the McE_l with the public key $\widetilde{G}'$ in distribution of error vector $\mathbf{e}_2$, chosen randomly from $\mathcal{E}_{n,t,\mathrm{supp}(\mathbf{e}_1)}$. However, as X and Y are considered independetly then according to Lemma 1, an error vector from $\mathcal{E}_{n,t,\mathrm{supp}(\mathbf{e}_1)}$ and an error vector from $\mathcal{E}_{n,t,\varnothing}$ have the same distribution. From here Y is indistinguishable ($\sim$) from a ciphertext of the McE_l with the public key $\widetilde{G}'$.

Now we have to consider the dependence between X and Y. The aim is to prove that Y does not provide any additional information about X for a polinomial time adversary. For this, we denote $Z = X \oplus Y$ and consider the vector $\mathbf{c}'$ obtained by adding X to the second part:

$$\mathbf{c}' = X \parallel (Y \oplus X) = X \parallel Z, \tag{9}$$

where

$$X = \mathbf{m}\widetilde{G}_\omega^1 \oplus \mathbf{r}_1\widetilde{G}_\omega^2 \oplus \mathbf{e}_1,$$
$$Z = \mathbf{1}\widetilde{G}_\omega^2 \oplus \mathbf{e}_1 \oplus \mathbf{e}_2.$$

Since ω is unknown for the adversary, then for a random choice of ω the vector $\mathbf{1}\widetilde{G}_\omega^2 \oplus \mathbf{e}_1$ is a ciphertext of the McE corresponding to a random information message with a fixed weight $k-l$. The vector $\mathbf{1}\widetilde{G}_\omega^2 \oplus \mathbf{e}_1 \oplus \mathbf{e}_2$ is also a ciphertext of the McE corresponding to a random information message with a fixed weight $k-l$, but with an error vector having a weight of $2t$ (see (4)). It should be noted that Z does not contain any secret information about X, except ω and the error vector $\mathbf{e}_1$. Lets show that any polynomial adversary cannot recover ω and $\mathbf{e}_1$ from Z (ω cannot be recovered from X). Suppose that the adversary can obtain ω. For this one have to know the vector $(\mathbf{0} \parallel \mathbf{1})R_\pi\widetilde{G} = \mathbf{1}\widetilde{G}_\omega^2$. But the vector $(\mathbf{0} \parallel \mathbf{1})R_\pi$ is random vector of weight $k-l$. Since the cryptosystem McE is $\mathrm{OW}-\mathrm{CPA}$ secure [16], then the adversary cannot recover $\mathbf{1}\widetilde{G}_\omega^2$ (and therefore the set ω) from Z. It also means that the adversary cannot recover $\mathbf{e}_1 \oplus \mathbf{e}_2$ and $\mathbf{e}_1$ from Z.

Note that the adversary also cannot find $\mathbf{e}_1$ from X. Then for unknown $\mathbf{e}_1$ according to Lemma 1 the error vector $\mathbf{e}_2$ has random distribution over $\mathcal{E}_{n,t}$. Now we consider $\mathbf{1}\widetilde{G}^2_\omega \oplus \mathbf{e}_2$. Since ω is unknown for the adversary, then for a random choice of ω the vector $\mathbf{1}\widetilde{G}^2_\omega \oplus \mathbf{e}_2$ is indistinguishable from a ciphertext of the McE corresponding to a random information message with a fixed weight $k-l$. However, by the assumption 3, a ciphertext of the McE corresponding to a random information message with a fixed weight $k-l$ is indistinguishable from a ciphertext of the McE corresponding to a random information message with a weight less than or equal to $k-l$. In other words, $\mathbf{1}\widetilde{G}^2_\omega \oplus \mathbf{e}_2$ is indistinguishable from a vector of the form: $\mathbf{r}'\widetilde{G} \oplus \mathbf{e}_2$, where $\mathrm{wt}(\mathbf{r}') \leqslant k-l$. According to [9] a vector of the form $\mathbf{r}'\widetilde{G} \oplus \mathbf{e}_2$ is pseudorandom vector and indistinguishable from a random vector $\mathbf{u} \in \mathbb{F}_2^n$. From here $Z = \mathbf{1}\widetilde{G}^2_\omega \oplus \mathbf{e}_2 \oplus \mathbf{e}_1$ is indistinguishable from $\mathbf{u} \oplus \mathbf{e}_1$. Thus the polynomial adversary cannot recover any information about $\mathbf{e}_1$ from Z. Summing up, the vector $\mathbf{c}'$ is indistinguishable from vector

$$\mathbf{m}\widetilde{G}^1_\omega \oplus \mathbf{u}_1 \parallel \mathbf{e}_1 \oplus \mathbf{u}_2, \tag{10}$$

where random vector $\mathbf{u}_1$ is indistinguishable from $\mathbf{r}_1\widetilde{G}^2_\omega \oplus \mathbf{e}_1$ and random vector $\mathbf{u}_2$ is indistinguishable from $\mathbf{1}\widetilde{G}^2_\omega \oplus \mathbf{e}_2$. Note, that $\mathbf{e}_1$ has information about $\mathbf{r}_1\widetilde{G}^2_\omega \oplus \mathbf{e}_1$ ($\sim \mathbf{u}_1$) but it is masked in Z by $\mathbf{1}\widetilde{G}^2_\omega \oplus \mathbf{e}_2$ ($\sim \mathbf{u}_2$). Thus $\mathbf{c}'$ is indistinguishable from a random vector.

4.3 Security of bMcE^s_l

We say that $\widehat{\mathrm{bMcE}^s_l}$ is $(1-\varepsilon)$-verifiable if there is such polynomial time algorithm *Verify* with binary output that if *Verify* outputs 1, then $\mathcal{D}_{\widehat{\mathrm{bMcE}^s_l}}$ outputs some vector $\mathbf{m}' \neq \perp$ with probability $1-\varepsilon$, otherwise, when *Verify* outputs 0 the algorithm $\mathcal{D}_{\widehat{\mathrm{bMcE}^s_l}}$ outputs $\perp$ with probability $1-\varepsilon$, where $\varepsilon = \varepsilon(N)$ is negligible in N.

Lemma 2. *$\widehat{\mathrm{bMcE}^s_l}$ is $(1-\varepsilon)$-verifiable.*

To prove we should construct algorithm *Verify*. By definition *Verify* takes as input a vector $\mathbf{c}'$ of the form (6), a public key pk and one $sk_i, i \in [s]$. Using sk_i algorithm *Verify* decrypts corresponding subvector $\mathbf{c}'_i$ and finds the set η_i ($\eta_i = \omega$ if $\mathbf{c}'_i$ does not changed during transmission). Now *Verify* calculates $\mathbf{x}_j = \mathbf{c}'_{j,1} \oplus \mathbf{c}'_{j,2}$ for each $j \neq i$ ($j \in [s]$) ($\mathbf{c}'_{j,1}, \mathbf{c}'_{j,2}$ are from (6)). Since η_i and pk (matrix $\widetilde{G}_j$) are known, then it is easy to construct the vector $(\mathbf{0} \parallel \mathbf{1})R_{\pi_i}\widetilde{G}_j$, where π_i is any permutation from $\mathcal{G}(\eta_i)$. On the next step one can calculate $\mathbf{z}_j = \mathbf{x}_j \oplus (\mathbf{0} \parallel \mathbf{1})R_{\pi_i}\widetilde{G}_j$. Now *Verify* has to check that $\mathrm{wt}(\mathbf{z}_j) = 2t$ for

each $j \in [s]$ (see (4)). If at least one check fails then *Verify* returns 0. From Singleton bound we have $2t \leqslant n-k$. So the number of coordinates without errors $n-2t$ not less than k and information set decoding algorithm (see for example [18]) may be executed. As coordinates with errors are known (from $\mathrm{supp}(\mathbf{z})$), then *Verify* can execute information set decoding algorithm in polynomial time to decrypt vector $\mathbf{x}_j$. Using decrypted $\mathbf{x}_j$ *Verify* finds η_j and checks that $\eta_j = \eta_i$ for each $j \neq i$ ($j \in [s]$). If the equalities are satisfied then *Verify* returns 1, otherwise *Verify* returns 0. So, by the construction, if *Verify* returns 1 then $\mathcal{D}_{\widehat{\mathrm{bMcE}_l^s}}(\mathbf{c}')$ will output $\mathbf{m}' \neq \perp$ with probability $1-\varepsilon'$, where ε is the probability of information set decoding algorithm error. Here, ε' is the probability that the submatrix $\widetilde{G}_j^0$ composed of columns of matrix $\widetilde{G}_j$ with numbers from $[n] \setminus \mathrm{supp}(\mathbf{z}_j)$ has not full rank. According to assumption 1 the matrix $\widetilde{G}_j$ is indistinguishable from a random matrix. Since $\widetilde{G}_j^0$ is a random set of columns of matrix $\widetilde{G}_j$, then $\widetilde{G}_j^0$ is also indistinguishable from a random $(k \times n-2t)$-matrix. According to [19] (Theorem 1) for the probability P' that a random $(k \times n-2t)$-matrix has full rank the following inequality holds:

$$P' \geqslant \begin{cases} 0.288, n-2t=k \\ 1-2^{-(n-2t-k)}, n-2t>k \end{cases} \tag{11}$$

Since $n-2t-k$ grows with growth of N, then $2^{-(n-2t-k)}$ is negligible in N. In other words, $\varepsilon' = 2^{-(n-2t-k)}$. From here the probability that information set decoding algorithm not failed for all of the s matrices is $(1-\varepsilon')^s$. It means that $\varepsilon = 1-(1-\varepsilon')^s$. For fixed s the ε is negligible in N.

Theorem 2. $\widehat{\mathrm{bMcE}_l^s}$ *is* IND-CPA *secure if assumptions 1-3 hold.*

Let $\mathbf{c} = \{\mathbf{m}\}_{pk}^{\widehat{\mathrm{bMcE}_l}} = [\mathbf{c}_{1,1} \parallel \mathbf{c}_{1,2}] \parallel ... \parallel [\mathbf{c}_{s,1} \parallel \mathbf{c}_{s,2}]$. Consider a vector $\mathbf{c}' = [\mathbf{c}_{1,1} \parallel \mathbf{c}_{1,2} \oplus \mathbf{c}_{1,1}] \parallel ... \parallel [\mathbf{c}_{s,1} \parallel \mathbf{c}_{s,2} \oplus \mathbf{c}_{s,1}]$. Using the representation (9) we have:

$$\begin{aligned} \mathbf{c}' = & [\mathbf{m}_1 \widetilde{G}_{1,\omega}^1 \oplus \mathbf{r}_{1,1} \widetilde{G}_{1,\omega}^2 \oplus \mathbf{e}_{1,1} \parallel \mathbf{1}\widetilde{G}_{1,\omega}^2 \oplus \mathbf{e}_{1,1} \oplus \mathbf{e}_{1,2}] \parallel \\ & ... \\ & [\mathbf{m}_s \widetilde{G}_{s,\omega}^1 \oplus \mathbf{r}_{s,1} \widetilde{G}_{s,\omega}^2 \oplus \mathbf{e}_{s,1} \parallel \mathbf{1}\widetilde{G}_{s,\omega}^2 \oplus \mathbf{e}_{s,1} \oplus \mathbf{e}_{s,2}] \end{aligned}$$

According to (10) we can rewrite the left parts:

$$\begin{aligned} \mathbf{c}' = & [\mathbf{m}_1 \widetilde{G}_{1,\omega}^1 \oplus \mathbf{u}_{1,1} \parallel \mathbf{1}\widetilde{G}_{1,\omega}^2 \oplus \mathbf{e}_{1,1} \oplus \mathbf{e}_{1,2}] \parallel \\ & ... \\ & [\mathbf{m}_s \widetilde{G}_{s,\omega}^1 \oplus \mathbf{u}_{s,1} \parallel \mathbf{1}\widetilde{G}_{s,\omega}^2 \oplus \mathbf{e}_{s,1} \oplus \mathbf{e}_{s,2}] \end{aligned}$$

Group the left and right parts of the subvectors together (using a simple and not secure permutation) and get an equivalent vector $\mathbf{c}''$:

$$\begin{aligned}\mathbf{c}'' =&[\mathbf{m}_1\widetilde{G}^1_{1,\omega} \oplus \mathbf{u}_{1,1} \parallel ... \parallel \mathbf{m}_s\widetilde{G}^1_{s,\omega} \oplus \mathbf{u}_{s,1}] \parallel \\ &[\mathbf{1}\widetilde{G}^2_{1,\omega} \oplus \mathbf{e}_{1,1} \oplus \mathbf{e}_{1,2} \parallel ... \parallel \mathbf{1}\widetilde{G}^2_{s,\omega} \oplus \mathbf{e}_{s,1} \oplus \mathbf{e}_{s,2}].\end{aligned}$$

Let $\bar{G}_1 = \mathrm{diag}(\widetilde{G}^1_{1,\omega}, ..., \widetilde{G}^1_{s,\omega})$, $\bar{G}_2 = (\widetilde{G}^2_{1,\omega} \parallel ... \parallel \widetilde{G}^2_{s,\omega})$, $\bar{\mathbf{u}}_1 = (\mathbf{u}_{1,1} \parallel ... \parallel \mathbf{u}_{s,1})$ and $\bar{\mathbf{e}}_j = (\mathbf{e}_{1,j} \parallel ... \parallel \mathbf{e}_{s,j})$, $(j \in \{1, 2\})$. Then we can write

$$\mathbf{c}'' = \mathbf{m}\bar{G}_1 \oplus \bar{\mathbf{u}}_1 \parallel \mathbf{1}\bar{G}_2 \oplus \bar{\mathbf{e}}_1 \oplus \bar{\mathbf{e}}_2.$$

Consider the left part. Since $\mathbf{u}_{i,1}$ $(i \in [s])$ are independent of each other, then $\bar{\mathbf{u}}_1$ is pseudorandom vector and $\mathbf{m}\bar{G}_1 \oplus \bar{\mathbf{u}}_1$ is pseudorandom vector. Now one can look at the right part. According to assumption 3 and [9] and as proved in the theorem 1 for smaller dimensions the right part can be rewritten as $\bar{\mathbf{e}}_1 \oplus \bar{\mathbf{u}}_2$, where $\bar{\mathbf{u}}_2 \in \mathbb{F}_2^{sn}$ is a pseudorandom vector. Similarly to theorem 1 vector $\bar{\mathbf{e}}_1$ consists information about $\bar{\mathbf{u}}_1$ but it is masked by $\bar{\mathbf{u}}_2$. Thus $\mathbf{c}''$ is indistinguishable from a random vector.

We will prove the following theorem by very close technique to [10].

Theorem 3. *Let* SS *be one-time strongly unforgeable signature scheme. Then* bMcE^s_l *with security parameter* N *and fixed* s *is* IND-CCA2 *secure if assumptions 1-3 hold.*

Let $\mathcal{A}$ is the IND-CCA2 adversary. Similar to [10] we consider two games: Game1 which is equivalent to the CCA2 (see (1) where $\mathbf{g} = \mathrm{CCA2}$) game and Game2, the same as Game1, except that the signature-keys $(\mathbf{vk}^*, \mathbf{dsk}^*)$ that are used for the challenge-ciphertext $\mathbf{c}^*$ are generated before the interaction with $\mathcal{A}$. Note that Game2 always outputs $\perp$ if $\mathcal{A}$ sends a decryption query $\mathbf{c} = \mathbf{c}' \parallel \mathbf{vk} \parallel \sigma$ with $\mathbf{vk} = \mathbf{vk}^*$.

In [10] (Lemma 2) it is proved that Game1 and Game2 are indistinguishable to adversary $\mathcal{A}$ if SS is one-time strongly unforgeable signature scheme. Now we will prove that $\mathbf{Adv}^{\mathrm{Game2}}_{\mathrm{bMcE}^s_l,\mathcal{A}}(N)$ is negligible in N. On the contrary, let $\mathbf{Adv}^{\mathrm{Game2}}_{\mathrm{bMcE}^s_l,\mathcal{A}}(N) > \gamma$, where $\gamma = \gamma(N)$ is not negligible in N function. Now we consider CPA-game (see (1) where $\mathbf{g} = \mathrm{CPA}$) and construct an algorithm $\widehat{\mathcal{A}}$ on the basis of $\mathcal{A}$. Let $pk^* = (pk^*_1, ..., pk^*_s)$ is a public key formed in CPA-game. $\widehat{\mathcal{A}}$ generates a pair $(\mathbf{vk}^*, \mathbf{dsk}^*) \leftarrow \mathcal{K}(N)$. Then $\widehat{\mathcal{A}}$ computes the public key pk of the form (7) by setting $pk^{\mathbf{vk}^*} = pk^*$ (see (8)) and remaining components pk^i_j are generated using $\mathcal{K}(N)$. Thus, computed pk by $\widehat{\mathcal{A}}$ is identically distributed to the pk generated by Game2. On the next step, when $\mathcal{A}$ sends $\mathbf{c} = \mathbf{c}' \parallel \mathbf{vk} \parallel \sigma$, where $\mathbf{vk} \neq \mathbf{vk}^*$, $\widehat{\mathcal{A}}$ pick an

index i such that $vk_i \neq vk_i^*$ and checks the result of $Verify(\mathbf{c}', pk, sk_i^{vk_i})$. If $Verify(\mathbf{c}', pk, sk_i^{vk_i}) = \perp$, then $\widehat{\mathcal{A}}$ returns $\perp$, otherwise $\widehat{\mathcal{A}}$, using information set decoding, decodes all vectors $[\mathbf{c}'_{i,1}]$. Note that information set decoding is performed in polynomial time because $\widehat{\mathcal{A}}$ knows the set ω and error-free coordinates ($\widehat{\mathcal{A}}$ finds ω and $\mathbf{e_1} \oplus \mathbf{e_2}$ while running the algorithm *Verify*). Thus $\widehat{\mathcal{A}}$ finds the information message $\mathbf{m}$. By the definition of *Verify* we have that $\mathcal{D}_{\widehat{\text{bMcE}_l^s}}$ output vector $\mathbf{m}$ with probability $1-\varepsilon$, where $\varepsilon = \varepsilon(N)$ is negligible in N function. From here the distribution of the output of $\widehat{\mathcal{A}}$ in CPA-game is indistinguishable from distribution of the output in Game2 (since $\widehat{\text{bMcE}_l^s}$ is $(1-\varepsilon)$-verifiable, the distributions differ by ε, where ε is negligible function). When $\mathcal{A}$ sends the challenge-messages $\mathbf{m}_0, \mathbf{m}_1$, $\widehat{\mathcal{A}}$ forwards $\mathbf{m}_0, \mathbf{m}_1$ to the CPA-game and receives a challenge ciphertext $\mathbf{c}^{*'}$. After $\widehat{\mathcal{A}}$ computes $\sigma = Sign(\mathbf{dsk}^*, \mathbf{c}^{*'})$ and sends $\mathbf{c}^* = \mathbf{c}^{*'} \parallel \mathbf{vk}^* \parallel \sigma^*$ to $\mathcal{A}$. Note that the distribution of $\mathbf{c}^*$ is indistinguishable from distribution as in Game2. From here

$$\mathbf{Adv}^{\text{CPA}}_{\text{bMcE}_l^s,\widehat{\mathcal{A}}}(N) = \mathbf{Adv}^{\text{Game2}}_{\text{bMcE}_l^s,\mathcal{A}}(N) - \varepsilon > \gamma - \varepsilon.$$

Since $\gamma - \varepsilon$ is not negligible then $\widehat{\mathcal{A}}$ breaks IND-CPA security of bMcE_l^s, which contradicts Theorem 2.

5 Assessment of the constructed system

5.1 Security parameters

Let us consider the general security parameters of the constructed system - underlying code C, plaintext length l and one-time strong signature scheme SS. Since $(pk_i^b, sk_i^b) = \mathcal{K}_{\text{bMcE}_l}(N) = \mathcal{K}_{\text{McE}}(N)$, $b \in \{0,1\}$, $i \in [s]$ then one can use known results of evaluating the code parameters of the original McEliece cryptosystem. In the general case, in [20] it is recommended to choose code parameters with at least 86 security bits (for 2020 year). So, in accordance with the table 1.1 from [21] it is suggested to use $[4096, 3604, 83]$-code with 129 security bits. The fulfilment of the assumption 3 depends on the choice of the parameter l (in our case $L = k - l$). Unfortunately, we have not yet studied the exact boundaries at which this assumption satisfied. But, in order to complicate the possibility of distinguishing by finding ω from $(\mathbf{0} \parallel \mathbf{1})R_\pi \widetilde{G} = \mathbf{1}\widetilde{G}^2_\omega$ we can recommend to choose $l = k/2$. Then the adversary has to enumerate $\binom{k/2}{k}$ variants to find ω from $\mathbf{1}\widetilde{G}^2_\omega$. It is proposed to use a one-time strong signature scheme, on the one hand resistant to quantum attacks, on the other hand, having a small public key size (since

the number of repetitions s is equal to the size of the verification key). In [22] authors compared different signature schemes. So, according to table 2 from [22] we suggest to use Stern signature as a one-time strong signature scheme with a small public key size (347 bits).

5.2 Comparison

Let us to make a comparison of the proposed scheme with some known IND-CCA2 secure schemes based on McEliece cryptosystem.

Scheme	Plaintext	Cyphertext	Public key	Secret key
Dottling et al.	l	$n \cdot \|vk\| + \|vk\| + \|\sigma\|$	$2 \cdot \|vk\| \cdot \|pk_{\text{McE}}\|$	$2 \cdot \|vk\| \cdot \|sk_{\text{McE}}\|$
Persichetti	$*$	$n \cdot k + \|vk\| + \|\sigma\|$	$2 \cdot k \cdot \|pk_{\text{McE}}\|$	$2 \cdot k \cdot \|sk_{\text{McE}}\|$
Proposed	$l \cdot \|vk\|$	$2 \cdot n \cdot \|vk\| + \|vk\| + \|\sigma\|$	$2 \cdot \|vk\| \cdot \|pk_{\text{McE}}\|$	$2 \cdot \|vk\| \cdot \|sk_{\text{McE}}\|$

$|vk|$ – verification key size, $|\sigma|$ – signature size, $|pk_{\text{McE}}|$ – McE public key size, $|sk_{\text{McE}}|$ – McE private key size, $*$ – is explained below.

Table 1: IND-CCA2 secure schemes based on McEliece cryptosystem.

All cryptosystems in the Table 1 use s repetition method. So, for convenience, we have replaced the number of repetitions s with the corresponding parameter. In particular, in Dottling and proposed scheme s is replaced with verification key size $|vk|$. In Persichetti's work, the number of repetitions is equal to code dimension k. It is important to note, that we did not specify plaintext size of the scheme proposed by Persichetti, because its original size is 1 bit. However, it can be extended to multiple bits using hard-core functions. Unfortunately, Persichetti's scheme has a large cyphertext size. For instance, in Table 2 compared cyphertext sizes for considered schemes for suggested code and one-time strong signature scheme parameters.

Scheme	Cyphertext size
Dottling et al.	1.541.659
Persichetti	14.882.331
Proposed	2.962.971

Table 2: Cyphertext sizes for suggested parameters.

Thus, for the suggested parameters, the data transfer rate of the proposed scheme is approximately 186 times higher than in Dottling's scheme (for Persichetti's scheme the data transfer rate not compared because plaintext size is not specified).

References

[1] McEliece R.J., "A Public-Key Cryptosystem Based On Algebraic Coding Theory", *DSN Progress Report*, 1978, 42–44

[2] Sidel'nikov V.M., Shestakov S.O., "On an Encoding System Constructed on the Basis of Generalized Reed – Solomon Codes", *Discrete Math. Appl.*, **4**, 1992, 439–444

[3] Borodin, M. A., Chizhov, I. V., "Effective attack on the McEliece cryptosystem based on Reed-Muller codes", *Discrete Math.*, **26**, 2014, 10–20

[4] Deundyak V.M., Kosolapov Yu.V., "The use of the direct sum decomposition algorithm for analyzing the strength of some McEliece type cryptosystems", *Vestnik YuUrGU. Ser. Mat. Model. Progr.*, **12**, 2019, 89–101

[5] Sidel'nikov V.M., "Open coding based on Reed–Muller binary codes", *Discrete Math.*, **2**, 1994, 3–20

[6] Egorova E., Kabatiansky G., Krouk E., Tavernier C., "A new code-based public-key cryptosystem resistant to quantum computer attacks", *J. Phys. Conf. Ser.*, **1163**, 2019, 1–5

[7] Deundyak V.M., Kosolapov Yu.V., "On the strength of asymmetric code cryptosystems based on the merging of generating matrices of linear codes", *Proceedings of XVI Intern. Symposium Prob. of Redundancy in Information and Control Systems. Moscow. Russia.*, 2019, 143–148

[8] Hamdaoui Y., Sendrier N., "A non asymptotic analysis of information set decoding", *IACR Cryptology ePrint Archive*, 2013

[9] Nojima R., Imai H., Kobara K., et al., "Semantic security for the McEliece cryptosystem without random oracles.", *Designs, Codes and Cryptography*, **49**, 2008, 289—305

[10] Dottling N., Dowsley R., Muller-Quade J. and Nascimento A. C. A., "A CCA2 Secure Variant of the McEliece Cryptosystem", *Transactions on Information Theory*, **58**, IEEE, 2012, 6672–6680

[11] Bellare M., Desai A., Pointcheval D. and Rogaway P., "Relations Among Notions of Security for Public-Key Encryption Schemes.", *LNCS*, Advances in Cryptology — CRYPTO '98. CRYPTO, **1462**, eds. Krawczyk H., Springer, Berlin, Heidelberg, 1998, 26–45

[12] Lamport L., "Constructing Digital Signatures from One-Way Functions", *SRI intl. CSL-98*, 1979

[13] Naor M., Yung M., "Universal One-Way Hash Functions and their Cryptographic Applications", 21st STOC., 1989, 33–43

[14] Berson T., "Failure of the McEliece public-key cryptosystem under message-resend and relatedmessage attack", *LNCS*, 17th Annual Intern. Cryptology Conf., Santa Barbara, California, USA, **1294**, Springer, Berlin, Heidelberg, 1997, 213–220

[15] Berlekamp E. R., McEliece R. J., van Tilborg H. C., "On the inherent intractability of certain coding problems", *Transaction on Information Theory*, **24**, IEEE, 1978, 384–386

[16] Kobara K., Imai H., "On the one-wayness against chosen-plaintext attacks of the Loidreau's modified McEliece PKC", *Transactions on Information Theory*, **49**, IEEE, 2003, 3160–3168

[17] Persichetti E., "On a CCA2-secure variant of McEliece in the standard model", *Provable Security*, **11192**, 2018, 165–181

[18] Peters C., "Information-Set Decoding for Linear Codes over $\mathbb{F}_q$", *PQCrypto 2010: Post-Quantum Cryptography*, Springer, Berlin, Heidelberg, 2010, 81–94

[19] Richard P. Brent, Shuhong Gao, and Alan G. B. Lauder, "Random Krylov Spaces over Finite Fields", *SIAM J. Discrete Math.*, **16**, SIAM Publications Online, 2003, 276–287

[20] Lenstra A.K. and Verheul E.R., "Selecting Cryptographic Key Sizes", *Public Key Cryptography. PKC 2000*, **1751**, Springer, Berlin, Heidelberg, 2000, 446–465

[21] Bernstein D.J., Chou T. and Schwabe P., "McBits: Fast Constant-Time Code-Based Cryptography", *Cryptographic Hardware and Embedded Systems - CHES 2013*, **8086**, Springer, Berlin, Heidelberg, 2013, 250–272

[22] Barreto A. and Misoczki R., "A new one-time signature scheme from syndrome decoding", *IACR Cryptology ePrint Archive*, 2010

Information theoretically secure key sharing protocol executing with constant noiseless public channels

Valery Korzhik, Vladimir Starostin, Muaed Kabardov,
Aleksandr Gerasimovich, Viktor Yakovlev, and Aleksej Zhuvikin

The Bonch Bruevich Saint-Petersburg State University of Telecommunications, Russia.
val-korzhik@yandex.ru, star_vs_47@mail.ru, alexgera93@gmail.com

Abstract

We propose a new key sharing protocol executing with constant public noiseless (at least of eavesdroppers) channels. In contrast to well-known protocols (like Diffie-Hellman etc.) it does not use cryptographic assumptions (like integer factoring, discrete logarithm etc.). This protocol does not imply any advantages for legitimate users against eavesdroppers except for authentication. It is based on EVSKey Scheme, proposed recently by G. Qin and Z. Ding. But because we prove that such scheme is insecure, it needs significant modification. We introduce an artificial noise and privacy amplification procedure for this purpose. Simulation results are presented concerning key bit error probabilities for both legitimate and illegal users. The error decoding probabilities are calculated for LDPC codes application. The amount of Shannon information leaking to eavesdroppers is estimated. The channel traffic needed for execution of the proposed protocol is given too.

Keywords: key sharing, physical layer security, privacy amplification, quantum computers.

1 Introduction

Solving the key sharing problem between legitimate users, connected by some communication channels, has been in research focus within many years. But it has not yet been solved completely.

The protocols based on some cryptographic assumptions (factoring, discrete log, error correction) and first of all Diffie and Hellman Scheme were known many years ago [1]–[4]. But some of them can be broken if quantum computers occur realized in the future.

A new approach to key distribution problem based on the notion of *physical layer security* (PHY) was developed in recent years (see excellent survey [5]). This approach exploits some physical properties of real communication channels connecting legitimate users sharing a secret key in the presence

of eavesdroppers. A pioneer paper by A. Wyner [6] and its extension in the papers [7], [8], covering the issue, is worth to mention too. Legitimate channels were there supposed to be superior to eavesdropper ones on the SNR (Signal-Noise Ratio) parameter.

This approach was further developed by Maurer [9]. He proposed the use of so-called *public discussion* and privacy amplification. It enables to transform unfavorable SNR of legitimate users against eavesdroppers into advantageous one at the cost of additional information exchange on public channels. Some results for the case of active eavesdropper were presented in the paper [10].

Other PHY-based protocols execute channels with random parameters (say, fading channels with multipath wave propagation). This technique was used also in MIMO-based systems intended for communication between mobile units [11]–[13]. But it is worth to note, that all the key sharing methods mentioned above had been designed for known SNR in the eavesdropper channels or for the case, where the number of antennas in the eavesdropper MIMO-based systems was limited by some value. However, such requirements to enemy system is obviously unrealistic. Key distribution problem can be solved effectively also in the frame of so-called quantum cryptography, where however special quantum channels and devices [14] should be implemented.

Also there is a demand to share secret keys between users, connected by constant (practically noiseless) channels (as Internet for example), without any cryptographic assumption due to a risk of quantum computers to be applied in the future.

In Section 2 we remind the key sharing protocol based on extraction of matrix eigenvalues described in [11] as EVSKey Scheme. We show that it is in fact insecure. Next, we improve this protocol in order to provide the upper bound for SNR in eavesdropper channel. In Section 3 we present some channel transform primitives. Section 4 is devoted to results of simulation. In Section 5 we optimize protocol parameters to provide both security and reliability of the shared key. Section 6 concludes the paper and proposes the problems for further investigation.

2 Key sharing protocol based on extraction of matrix characteristic polynomials

Let us remind the scheme EVSKey [11] used in the current paper in order to generate the *binary raw sequence* for further creation of the shared key. The scenario corresponding to this scheme is presented in Figure 1.

Before a transmission Alice (A) and Bob (B) generate their own random unitary reference matrices $X_A, X_B \in \mathbb{C}^{n\times n}$ as well as random unitary matrices $G_A, G_B \in \mathbb{C}^{n\times n}$. n is the number of "antennas" employed by each of the users, and length of pilot signal too. Matrices H_{AB}, H_{BA} are $n \times n$ channel matrices with independent Gaussian matrix elements distributed according to $(h_{AB})_{ij}, (h_{BA})_{ij} \sim CN(0;1)$. N_{A1}, N_{B1} are AWGN (Additive White Gaussian Noise) matrices $(n_{A1})_{ij}, (n_{B1})_{ij} \sim CN(0;\sigma^2)$ of proper noises for legitimate users A and B, respectively.

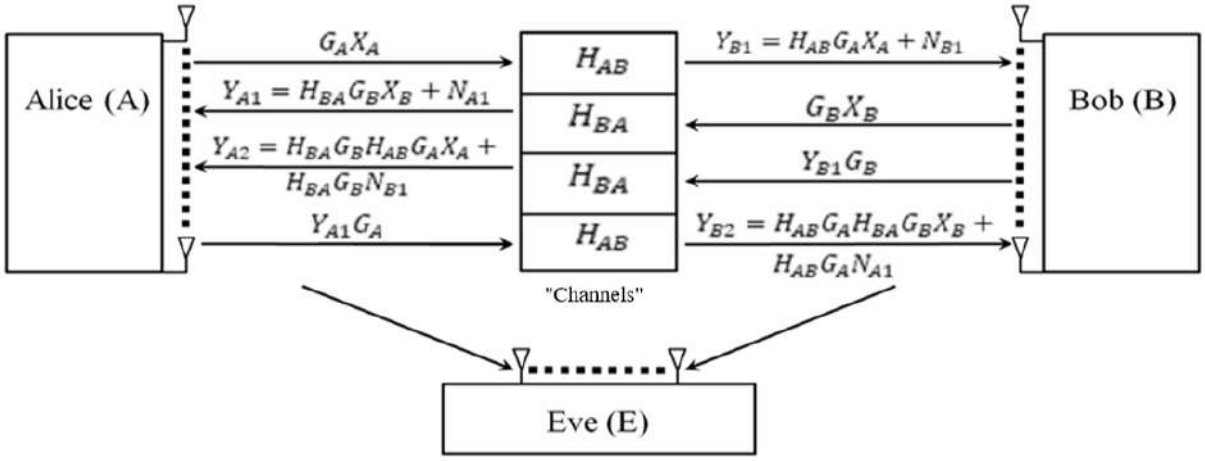


Figure 1: The scenario corresponding to EVSKey Scheme.

Let us introduce the following matrices: $P = H_{BA}G_B$, $Q = H_{AB}G_A$. Then PQ and QP can be estimated by users via the least square method as:

$$\begin{aligned} PQ &\approx Y_{A2}(X_A)^{-1} \\ QP &\approx Y_{B2}(X_B)^{-1} \end{aligned} \tag{1}$$

It is well known [11] that square matrices of the same size PQ and QP have the same eigenvalues. Therefore, they have the same *characteristic polynomials* (CP):

$$CP[PQ] = CP[QP] \tag{2}$$

Thus, from (2) we conclude that the legitimate users A and B are able to extract the same characteristic polynomials after a completion of protocol through noiseless channels although matrices PQ and QP can be different.

But, firstly, we are to demonstrate, that E is able to intercept key bits, although it was claimed in [11], that it is impossible. Unfortunately, the last statement is wrong.

In order to prove our claim, let us consider firstly a scenario where the

"combined" eavesdropper E intercepts signals:

$$\begin{array}{ll} \widetilde{Y}_{A1} = H_{BE}G_BX_B, & \widetilde{Y}_{A2} = H_{BE}G_BH_{AB}G_AX_A \\ \widetilde{Y}_{B1} = H_{AE}G_AX_A, & \widetilde{Y}_{B2} = H_{AE}G_AH_{BA}G_BX_B \end{array} \tag{3}$$

where H_{AE}, H_{BE} denote E's channel matrices and all noises are equal to zero.

Proposition 1. *Let $EV(Y)$ denote the set of eigenvalues of matrix Y. Then*

$$EV(Y) = EV(PQ) = EV(QP)$$

where

$$Y = \widetilde{Y}_{A2}(\widetilde{Y}_{B1})^{-1}\widetilde{Y}_{B2}(\widetilde{Y}_{A1})^{-1} \tag{4}$$

and $(Y)^{-1} = (Y)_P^{-1}$ is the Moore-Penrose pseudoinverse matrix [15] in the general case of rectangular (not square) matrix Y.

For all random matrices described above, the $(Y)^{-1}$ does exist with 100% probability.

Proof. Substituting (3) into (4), we get:

$$Y = H_{BE}G_BH_{AB}G_AX_AX_A^{-1}G_A^{-1}H_{AE}^{-1}H_{AE}G_AH_{BA}G_BX_BX_B^{-1}G_B^{-1}H_{BE}^{-1}$$

After simple matrix transforms, Y can be presented as follows:

$$Y = H_{BE}G_BH_{AB}G_AH_{BA}G_B(H_{BE}G_B)^{-1} = (H_{BE}G_B)QP(H_{BE}G_B)^{-1}$$

The last relation means that Y is similar to matrix QP and thus $EV(Y) = EV(QP)$ [16] for any matrices H_{AE}, H_{BE}. □

If both legitimate users and eavesdropper have practically the same noises, then simulation shows that the probabilities of the shared key bits be close to one another for legitimate users and eavesdropper.

Hence, the original scheme EVSKey is useless for key sharing. Fortunately, it can be modified with the use of artificial noises N_{A1}, N_{B1} providing lower noisy power bound for the eavesdropper, that cannot be decreased, because it is controlled by the legitimate users only.

Before we present the following part of key sharing protocol, it is important to show that not one, but, at least, two, artificial noises N_{A1}, N_{B1} should be added. Otherwise, the eavesdropper would be able to intercept the legitimate key without any errors. Indeed, let us assume that only B creates artificial noise. Then we get:

$$\begin{array}{ll} Y_{B1} = QX_A, & Y_{A2} = P(Y_{B1} + N_{B1}) \\ Y_{A1} = PX_B, & Y_{B2} = QY_{A1} \end{array}$$

Next, A extracts CP from the matrix:

$$Y_{A2}X_A^{-1} = P(Y_{B1} + N_{B1})X_A^{-1} = PQ + PN_{B1}X_A^{-1}$$

whereas B extracts the key from CP of the matrix:

$$Y_{B2}X_B^{-1} = QY_{A1}X_B^{-1} = QP$$

The eavesdropper E extracts the key from CP of the matrix:

$$Y = Y_{A2}(Y_{B1})^{-1}Y_{B2}(Y_{A1})^{-1} = PQ + PN_{B1}X_A^{-1} \tag{5}$$

Thus (5) implies that E gets exactly the same key as legitimate user A. This means that such situation has to be excluded.

3 Description of channel transform primitives

In the following section there will be presented the results of simulation regarding the key bit errors in the presence of two artificial noises N_{A1}, N_{B1}. Let P_l, P_e be the key *bit error rate* (BER) for legitimate users and eavesdropper, respectively. If legitimate users dominate over eavesdropper, that is $P_l < P_e$, then we can apply *privacy amplification theorem* [9]. It states that, there such an algorithm exists, which provides approaching to zero both key BER for legitimate users and Shannon information leaking to an eavesdropper as the length of code words is increasing.

But, for the situation, where the key BER's satisfy inequality $P_l > P_e$, it is necessary to apply in advance some additional protocol (primitive), that reduces the previous inequality to the opposite one ($P_l < P_e$).

Several examples of such primitives are given in [9]. It seems, that the best of them is protocol known as *"a preference improvement of the main channel"* (PIMC). Let us consider the protocol PIMC, slightly modified for our purpose, in more detail.

Let legitimate users exploit constant noiseless channels. Let Alice and Bob are exchanging by messages Y_{A1}, Y_{A2}, Y_{B1}, Y_{B2} as it is shown in Figure 1. Having estimated the matrices $Y_{A2}(X_A)^{-1}$, $Y_{B2}(X_B)^{-1}$ they get sets of eigenvalues, which differ due to artificial noises. After quantization (see Section 4) users get a little bit different binary strings K_A, K_B. Those raw key bit strings will be used later to form the final shared key bit string. Obviously, such a bit exchange could be interpreted in terms of binary symmetric channels without memory (BSC). Eve in her turn gets signal Y (5) (both noises N_{A1}, N_{B1} included) and therefore her sample of raw key bit string

K_E. Thus, both legitimate and illegal users implement two BSC: one with BER P_l, another – with BER P_e respectively and $P_l > P_e$ supposedly.

To inverse the last inequality legitimate users repeat S times each bit transmitting over the channel we call the main. They agree to accept such S-blocks if and only if each of them separately receives the same bit S times, no matter if these blocks are identical or not. This case both inform each other about block acceptance via the public noiseless channel.

Supposing that bit errors are independent it is easy to see, that such protocol forms the following BER in the main (legal) channel:

$$\widetilde{P}_l = \frac{2P_l^S(1-P_l)^S}{((1-P_l)^S + P_l^S)^2} \tag{6}$$

To get such S-fold bit repetition disturbed by noises Alice and Bob generate S times in a row one and the same random matrices P, Q, X_A, X_B and each time different random noise matrices N_{A1}, N_{B1} (only these two artificial noises N_{A1}, N_{B1} are added).

At the same time eavesdropper E intercepts S-blocks over BSC with BER P_e and controls public noiseless channels. E knows exactly which S-blocks were accepted by B. But because E's channel is believed statistically independent of the main channel ($A \to B$), she should take decision about bits corresponding to S-block using *majority rule*. This means, that she takes a decision, that S-block carries bit "0", if this block has more zeroes than ones. And she decides the bit to be "1", if the number of ones in that S-block is larger than that of zeroes. Then the BER after such decision will be for odd S the following:

$$\widetilde{P}_e = \sum_{i=\frac{S+1}{2}}^{S} \binom{S}{i} P_e^i (1-P_e)^{S-i} \tag{7}$$

4 Results of simulation

Quantization of eigenvalues sets.

To get binary key bit strings K_A, K_B legal users are first to quantize eigenvalue sets of matrices $Y_{A2}(X_A)^{-1}$, $Y_{B2}(X_B)^{-1}$. In a similar manner matrix Y (4) is also quantized by E. Since matrices are complex, eigenvalues can be quantized both on magnitude and on phase. To provide bit strings resembling random binary string one has to divide complex plane into parts in such a way that each eigenvalue hits each cell with equal probability. In first approximation, matrices whose eigenvalues are to be quantized equal to

PQ, QP (1). Both matrices P, Q are products of Ginibre matrix H and unitary one G, thus being both Ginibre ones too. Eigenvalues distribution of independent Ginibre matrices product is well known [17]. The *probability density function* (PDF) $f(\lambda)$ of eigenvalues λ of matrices PQ, QP is even, i.e. depends only on absolute value $r = |\lambda|$. For the product of two independent Ginibre square $n \times n$ matrices PDF equals to

$$f(\lambda) = \frac{2}{\pi n} K_0(2r) \sum_{m=0}^{n-1} \frac{r^{2m}}{(m!)^2}$$

Here $K_0(z)$ is the modified Bessel function of the second kind (of zero order).

Therefore, we divided the complex plane into sectors of equal angles. We also divided the plane into rings $[0; r_1]$, $[r_1; r_2]$, ... of equal probability to be hit by eigenvalues:

$$\int_0^{r_1} f(r)dr = \int_{r_1}^{r_2} f(r)dr = \dots$$

Thus, we got segments "of equal probabilities".

The optimal number of sectors and rings is a matter of trade-off and investigation. We mean trade-off between numbers of bits one could extract by quantization and magnitude of BER due to noises. If the number of segments N is much greater than that of eigenvalues n then the total information one could get per each session of protocol (matrix exchange) can be estimated as [13]

$$\log_2 \binom{N+n-1}{n} = \log_2 \frac{(N+n-1)(N+n-2)\dots N}{n!}$$

In this article we assume the complex plane to be divided into 8 sectors and 8 rings, i.e. $8 \times 8 = 64$ segments.

To each segment we assign an address $(i; j)$, where $i, j = 0, 1, \dots, 7$ are the numbers of sectors and rings respectively. Each eigenvalue we associate with segment address into which it gets. Addresses are converted to binary form. Thus, one gets $6 \times n$ bits string. To get longer string one repeats the procedure and concatenates the resulting strings. The final binary string forms the *raw shared key* (K_A, K_B).

Modified key sharing protocol was numerically modeled. Results of simulations are presented in Table 1. The simulations have been carried out for two sizes ($n = 4$, 64) of square matrices P, Q, for two noise values ($\sigma^2 = 0.1$, 0.2) and two values of the number of repetitions ($S = 3$, 5), when executing the primitive described in Section 3.

The table displays estimates of BER's, which occur during a single signal exchange (P_l, P_e) and those observed after S times repetition ($\widetilde{P}_l^{exp}$, $\widetilde{P}_e^{exp}$), for both legal and illegal users. The table also shows the values of the probabilities ($\widetilde{P}_l^{theor}$, $\widetilde{P}_e^{theor}$) calculated by the formulas (6, 7).

Table 1: Simulation results and theoretical predictions for the BER: of legal users $\widetilde{P}_l^{exp}$, $\widetilde{P}_l^{theor}$ and eavesdropper $\widetilde{P}_e^{exp}$, $\widetilde{P}_e^{theor}$.

$\mathbf{n = 4},\ \sigma^{\mathbf{2}} = \mathbf{0.1},\ \mathbf{P_l = 0.260},\ \mathbf{P_e = 0.212}$:

S	$\widetilde{P}_l^{exp}$	$\widetilde{P}_l^{theor}$	$\widetilde{P}_e^{exp}$	$\widetilde{P}_e^{theor}$
3	0.031	0.080	0.034	0.116
5	0.003	0.011	0.0056	0.068

$\mathbf{n = 4},\ \sigma^{\mathbf{2}} = \mathbf{0.2},\ \mathbf{P_l = 0.294},\ \mathbf{P_e = 0.251}$:

S	$\widetilde{P}_l^{exp}$	$\widetilde{P}_l^{theor}$	$\widetilde{P}_e^{exp}$	$\widetilde{P}_e^{theor}$
3	0.043	0.125	0.048	0.157
5	0.004	0.024	0.009	0.100

$\mathbf{n = 64},\ \sigma^{\mathbf{2}} = \mathbf{0.1},\ \mathbf{P_l = 0.083},\ \mathbf{P_e = 0.091}$:

S	$\widetilde{P}_l^{exp}$	$\widetilde{P}_l^{theor}$	$\widetilde{P}_e^{exp}$	$\widetilde{P}_e^{theor}$
3	0.0037	0.0015	0.019	0.023
5	0.00017	0.000012	0.0085	0.0065

$\mathbf{n = 64},\ \sigma^{\mathbf{2}} = \mathbf{0.2},\ \mathbf{P_l = 0.085},\ \mathbf{P_e = 0.117}$:

S	$\widetilde{P}_l^{exp}$	$\widetilde{P}_l^{theor}$	$\widetilde{P}_e^{exp}$	$\widetilde{P}_e^{theor}$
3	0.0039	0.0016	0.043	0.038
5	0.00014	0.000014	0.031	0.013

It is clear, that users win due to repetition procedure: error probabilities are reducing. But legal users have an advantage: their BER decreases with the increase of the matrix size more rapidly than the eavesdropper BER. Although decreasing is not so swift as theory predicts.

Both formulas (6, 7) for $\widetilde{P}_l$, $\widetilde{P}_e$ are based on the assumption of *independence* of errors between the legal and eavesdropper channels in each bit. While constructing long raw key bit string from sequence of several exchanging matrices it is true for bits originating from different independent matrices. And it is not so for bits got from eigenvalues of one and the same matrix. Probably this is one of the reasons for the discrepancy between experimental and theoretical $\widetilde{P}_l$, $\widetilde{P}_e$ values. The second reason is that both legitimate users and eavesdropper are subject to the same artificial noises N_{A1}, N_{B1}. But in

the sequel we operate only with our experimental results.

5 Optimization of key-sharing protocol parameters in order to provide given security and reliability

It has been proved by the *Enhanced Privacy Amplification Theorem* [18], that the Shannon information I received by the eavesdropper, about the final key sequence shared by the legitimate users, satisfies the inequality:

$$I \leq \frac{2^{-(k-t_c-l_0-r)}}{\alpha \ln 2} \tag{8}$$

where k is the length of the string, generated by A and B after a completion of the protocol PIMC; t_c is the Renyi (or collision) information, obtained by eavesdropper E about the string, received by E through a BSC with BER equals to $\widetilde{P}_e$; r is the number of check bits sent by one of the legitimate users to another in order to reconcile their strings; l_0 is the length of the final key. α is a coefficient, that approaches to 0.42 for any fixed r, as k, r, and $k-r$ are increasing. We recall, that the *privacy amplification procedure*, providing the inequality (8), can be performed in two stages: firstly with the use of a hash function chosen randomly from universal$_2$ class and, secondly, by special *"puncturing" of hash string* [18].

Let us consider a scenario, that allows to optimize parameters: k, r, S (see (6, 7)) for given prior values l_o, I_o (an upper bound of information leakage) and $\widetilde{P}_{ld}$ – the probability of incorrect decoding of final key string by legitimate users.

1. Given I_o, find the bound value

$$k - t_c - l_o - r = -\log_2 (I_o \alpha \ln 2) = \lambda_1 \tag{9}$$

2. Calculate the value of Renyi entropy [18]:

$$H_c = -\log_2 \left(\widetilde{P}_e^2 + \left(1 - \widetilde{P}_e\right)^2 \right)$$

3. Taking into account the relation

$$t_c = k - kH_c,$$

we get by (9)

$$kH_c - r = \lambda_1 + l_o.$$

4. In order to provide decrease of $\widetilde{P}_{ld}$ for bit string of length k and with execution of r check bits it is necessary to satisfy to the main Shannon inequality:
$$\frac{k}{k+r} < C,$$
where
$$C = 1 + \widetilde{P}_l \log_2 \widetilde{P}_l + \left(1 - \widetilde{P}_l\right) \log_2 \left(1 - \widetilde{P}_l\right)$$
is the channel capacity.

5. Therefore, to get the parameters of error correction code $(k + r; r)$ we have to solve the system of inequalities
$$\begin{cases} \frac{2^{-(k-t_c-l_0-r)}}{\alpha \ln 2} \leq I_o \\ \frac{k}{k+r} < C \end{cases}$$
It can be easily transformed into linear one:
$$\begin{cases} r \leq kH_c - \lambda_1 - l_0 \\ r > k\frac{H_L}{C} \end{cases} \tag{10}$$
where $H_L = 1 - C = -\widetilde{P}_L \log \widetilde{P}_L - (1 - \widetilde{P}_L) \log(1 - \widetilde{P}_L)$.

The system has solution if, and only if, $C > H_L/H_C$. At low BER values it implies, that $\widetilde{P}_l$ should be approximately twice smaller than $\widetilde{P}_e$. In this case the feasible region is infinite wedge on $(k; r)$ plane of dots with integer coordinates. The solution of (10) corresponding to minimum sum $k + r$ is
$$\begin{cases} r_0 = H_L \frac{\lambda_1 + l_0}{CH_c - H_L} \\ k_0 = C \frac{\lambda_1 + l_0}{CH_c - H_L} \end{cases} \tag{11}$$

We can take different values P_l, P_e from simulation results (see Table 1) and vary parameter S in (6, 7) in order to obtain new values $\widetilde{P}_l$, $\widetilde{P}_e$, those that would improve our protocol. For example one could increase the length of final key l_o or to make it more secure by decreasing the value I_o. It is worth to note, that we do not find so far a final key reliability in terms of the $\widetilde{P}_{ld}$ value, but we only guarantee (due to Shannon's theorem) the existence of such encoding and decoding procedures, that provide an approaching of this probability to zero.

Selection of the constructive encoding/decoding procedures requires further investigations. Seemingly, it should be one of the well-known class of codes like LDPC *close to the Shannon limit* for large block lengths [19]. But before, it is necessary to specify the value I_o in a reasonable manner.

Let us present a lower bound for $\widetilde{P}_{ed}$, based on Fano's inequality [20]:

$$H(U/V) \le h(\widetilde{P}_{ed}) + \widetilde{P}_{ed} \log_2(M-1) \tag{12}$$

where $H(U/V)$ is *conditional entropy* for eavesdropper E;

$$h(x) = -x \log_2 x - (1-x)\log_2(1-x), \;\; 0 \le x \le 1$$

M is the number of possible keys, in our case it is equal to 2^{l_0}. $\widetilde{P}_{ed}$ is the probability of incorrect decoding by E, that is a transformation of the key string into another one. It is worth to note that the inequality (12) entails large probability $\widetilde{P}_{ed}$ of incorrect decoding at entropy $H(U/V)$ large enough.

Hence for given $M = 2^{l_o}$ and I_o, we can find the lower bound for $\widetilde{P}_{ed}$. It is easy to show that the probability $\widetilde{P}_{ed}$ satisfying to (12) approximately is:

$$\widetilde{P}_{ed} = \frac{H(U/V)}{l_o} = \frac{l_o - I_o}{l_o} = 1 - \frac{I_o}{l_o}$$

Thus we conclude, that if $l_o = 64$ and $I_o = 10^{-3}$, then the probability of correct key string decoding $\widetilde{P}_{cd} = 1 - \widetilde{P}_{ed} = I_o/l_o = 1.5625 \cdot 10^{-5}$. It is no problem to decrease $\widetilde{P}_{ed}$ by decreasing the value I_o.

Examples:

So, let us adopt $n = 64$ and the amount of information leakage be $I_0 = 10^{-3}$ bit. Using formulas (11), we get a set (see Table 2) of parameters for error correction codes k_0, r_0 to bring in the very end (after hashing procedure) the final l_0 length keys and $I_o = 10^{-3}$.

Table 2: Parameters k_0, r_0 of the recommended error correction codes

$\sigma^2 = 0.1,\ \mathbf{P_l} = 0.083,\ \mathbf{P_e} = 0.091$:

S	3	5	3	5	3	5	3	5
$\mathbf{l_0}$	64	64	128	128	256	256	512	512
$\mathbf{k_0}$	4138	3420	7634	6310	14626	12091	28611	23651
$\mathbf{r_0}$	151	8	279	15	534	29	1044	56

$\sigma^2 = 0.2,\ \mathbf{P_l} = 0.085,\ \mathbf{P_e} = 0.117$:

S	3	5	3	5	3	5	3	5
$\mathbf{l_0}$	64	64	128	128	256	256	512	512
$\mathbf{k_0}$	884	867	1631	1599	3125	3064	6113	5993
$\mathbf{r_0}$	34	2	62	3	120	6	234	12

To implement the protocol one has to elaborate a suitable error correction code. As a trial code we chose *lower-density parity-check code* (LDPC) with parameters taken close to the presented in Table 2.

The probabilities P_{ld} of incorrect decoding of the key bit string by legitimate user after error correcting procedure had been fulfilled were got by simulation. It was conducted with given parameters and given BER obtained by formulas taken from [19]. Results could be seen in last column of Table 3.

Eventually let us estimate traffic (Tr) that is needed in order to form the final key of length l_0 after performing the key sharing protocol.

1. In one attempt users exchange $n \times n$ matrices QX_A+N_A, $P(QX_A+N_A)$, PX_B+N_B, $Q(PX_B+N_B)$. Suppose the matrix elements are represented in β bits format (for example, if real numbers are stored in standard 32 bits format, then $\beta = 64$ bits). The users need to transmit βn^2 bits for each matrix, in total $4\beta n^2$ bits.

2. As a result, each user gets n eigenvalues and, after quantization, $m = \lceil \log_2 N \rceil$ bits string for one eigenvalue or mn bits for the whole set of eigenvalues. Here N is the number of quantization levels; the brackets denote the *ceiling function*: $\lceil x \rceil = \mathrm{ceil}\,(x)$.

3. Steps 1 and 2 are repeated S times, after which Alice sends Bob nm-bits string γ_A via public channel to inform about S-fold repeats of the same bit on her side. Bob replies with a similar string γ_B. The intermediate value of traffic at this step is

$$4S\beta n^2 + 2nm$$

4. Alice and Bob reduce their bit strings with regard to coinciding ones '1' in the strings γ_A и γ_B. Let us denote by $\mathcal{P}$ the probability that a definite bit has the same value S times in a row for both users notwithstanding whether the S-blocks coincide between the users or not:

$$\mathcal{P} = \left((1 - P_l)^S + P_l^S\right)^2$$

   This probability can be replaced by *relative frequency*. After reduction of their bit strings, each of the users gets $mn\mathcal{P}$-bits string.

Now, to produce k_0-bits string one needs to fulfill the inequality

$$Xmn\mathcal{P} \geq k_0$$

by a proper choice of the number X of session reruns. The smallest possible value is

$$X = \left\lceil \frac{k_0}{nm\mathcal{P}} \right\rceil.$$

Table 3: Parameters of some LDPC codes, chosen close to recommended ones (Table 2), where I_o^{eff} is the upper bound of information leakage, corresponding to specified before (k_0, r_0) parameters LDPC codes.

$\sigma^2 = 0.1,\ P_l = 0.083,\ P_e = 0.092$:

S l_0	k_0 r_0	$\frac{k_0}{k_0 + r_0}$	I_0^{eff}(bit)	P_{ld} Tr(MB)
3 64	4140 150	0.965	0.00089	0.0046 7.11
5 64	3420 8	0.998	0.00091	0.0022 14.18
3 128	7636 276	0.965	0.00015	0.0050 13.12
5 128	6310 15	0.998	0.00354	0.0007 26.19
3 256	14630 532	0.965	0.00023	0.0026 25.14
5 256	12510 30	0.998	2×10^{-6}	0.0001 51.87
3 512	28635 1035	0.965	3×10^{-7}	0.0023 49.23
5 512	24092 57	0.998	9×10^{-7}	0.0032 99.89

$\sigma^2 = 0.2,\ P_l = 0.085,\ P_e = 0.117$:

S l_0	k_0 r_0	$\frac{k_0}{k_0 + r_0}$	I_0^{eff}(bit)	P_{ld} Tr(MB)
3 64	910 35	0.963	0.00025	0.0020 1.58
5 64	1301 3	0.998	4×10^{-15}	0.0013 5.41
3 128	1659 63	0.963	0.00014	0.0030 2.89
5 128	1599 3	0.998	0.0088	0.0016 6.63
3 256	3135 120	0.963	0.00061	0.0027 5.46
5 256	3064 6	0.998	0.00091	0.0005 12.70
3 512	6123 234	0.963	0.00052	0.0020 10.44
5 512	5996 12	0.998	0.00086	0.0003 24.86

Therefore, the traffic in total is equal to

$$Tr = \left\lceil \frac{k_0}{nm\mathcal{P}} \right\rceil \left(4S\beta n^2 + 2nm\right) \text{ bit}$$

Assuming

$$N = 8 \times 8 = 64, \quad \beta = 64, \quad \mathcal{P} = \left((1-P_l)^S + P_l^S\right)^2$$

and adding r_0 check bits one gets the traffic in total as follows:

$$Tr = \left\lceil \frac{k_0}{6n\left((1-P_l)^S + P_l^S\right)^2} \right\rceil \left(256Sn^2 + 12n\right) + r_0 \text{ bit}$$

or

$$Tr \approx 32 \cdot 10^{-6}\, n^2 S \left\lceil \frac{k_0}{6n\left((1-P_l)^S + P_l^S\right)^2} \right\rceil \text{ MB} \tag{13}$$

The calculated values by (13) are inserted in the 5th column of Table 3 (lower rows). We can see that having selected given protocol parameters, we can perform a trade-off between security (I_o) and reliability (P_{ld}).

There is only one new procedure (verification of key string authenticity), that has not been discussed before. By the way, this procedure is necessary for *any key sharing protocol in presence of an active adversary (eavesdropper).* Otherwise, the adversary could impersonate legitimate users and eventually share with them a common key.

It is possible to use authentication method based on the so-called short-key [21]. The Needham-Schroder authentication protocol [22] can also be used if users have initially distributed short keys to some trusted center. Another way is if users can provide the so-called *paring procedure* during their "face to face" meeting (like Mag Pairing or Physical vibration [23], [24]).

6 Conclusion

We have proposed key sharing protocol for noiseless public constant parameter communication channels (like Internet or "Direct seen" between users). The main novelty of our scenario is that *it is not based on some unrealistic requirements* like given SNR, cryptographic assumption for eavesdropper or multipath wave propagation, which are different for legitimate users and eavesdropper. As far as we know our solution is a novelty among the scenarios mentioned above. The core of our protocol is the *EVSKey Scheme* proposed in [11]. But we have proved that such protocol itself is insecure. Therefore, we modified it by introducing artificial noise by legitimate users

that does not allow a decreasing of this noise power by eavesdropper. We also modified the PIMC protocol of S-fold bit repetition. Next, we apply effective procedure of privacy amplification that provides both security and reliability for legitimate users. It seems at first glance that the paper [25] is devoted to a solution of the same problem as our paper. But in fact, it has only one common notion – "artificial noise", while many differences, namely:

- it is executed there either a MIMO system in fading channels or a set of "helpers"; our protocol is used with constant parameter public channel due to information exchange between two users without any helpers,
- in [25] it is created noise in "zero-space", whereas we execute special protocol imposing to eavesdropper an artificial noise,
- in [25] it is provided zero noise by "zero-forcing", but we provide a lower bound only for noise power,
- finally, in [25] it is guaranteed only some given *secrecy capacity*, but it is still unknown how to realize it, namely – how to provide constructive encoding/decoding procedures? But we, on the contrary, calculate Shannon information leakage to eavesdropper after application of the known privacy amplification procedure and find LDPC codes that provide high reliability of the key bits delivering to legitimate users.
- We have published recently the paper [27], which treated the same problem and used similar approach of its solution (artificial noise and privacy amplification). However, in the current paper we have made a significant advance. First of all now this protocol of matrices exchange is used itself to produce key bit string. In [27] it was used to produce bit string carrier for genuine key string. Secondly, now we have another method of PIMC protocol execution and application of error correction by LDPC codes with a calculation of the error probabilities after decoding procedure. The required error correction complexity and channel traffic are also presented.

One important open problem for further investigation is an elaboration of error correction algorithm for LDPC-codes. It is known from [26] that effective implementation of LDPC decoders is so-called *belief propagation* consisting of consecutive iterations where each of them has two main steps: first step – *check-node update* and second one – *variable-node update*. If the parity-check matrix of LDPC code consists of d_v ones in each column and d_c

ones in each row, then the number of arithmetic operations on the first step is approximately

$$n(d_c + \lfloor \log_2 d_c \rfloor - 2), \tag{14}$$

where $\lfloor a \rfloor$ is the greatest integer less than or equal to a. The second step takes about nd_v operations.

Let us consider the 13th example taken from Table 3 with $n = k_0 + r_0 = 3255$. Taking (217.8) usable LDPC code, we get $d_c = 8$ and the total number of arithmetic operations at one iteration is approximately 735630. The number of the requested iterations has to be specified with further investigation, but typically it is $5 \div 10$.

References

[1] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, ISBN 0-8493-8523-7, The CRC Press series on discrete mathematics and its applications, USA: CRC Press, 2000 N.W. Corporate Blvd., Boca Raton, FL 33431-9868, 1997.

[2] W. Diffie and M. E. Hellman, *New directions in cryptography*, **22**:6 (1976), 644–654.

[3] Schneier B., "Applied Cryptography", *JW Incorp.*, 1994.

[4] B. Alpern and F. B. Schneider, "Key exchange using 'keyless cryptography'", *Inf. Process. Lett.*, **16**:2 (1983), 79–81, http://dblp.unitrier.de/db/journals/ipl/ipl16.html#AlpernS83.

[5] A. Mukherjee, et al., "Principles of Layer Security in Multiuser Wireless Network: A Survey", 2014, arXiv:1011.3754.3 [cs. IP].

[6] A. Wyner, "Wire-tap channel concept", *Bell System Technical Journal*, **54** (1975), 1355–1387.

[7] I. Csiszár and J. Körner, "Broadcast channel with confidential messages", *IEEE Transactions on Information Theory*, **24**:2 (1978), 339–348.

[8] V. Korjik and V. Yakovlev, "Non-asymptotic estimates for efficiency of code jamming in a wire-tap channel", *Problems of Information Transmission*, **17** (1981), 223–22,.

[9] U. Maurer, "Secret key agreement by public discussion from common information", *IEEE Transactions on Information Theory*, **39**:3 (1993), 733–742.

[10] V. Yakovlev, V. I. Korzhik, G. Morales-Luna, "Key distribution protocols based on noisy channels in presence of an active adversary: Conventional and new versions with parameter optimization", *IEEE Transactions on Information Theory*, **54**:6 (2008), 2535–2549.

[11] D. Qin and Z. Ding, "Exploiting Multi-Antenna Non-Reciprocal Channels for Share Secret Key Generation", *IEEE Transactions on Information Forensics and Security*, **11**:10 (2016), 2691–2705.

[12] J.M.Wallace and D.K.Sharma, "Automatic-Secret Keys from Reciprocal MIMO Wireless Channel Measurements and Analysis", *IEEE Transactions on Information Forensics and Security*, **5**:3 (2010), 381–392.

[13] V.Starostin, V. Korzhik, M.Kabardov, A.Gerasimovich, V.Yakovlev, and G. Morales-Luna, "Key Generation protocol executing through non-reciprocal fading channels", *International Journal of Computer Science and Applications*, **16**:1 (2019), 1–16.

[14] C. H. Bennett, F. Bessette, G. Brassard, L. Salvail, and J. Smolin, "Experimental quantum cryptography", *J. Cryptol.*, **5**:1 (1992), 3–28, http://dl.acm.org/citation.cfm?id=146395.146396.

[15] Ben-Israel, Adi; Greville, Thomas N.E., *Generalized inverses: theory and applications*, ISBN0-387-00293-6, NY: Springer, 2003.

[16] Home and Johnson, *Matrix Analysis*, Cambr. Univ.Press, 1985.

[17] G. Akemann, Z. Burda, "Universal microscopic correlation functions for products of independent Ginibre matrices", 2012, arxiv:1208.01.87v2 [math-ph].

[18] V. Korjik, G. Morales-Luna, and V. Balakirsky, "Privacy amplification theorem for noisy main channel", *Lecture Notes in Computer Science*, **2200** (2001), 18–26.

[19] K. Shalkoska, "Implementation of LDPC Algorithm", *In C Programming Language*, ISBN9783330026049, LAP LAMBERT Academic Publishing, 2017, https://books.google.com.mx/books?id=1yNcMQAACAAJ.

[20] Fano R.M., *Transmission of Information. A statistical theory of communication*, Willy Bullisher, 1961.

[21] D. Dasgupta, A. Roy, and A. Nag, *Advances in User Authentication*, ISBN 3319588060,9783319588063, Springer Publishing Company, Inc., 2017.

[22] R.M. Needham and M.D. Schroeder, "Using Encryption for authentication in Large Network of computers", *ACM*, **21** (1978), 993–999.

[23] Jin R. et al., "MagPairing: Pairing Smartphones in close proximity using magnetometer", *IEEE Trans. of Information Forensics and Security*, **6** (2016), 1304–1319.

[24] Roy N. et al., "Faster Communication through Physical vibration", USENIX Symp. Netw. Syst. Design, 2016, 671–675.

[25] Goel S. and Negi R., "Guaranteeing Secrecy using Artificial Noise", *IEEE Trans. of Wireless Communication*, **7**:6 (2008), 180–189.

[26] M.P.C. Fossorier, M. Mihaljevic, H. Imai, "Reduced complexity iterative decoding of low-density parity check codes based on belief propagation", *IEEE Transactions on Communications*, **47**:5 (1999), 673–680.

[27] V. Korzhik, V. Starostin, M Kabardov, G. Morales-Luna, (A. Gerasimovich, V. Yakovlev , A. Zhuvikin, "Information theoretical secure key sharing protocol for noiseless public constant parameter channels with nothing cryptographic assumptions", Proc. of International conference FedCSIS (Germany), 2019, 361–366.

A digital signature scheme mCFS$^{\mathrm{QC-LDPC}}$ based on QC-LDPC codes

Ernesto Dominguez Fiallo

Institute of Cryptography, Havana University, Cuba
edominguezfiallo@nauta.cu

Abstract

In this paper, we follow the ideas of Ren et al. (IJ Network Security, 19(6), 1072-1079., 2017) and Dallot, L. (Western European Workshop on Research in Cryptology (pp. 65-77). Springer, Berlin, Heidelberg, 2008) and we propose to replace the Goppa codes with QC-LDPC codes in the digital signature scheme mCFS. With this modification, we obtain a considerable reduction in public key sizes (the main problem in code-based cryptography) whithout losing security in the scheme. Indeed, our theoretical security model is the same of the mCFS scheme and we perfomed a security analysis for the hash function and the public/private key setting with the new family of codes introduced. We also propose a set of parameters for different security levels in the scheme; for example, we can get 80 bits of security with 61 270 bits in the public key size, 128 bits of security with 179 904 bits in the public key size and 256 bits of security with 498 944 bits in the public key size.

Keywords: digital signatures scheme based on codes, QC-LDPC codes.

1 Introduction

In 1994, Shor proposed a quantum polynomial time algorithm for solving the Integer Factorization Problem and Discrete Logarithm Problem [5]. Therefore, all public-key cryptosystems that are currently used in practice will become insecure once sufficiently large quantum computers can be built. The research on post-quantum cryptography (secure in the era of quantum computer) has grown considerably. In 2015, the National Security Agency (NSA) announced a transition to quantum-resistant algorithms, and in 2016, the National Institute of Standards and Technology (NIST) published a standardization plan for post-quantum cryptography [6, 7]. Since then the code-based cryptography became a serious candidate to replace the current asymmetric cryptography.

Design a digital signature scheme based on codes is a challenging task. The classic idea of hash-based signature schemes is as follows: (a) apply a hash function $\mathcal{H}$ to the message msg getting $\mathcal{H}(msg)$, (b) consider the hash

value as the cipher text and decrypt it with the private key $\mathcal{D}_{k_{priv}}(\mathcal{H}(msg))$, (c) conform the signature as the pair $\left(msg, \mathcal{D}_{k_{priv}}(\mathcal{H}(msg))\right)$. For code based signature algorithm, however, it's very hard to accomplish the second step. The reason is the output of cipher text should be a syndrome associated with a vector whose distance to a codeword must be less than the error correction capacity t of the code.

The CFS [8] is the main proposal of digital signature algorithm based on codes. It's a probabilistic algorithm, which could not pause transforming the hash value of the message repeatedly until a valid syndrome has been found. It's uses an increment counter to tag the number of decoding attempts and on average this number is $t!$. This number could grow relatively fast and it's the reason for the main inefficiency of the CFS scheme.

In [2] it was developed a mCFS algorithm which is based on CFS signature algorithms but much secure. The idea of the proposal is to replace the counter by a random value uniformly distributed over $\{1, \ldots, 2^{n-k}\}$ but it does not solve the problem of inefficiency. Using the Merkle-Damgard principle [9, 10] to design hash functions, in [3] an improvement to the mCFS signature algorithm was proposed to obtain an efficient code based digital signature algorithm, namely mCFS$_c$. The design of the scheme is based on a compression function whose output is a syndrome associated with a vector whose distance to a codeword is less than or equal to the error correction capacity of the code t. This solves the inefficiency problems of the mCFS scheme without reducing security and the main disadvantage is that by using the Goppa codes, the sizes of keys are very large, which has been the traditional problem to solve in code-based cryptography.

In this paper we propose to replace the Goppa codes with QC-LDPC codes in the digital signature scheme mCFS. The use of this family of codes is one of the main alternatives proposed to NIST as a post quantum standard [4]. With this modification, we obtain a considerable reduction in public key sizes (the main problem in code-based cryptography) whithout losing security in the scheme. Our theoretical security model is the same of the mCFS scheme and we perfomed a security analysis for the hash function and the public/private key setting with the new family of codes introduced. We describe how to design the base compression function of the hash function and the process of generating and verifying the signature using this family of codes. We also propose a set of parameters for different security levels in the scheme.

2 Preliminaries

We follows [19] to show the essential elements of coding theory.

Definition 1. *A $[n, k]$ binary linear code $\mathcal{C}$ of length n and dimension k is a k-dimensional subspace of $\mathbb{F}_2^n$, which can be represented by two matrices; a $k \times n$ generator matrix G, such that $\mathcal{C} = \{mG,\ m \in \mathbb{F}_2^k\}$ or by a $(n-k) \times n$ parity check matrix H, such that $\mathcal{C} = \{c \in \mathbb{F}_2^n,\ cH^T = 0\}$, where c is a codeword of $\mathcal{C}$.*

Definition 2. *The Hamming weight of a binary codeword is the number of non-zero coordinates in the codeword which is denoted by $w(x)$. The Hamming distance d between two codewords is defined as the number of positions in which the two codewords differ. The minimum distance $d_{\min}$ of a code is the minimum Hamming distance between any two of its codewords.*

Definition 3. *A $p \times p$ circulant matrix over $\mathbb{F}_2[x]$ is obtained by cyclically right shifting of the first row as follows: $A = \begin{pmatrix} a_0 & a_1 & \dots & a_{p-1} \\ a_{p-1} & a_0 & \dots & a_{p-2} \\ \vdots & \vdots & \ddots & \vdots \\ a_1 & a_2 & \dots & a_0 \end{pmatrix}$. A circulant matrix is completely described by only its first row.*

If we consider the ring $\mathbb{F}_2[x]/(x^p+1)$, the map $A \rightsquigarrow a(x) = \sum_{i=0}^{p-1} a_i x^i$ is an isomorphism and therefore, a circulant matrix is associated to a polynomial in the variable x with coefficients over $\mathbb{F}_2$ given by the elements of the first row of the matrix.

Definition 4. *A Quasi-Cyclic (QC) code of length n and dimension k is a linear code where $k = k_0 \cdot p$; $n = n_0 \cdot p$ and its parity check matrix has the form*

$$H = \begin{pmatrix} H_{00} & H_{01} & \dots & H_{0(n_0-1)} \\ H_{10} & H_{11} & \dots & H_{1(n_0-1)} \\ \vdots & \vdots & \ddots & \vdots \\ H_{(r_0-1)0} & H_{(r_0-1)1} & \dots & H_{(r_0-1)(n_0-1)} \end{pmatrix}$$

where each submatrix $H_{ij},\ 0 \le i \le r_0 - 1,\ 0 \le j \le n_0 - 1$ is a circulant matrix of order p. The main property of QC codes is that each cyclic shift of a codeword by p positions is also a codeword.

Definition 5. *[1] A Low Density Parity Check (LDPC) code is a linear code admitting a parity-check matrix with constant row weight $w = \mathcal{O}(1)$ when $n \to \infty$.*

Definition 6. *A Quasi-Cyclic Low-Density Parity-Check (QC-LDPC) codes is a particular class of QC codes that are characterized by low-density-parity-check matrices.*

3 Digital signature scheme mCFS$^{\mathbf{QC\text{-}LDPC}}$

3.1 Key generation

It is considered a binary QC-LDPC code with length $n = n_0 p$, dimension $k = k_0 p$ and redundancy $r = p$, where $k_0 = n_0 - 1$.

The private key is formed by two matrices: (1) the the full-rank sparse parity-check matrix H, randomly chosen, having the following form

$$H = \begin{pmatrix} H_0 & H_1 & \dots & H_{n_0-1} \end{pmatrix}$$

where each $H_i,\ i \in [0, n_0 - 1]$ is a circulant $p \times p$ matrix with weight d_v in each row or column; and (2) the sparse $n_0 p \times n_0 p$ non-singular transformation matrix Q

$$Q = \begin{pmatrix} Q_{00} & Q_{01} & \cdots & Q_{0(n_0-1)} \\ Q_{10} & Q_{11} & \cdots & Q_{1(n_0-1)} \\ \vdots & \vdots & \ddots & \vdots \\ Q_{(n_0-1)0} & Q_{(n_0-1)1} & \cdots & Q_{(n_0-1)(n_0-1)} \end{pmatrix}$$

where each $Q_{ij},\ i, j \in [0, n_0 - 1]$ is a circulant $p \times p$ matrix. The row/column weight of Q is constant and equal to $m = \sum_{i=0}^{n_0-1} m_{ij}$ for some fixed $j \in [0, n_0 - 1]$ where m_{ij} is the row/column weight of Q_{ij}.

The public key is obtained by multiplying the matrices H and $\left(Q^T\right)^{-1}$

$$L = H \left(Q^T\right)^{-1}$$

Due to the QC structure of matrix L, it is only necessary to store the first row of it. Therefore, the public key size is $n_0 p$ bits.

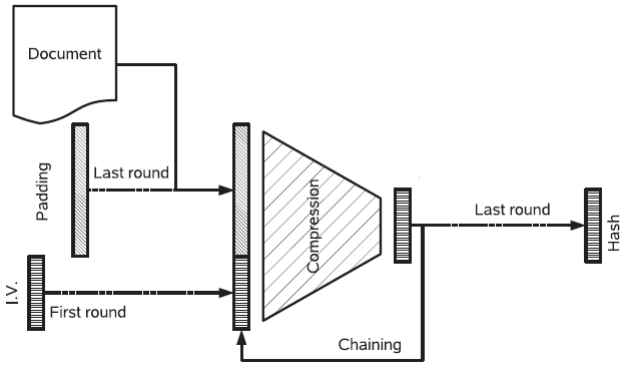
3.2 The hash function

The hash function $\mathcal{H}$ follow the ideas proposed in [11] based on Merkle-Damgard design principle. Let $f : \mathbb{F}_2^s \to \mathbb{F}_2^r,\ r < s$ be the compression function. The hash function $\mathcal{H}$ is obtained by doing the following:

– select the initial vector of length r and select $s-r$ initial bits msg_0 from a given message msg. Both vectors are concatenated forming the initial input to the compression function.

– the i-th iteration consists in the concatenation of the r output bits of the compression function f in the previous iteration $i-1$ with msg_i, that is, $s-r$ bits of the message.

The process ends when all the bits of the message are processed. During the final round, if the remaining bits of the message are insufficient to $s-r$ bits, a padding scheme should be used. The hash value will be the output of function f in the last round. The following figure illustrates the above procedure.



The compression function Given the QC-LDPC code of length n dimension k capable of correcting t errors, n_0 is selected such that $n_0 \leq t$. Any codeword can then be divided into n_0 blocks, each block of $n/n_0 = p$ bits.

Definition 7. *A codeword of weight n_0 is regular if it has exactly one non-zero position in each of the n_0 intervals* $\left[(i-1)\frac{n}{n_0}, i\frac{n}{n_0}\right]_{i=1,\ldots,n_0}$.

The public key matrix L can be divided into n_0 matrices $L = (L_1, L_2, \ldots, L_{n_0})$ of size $r \times \frac{n}{n_0}$ where

$$L_i = (l_{(i-1)\frac{n}{n_0}+1}, l_{(i-1)\frac{n}{n_0}+2}, \ldots, l_{i\frac{n}{n_0}}), \quad i = 1, \ldots, n_0$$

and $l_j,\ j = 1, \ldots, n$ is the jth column of L.

The compression function $f : \mathbb{F}_2^s \to \mathbb{F}_2^r$ where $s = n_0 \log_2 \frac{n}{n_0}$ and $r = p$ it is constructed by the algorithm 6.

Algorithm 6: The compression function

Data: $s = n_0 \log_2 \frac{n}{n_0}$ bits of the message msg

Result: a binary string of length p

1 For all $x \in \mathbb{F}_2^s$, x is divided into n_0 blocks of equal length: $x = (x_1, \ldots, x_{n_0})$, $x_i \in \mathbb{F}_2^{\log_2 \frac{n}{n_0}}$, $i = 1, \ldots, n_0$;

2 Convert each x_i to an integer between 0 and $\frac{n}{n_0} - 1$;

3 Choose the corresponding $(x_i + 1)$th column in each L_i, that is $l_{(i-1)\frac{n}{n_0}+x_i+1}$;

4 Calculate $f(x) = \bigoplus_{i=1}^{n_0} l_{(i-1)\frac{n}{n_0}+x_i+1}$;

The essential question in the definition of the above compression function is that $f(x)$ is exactly the syndrome of a regular vector y of length n and weight n_0, that is, $f(x) = Ly^T$ and $w(y) = n_0$ (see Theorem 1 of [3]). The above guarantees that the output of the hash function $\mathcal{H}$ is the syndrome of a regular vector y of length n and weight n_0.

Suppose that the message msg length is q bits. The number of binary XORs of the compression function required for each document input is estimated as follows: in each iteraion are perfomed n_0 XORs each with p bits, that is, $n_0 p = n$ XORs of bits. The number of bits read in the document at each iteration is $s - r = n_0 \log_2 \frac{n}{n_0} - p$ and therefore we have $\lceil \frac{q}{n_0 \log_2 \frac{n}{n_0} - p} \rceil$ iterations. We can approximate the number of binary XORs by

$$N_{XOR} \approx \frac{qn}{n_0 \log_2 p - p}$$

3.3 Signature generation and verification

Let msg be the message to sign, $\mathcal{H}$ the hash functionn previously proposed and $sign$ the signature. The algorithm 7 shows the process of signature generation.

Algorithm 7: Signature generation

Data: msg, L, $\mathcal{H}$

Result: $sign$

1 Choose a one-time random number $R \in \{1, 2, \ldots, 2^p\}$;

2 Calculate $x = \mathcal{H}\left(\mathcal{H}(msg) \| R\right)$;

3 Decode $v = \mathcal{D}_{ec}(x)$;

4 Calculate $y = vQ$;

5 $sign = (msg, R \| y)$;

Let $sign' = (msg, R'\|u)$ be the signed message received and the private key formed by the matrices H and Q. The algorithm 8 shows the process of signature verification.

Algorithm 8: Signature verification

Data: $sign' = (msg, R'\|u)$, L
Result: Accept or Reject
1 Calculate $a = \mathcal{H}\left(\mathcal{H}(msg)\|R'\right)$;
2 Calculate $b = Lu^T$;
3 Accept if $a = b$, Reject if otherwise;

Proof that signature verification works:

$$b = Lu^T = \left(H\left(Q^T\right)^{-1}\right)(vQ)^T = H\left(\left(Q^T\right)^{-1}Q^T\right)v^T = Hv^T$$

that is, b is a syndrome vector. If $R' = R$ and b corresponds to the output of the hash function $\mathcal{H}$, we have that

$$a = \mathcal{H}\left(\mathcal{H}(msg)\|R'\right) = \mathcal{H}\left(\mathcal{H}(msg)\|R\right) = Hv^T = b$$

4 Security Analysis

In our construction, the Goppa codes have been replaced by the QC-LDPC codes. This means two fundamental differences regarding the schemes mCFS and mCFS$_c$: (i) the random hash function h of the scheme mCFS and the Goppa code based hash function h_c of the scheme mCFS$_c$, is replaced by the QC-LDPC code based hash function $\mathcal{H}$; (ii) the relationship between the signer's public an private keys is now subject to the security of the McEliece variant based on QC-LDPC codes ([4] specifically). This means that the theorical security model of our signature scheme is equivalent to the mCFS and mCFS$_c$ schemes. We then focus the practical security analysis towards the replaced crypto primitives.

4.1 Security of the hash function

A cryptographic hash function has to be pre-image (inversion) resistant and collision resistant. For the hash function used, the inversion and collision finding is reduced to solve to two NP-complete problems: Regular Syndrome Decoding (RSD) and 2-Regular Null Syndrome Decoding (2-RNSD) [11].

From the practical point of view, in the same paper, was showed that there are two kinds of algorithms to attacks the hash function: Information

Set Decoding (ISD) and Wagner's Generalized Birthday Paradox [12]. Computational complexity is known for both attacks, that is, the Work Factor (WF):

$$WF_{Pre} = \frac{p^3 2^p}{\left(\frac{p}{n_0}\right)^{n_0}}, \qquad WF_{Col} = 2^{\frac{p}{3.3}}, \qquad WF_{Col}^{Wagner} = p2^a 2^{p/(a+1)}$$

where the parameter $a = 1, 2, \ldots$ is subject to the following restriction for its selection:

$$\frac{2^a}{a+1} \leq \frac{n_0}{p} \log_2 p$$

The use of QC codes in the hash function was proposed in [13] and was broken in [14] using a linearization technique based on the fact that ratio p/n_0 of the hash function is small ($p/n_0 \leq 2$) and does not take advantage of the QC codes.

In [15] a new collision attacks based on the structure of QC codes was introduced. This new attack is very efficient when $p/n_0 \leq 4$, and can be extended to the case $p/n_0 > 4$, but the complexity grows exponentially with p and also requires p to be a power of 2. The attack can be even improved if it is used together with Wagner's attack, so as to remove the dependency in the ration p/n_0.

The following table summarizes what was explained above and shows the WF of the attacks to the proposed hash function.

Attack	**Conditions**	**Work Factor(WF)**
Linearization	$p \leq 2n_0$	p^3
Cyclic	$p \leq 4n_0$	$(p/4)^3$
Cyclic + Wagner	-	$(p/2)2^{a'}2^{(p/2)/(a'+1)}$

For the improved cyclic attack with the use of the Wagner attack (Cyclic + Wagner), we have that $a' = a$ or $a' = a - 1$.

4.2 Security against Key Recovery Attacks (KRA)

If an attacker wants to compromise the signer's private key, he needs to perform a key recovery attack against the McEliece variant based on QC-LDPC codes. The only known way to do this -better than a exhaustive key search- is the dual code attack, which is based on the classic ISD algorithm for decoding.

In our case, only the dual code is used both in the private key matrix (private code) and in the public key matrix (public code). The relationship

between both matrices is given by $L = H\left(Q^T\right)^{-1}$ where H is sparse but $\left(Q^T\right)^{-1}$ is not. In fact, it is well known that the inverse of sparse matrix is not generally sparse. However, to evaluate the security against the dual code attack we will consider the worst possible case: $\left(Q^T\right)^{-1}$ it is also sparse of weight m. This guarantees a security level below of the actual security against the attack.

The WF of this attack is

$$WF_{DUAL}^{QC-LDPC} = \frac{WF_{ISD}(n, p, n_0 \cdot d_v \cdot m)}{p}$$

where $n_0 \cdot d_v \cdot m$ is the weight of each row of L and d_v is the column weight of H. To calculate $WF_{ISD}(n, p, n_0 \cdot d_v \cdot m)$ the approximation given in [16] can be used getting:

$$WF_{DUAL}^{QC-LDPC} = \frac{2^{n_0 \cdot d_v \cdot m \cdot \log_2 \frac{n_0}{n_0 - 1}}}{p}$$

5 Proposed parameters and comparisons

When choosing parameters we want cyclic together with Wagner's attack to be the most efficient against hash function. To avoid linearization and cyclic attacks we need $n_0 \leq p/4$. We select p prime. This is less efficient when is power of 2 but the code has the same kind of properties than a random code [17][18].

80-bit security: We select $p = 557$ which allows $n_0 \leq 139$. Set $n_0 = 110$, select the row/column weight of each circulant matrix $d_v = 9$ and select de row/column weight of the Q matrix $m = 7$. These parameters also guarantee 83 bits of security against KRA and the public key size is $n_0 p = 61\ 270$ bits.

128-bit security: We select $p = 937$ which allows $n_0 \leq 234$. Set $n_0 = 192$, select the row/column weight of each circulant matrix $d_v = 11$ and select de row/column weight of the Q matrix $m = 9$. These parameters also guarantee 134 bits of security against KRA and the public key size is $n_0 p = 179\ 904$ bits.

256-bit security: We select $p = 1\ 949$ which allows $n_0 \leq 487$. Set $n_0 = 256$, select the row/column weight of each circulant matrix $d_v = 17$ and select de row/column weight of the Q matrix $m = 11$. These parameters also

guarantee 260 bits of security against KRA and the public key size is $n_0 p = 498\ 944$ bits.

To compare the public key sizes with the scheme mCFS$_c$, we have relied on the security update to the McEliece based on Goppa codes given in [20]. The following table shows the public key sizes for each security level.

Security	**mCFS$_c$**	**mCFS$^{\mathrm{QC-LDPC}}$**
80 bits	454 839	61 270
128 bits	1 534 896	179 904
256 bits	7 685 340	498 944

6 Conclusions

We have to replace the Goppa codes with the QC-LDPC codes in the digital signature scheme mCFS which allowed us to greatly reduce the public key sizes without losing security. The corresponding security analysis was perfomed in the hash function and in the public/private key setting with the introduction of this new family of codes and a set of parameters was proposed for different levels of security.

We conjecture that with the corresponding analysis and a consequent adjustment in the parameters, the scheme is safe against attacks on quantum computers.

References

[1] Gallager, R. (1962). Low-density parity-check codes. *IRE Transactions on information theory*, 8(1), 21-28.

[2] Dallot, L. (2007, July). Towards a concrete security proof of Courtois, Finiasz and Sendrier signature scheme. In *Western European Workshop on Research in Cryptology* (pp. 65-77). Springer, Berlin, Heidelberg.

[3] Ren, F., Zheng, D., & Wang, W. (2017). An Efficient Code Based Digital Signature Algorithm. IJ Network Security, 19(6), 1072-1079.

[4] Baldi, M., Barenghi, A., Chiaraluce, F., Pelosi, G., Santini, P.: LEDApkc: LowdEnsity parity-check coDe-bAsed public-key cryptosystem. Specification revision 2.0 March 30, 2019. Available as a part of the LEDApkc submission package at: `https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Round1-Submissions`.

[5] Shor, P. W. (1994). Polynomial-time algorithm for prime factorization and discrete logarithms on a quantum computer: proc. In *35th Annual Symp. on the Foundations of Computer Science* (Vol. 124).

[6] Chen, L., Chen, L., Jordan, S., Liu, Y. K., Moody, D., Peralta, R., ... & Smith-Tone, D. (2016). *Report on post-quantum cryptography.* US Department of Commerce, National Institute of Standards and Technology.

[7] Alagic, G., Alagic, G.,Alperin-Sheriff, J., Apon, D., Cooper, D., Dang, Q.,... & Perlner, R. (2019). *Status report on the first round of the NIST post-quantum cryptography standardization process.* US Department of Commerce, National Institute of Standards and Technology.

[8] Courtois, N. T., Finiasz, M., & Sendrier, N. (2001, December). How to achieve a McEliece-based digital signature scheme. In *International Conference on the Theory and Application of Cryptology and Information Security* (pp. 157-174). Springer, Berlin, Heidelberg.

[9] Merkle, R. C. (1989, August). One way hash functions and DES. In *Conference on the Theory and Application of Cryptology* (pp. 428-446). Springer, New York, NY.

[10] Damgård, I. B. (1989, August). A design principle for hash functions. In *Conference on the Theory and Application of Cryptology* (pp. 416-427). Springer, New York, NY.

[11] Augot, D., Finiasz, M., & Sendrier, N. (2005, September). A family o f fast syndrome based cryptographic hash functions. In *International Conference on Cryptology in Malaysia* (pp. 64-83). Springer, Berlin, Heidelberg.

[12] Wagner, D. (2002, August). A generalized birthday problem. In *Annual International Cryptology Conference* (pp. 288-304). Springer, Berlin, Heidelberg.

[13] Finiasz, M., Gaborit, P., & Sendrier, N. (2007, May). Improved fast syndrome based cryptographic hash functions. In *Proceedings of ECRYPT Hash Workshop* (Vol. 2007, p. 155).

[14] Saarinen, M. J. O. (2007, December). Linearization attacks against syndrome based hashes. In *International Conference on Cryptology in India* (pp. 1-9). Springer, Berlin, Heidelberg.

[15] Fouque, P. A., & Leurent, G. (2008). Cryptanalysis of a hash function based on quasi-cyclic codes. In *Topics in Cryptology–CT-RSA 2008* (pp. 19-35). Springer, Berlin, Heidelberg.

[16] Torres, R. C., & Sendrier, N. (2016, February). Analysis of information set decoding for a sub-linear error weight. In *International Workshop on Post-Quantum Cryptography* (pp. 144-161). Springer, Cham.

[17] Chen, C. L., Peterson, W. W., & Weldon Jr, E. J. (1969). Some results on quasi-cyclic codes. *Information and Control*, 15(5), 407-423.

[18] Gaborit, P., & Zemor, G. (2008). Asymptotic improvement of the Gilbert-Varshamov bound for linear codes. *IEEE Transactions on Information Theory*, 54(9), 3865-3872.

[19] Shooshtari, M. K., Ahmadian-Attari, M., Johansson, T., & Aref, M. R. (2016). Cryptanalysis of McEliece cryptosystem variants based on quasi-cyclic low-density parity check codes. *IET Information Security*, 10(4), 194-202.

[20] Bernstein, D. J., Lange, T., & Peters, C. (2008). *Attacking and defending the McEliece cryptosystem.* In International Workshop on Post-Quantum Cryptography (pp. 31-46). Springer, Berlin, Heidelberg.

Security Analysis of the W-OTS$^+$ Signature scheme: Updating Security Bounds

Mikhail Kudinov, Evgeniy Kiktenko, and Aleksey Fedorov

Russian Quantum Center, Russia
QApp, Russia
mishel.kudinov@gmail.com, e.kiktenko@rqc.ru, akf@rqc.ru

Abstract

In this work, we discuss in detail a flaw in the original security proof of the W-OTS$^+$ variant of the Winternitz one-time signature scheme, which is an important component for various stateless and stateful many-time hash-based digital signature schemes. We update the security proof for the W-OTS$^+$ scheme and derive the corresponding security level. Our result is of importance for the security analysis of hash-based digital signature schemes.

Keywords: post-quantum cryptography, hash-based signatures, W-OTS signature.

1 Introduction

Many commonly used cryptographic systems are vulnerable with respect to attacks with the use of large-scale quantum computers. The essence of this vulnerability is the fact that quantum computers would allow solving discrete logarithm and prime factorization problems in polynomial time [1], which makes corresponding key sharing schemes and digital signatures schemes breakable. At the same time, there exist a number of mathematical operations for which quantum algorithms offer little advantage in speed. The use of such mathematical operations in cryptographic purposes allows developing quantum-resistant (or post-quantum) algorithms, i.e. cryptographic systems that remain secure under the assumption that the attacker has a large quantum computer. There are several classes of post-quantum cryptographic systems, which are based on error-correcting codes, lattices, multivariate quadratic equations and hash functions [2].

Among existing post-quantum cryptographic systems, hash-based signature schemes [3] attracted significant attention. This is easy to explain since the security of hash-based cryptographic primitives is a subject of extended

research activity, and hash functions are actively used in the existing cryptographic infrastructure. One of the main components of their security is as follows: For hash functions finding a pre-image for a given output string is computationally hard. Up to date known quantum attacks are based on Grover's algorithm [4], which gives a quadratic speed-up in the brute-force search. Quantum attacks, in this case, are capable to find (i) preimage, (ii) second preimage, and (iii) collision, with time growing sub-exponentially with a length of hash function output. Moreover, the overall performance of hash-based digital signatures makes them suitable for the practical use. Several many-time hash-based digital signatures schemes are under consideration for standardization by NIST [5] and IETF [6, 7].

We note that still the cryptographic security of hash-based digital signatures is a subject of ongoing debates, so security proofs for such schemes regularly appear (see e.g. [8, 9, 10, 11, 12, 13, 14]). These studies are partially focused on the security of basic building blocks of many-time hash-based digital signatures, which are one-time signature scheme. In particular, a variant of the Winternitz signature scheme, which is known as W-OTS$^+$ is considered. The original security proof for the W-OTS$^+$ scheme is presented in Ref. [8], and the W-OTS$^+$ scheme is used in XMSS(-MT) [7], SPHINCS [9], Gravity SPHINCS [12], and SPHINCS$^+$ [11] hash-based digital signatures. The security of many-time digital signatures obviously depends on the security level of the used one-time signature scheme.

In this work, we study the security of the W-OTS$^+$ signature scheme. We identify security flaws in the original security proof for W-OTS$^+$, which lead to the underestimated level of the security. We modify the security analysis of the W-OTS$^+$ scheme.

The paper is organized as follows. We introduce necessary definitions and notations as well as describe the W-OTS$^+$ scheme in Sec. 2. In Sec. 3 we provide a detailed updated security analysis of the W-OTS$^+$ and discuss its differences from the previous version. We conclude in Sec. 4.

2 Preliminaries

2.1 One-time and many-time hash-based signatures

The Winternitz one-time signature (W-OTS) [15, 16] has been introduced as an optimization of the seminal Lamport one-time signature scheme [17]. In order to use such one-time signature in practice several its modifications have been discussed. In particular, the W-OTS$^+$ scheme has received a sig-

nificant attention in the view of standardization processes, in which one of the candidates is the XMSS signature that uses the W-OTS$^+$ [5].

It order to use hash-based digital signatures in practice one should make them usable for many times. In order to do so it is possible to use Merkle trees. Using a root of the tree one can authenticate public keys of many one-time signature. This idea is used in several many-time hash-based signatures based on the W-OTS$^+$ scheme. The security of many-time digital signatures clearly depends on the security level of the used one-time signature scheme. The original security proof for the W-OTS$^+$ scheme is presented in Ref. [8].

We note that there are other modifications of the W-OTS scheme (e.g. see [6, 18]), however they are beyond the scope of the present paper.

2.2 Definitions and notations

We start our discussion with introducing basic definitions and notations also used in Ref. [8]. Let $x \xleftarrow{\$} X$ denote an element x chosen uniformly at random from some the set X. Let $y \leftarrow \mathsf{Alg}(x)$ denote an output of the algorithm Alg processed on the input x. We write log instead of $\log_2$ and denote a standard bitwise exclusive or operation with $\oplus$, $\lceil\cdot\rceil$ and $\lfloor\cdot\rfloor$ stand for standard ceiling and floor functions.

Definition 1 (Digital signature schemes)**.** *Let $\mathcal{M}$ be a message space. A digital signature scheme $\mathsf{Dss} = (\mathsf{Kg}, \mathsf{Sign}, \mathsf{Vf})$ is a triple of probabilistic polynomial time algorithms:*

- $\mathsf{Kg}(1^n)$ *on input of a security parameter 1^n outputs a private key sk and a public key pk;*
- $\mathsf{Sign}(\mathsf{sk}, M)$ *outputs a signature σ under secret key sk for message $M \in \mathcal{M}$;*
- $\mathsf{Vf}(\mathsf{pk}, \sigma, M)$ *outputs 1 iff σ is a valid signature on M under pk;*

such that $\forall(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Kg}(1^n), \forall(M \in \mathcal{M}) : \mathsf{Vf}(\mathsf{pk}, \mathsf{Sign}(\mathsf{sk}, M), M) = 1$.

Consider a signature scheme $\mathsf{Dss}(1^n)$, where n is the security parameter. A common definition for the security of $\mathsf{Dss}(1^n)$, which is known as the existential unforgeability under the adaptive chosen message attack (EU-CMA), is defined using the following experiment.

Experiment $\mathsf{Exp}^{\mathsf{EU-CMA}}_{\mathsf{Dss}(1^n)}(\mathcal{A})$

$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Kg}(1^n)$.

$(M^\star, \sigma^\star) \leftarrow \mathcal{A}^{\mathsf{sign}(\mathsf{sk},\cdot)}(\mathsf{pk})$.

$\{(M_i, \sigma_i)\}_{i=1}^q$ be the query answers for $\mathsf{Sign}(\mathsf{sk}, \cdot)$.

Return 1 iff $\mathrm{Vf}(\mathrm{pk}, \sigma^\star, \mathrm{M}^\star) = 1$ and $M^\star \notin \{M_i\}_{i=1}^q$.

In our work we consider one-time signatures, so the number of allowed quires q is set to 1.

Let $\mathsf{Succ}^{\mathsf{EU-CMA}}_{\mathsf{Dss}(1^n)}(\mathcal{A}) = \Pr\left[\mathsf{Exp}^{\mathsf{EU-CMA}}_{\mathsf{Dss}(1^n)}(\mathcal{A}) = 1\right]$ be the success probability of an adversary $\mathcal{A}$ in the above experiment.

Definition 2 (EU-CMA)**.** *Let $t, n \in \mathbb{N}$, $t = \mathrm{poly}(n)$, $\mathrm{Dss}(1^\mathrm{n})$ is a digital signature scheme. We call* Dss EU − CMA*-secure if the maximum success probability* $\mathrm{InSec}^{\mathrm{EU-CMA}}(\mathrm{Dss}(1^n), t)$ *of all possibly probabilistic adversaries $\mathcal{A}$ running in time $\leq t$ is negligible in n:*

$$\mathrm{InSec}^{\mathrm{EU-CMA}}(\mathrm{Dss}(1^n); t) \stackrel{\mathrm{def}}{=} \max_{\mathcal{A}} \left\{\mathsf{Succ}^{\mathsf{EU-CMA}}_{\mathsf{Dss}(1^n)}(\mathcal{A})\right\} = \mathrm{negl}(n).$$

We then consider proof of the EU-CMA property for the W-OTS$^+$ scheme on the basis of the assumption that the scheme is constructed with the function family having some particular properties. Let us discuss these required properties in detail.

Consider a function family $\mathcal{F}_n = \{f_k : \{0,1\}^n \to \{0,1\}^n\}_{k \in \mathcal{K}_n}$, where $\mathcal{K}_n$ is some set. We assume that it is possible to generate $k \stackrel{\$}{\leftarrow} \mathcal{K}_n$ and evaluate each function from $\mathcal{F}_n$ for given n in $\mathrm{poly}(n)$ time. Then, we require three basic security properties for $\mathcal{F}_n$: (i) it is one-way (OW), (ii) it has the second preimage resistance (SPR) property, and (iii) it has the undetectability (UD) property.

The success probabilities of an adversary $\mathcal{A}$ against OW and SPR of $\mathcal{F}_n$ are defined as follows:

$$\mathrm{Succ}^{\mathrm{OW}}_{\mathcal{F}_n}(\mathcal{A}) = \Pr[k \stackrel{\$}{\leftarrow} \mathcal{K}_n, x \stackrel{\$}{\leftarrow} \{0,1\}^n, y = f_k(x), x' \leftarrow \mathcal{A}(k, y) : y = f_k(x')] \quad (1)$$

and

$$\mathrm{Succ}^{\mathrm{SPR}}_{\mathcal{F}_n}(\mathcal{A}) = \Pr[k \stackrel{\$}{\leftarrow} \mathcal{K}_n, x \stackrel{\$}{\leftarrow} \{0,1\}^n, x' \leftarrow \mathcal{A}(k, x) : (x \neq x') \wedge (f_k(x) = f_k(x'))], \quad (2)$$

respectively. By using these notations, we introduce the basic definitions of OW and SPR.

Definition 3 (One-wayness and second preimage resistance of a function family)**.** *We call $\mathcal{F}_n$ one-way (second preimage resistant), if the success probability of any adversary $\mathcal{A}$ running in time $\leq t$ against the OW (SPR) of $\mathcal{F}_n$ is negligible:*

$$\mathrm{InSec}^{\mathrm{OW(SPR)}}(\mathcal{F}_n; t) \stackrel{\mathrm{def}}{=} \max_{\mathcal{A}}\{\mathrm{Succ}^{\mathrm{OW(SPR)}}_{\mathcal{F}_n}(\mathcal{A})\} = \mathrm{negl}(n). \tag{3}$$

To define the UD property we first need to introduce a definition of the (distinguishing) advantage.

Definition 4 (Advantage)**.** *Given two distributions $\mathcal{X}$ and $\mathcal{Y}$ we define the advantage* $\mathrm{Adv}_{\mathcal{X},\mathcal{Y}}(\mathcal{A})$ *of an adversary $\mathcal{A}$ in distinguishing between these two distributions as follows:*

$$\mathrm{Adv}_{\mathcal{X},\mathcal{Y}}(\mathcal{A}) = |\Pr\left[1 \leftarrow \mathcal{A}(\mathcal{X})\right] - \Pr\left[1 \leftarrow \mathcal{A}(\mathcal{Y})\right]|. \tag{4}$$

Consider two distributions $\mathcal{D}_{\mathrm{UD},\mathcal{U}}$ and $\mathcal{D}_{\mathrm{UD},\mathcal{F}_n}$ over $\{0,1\}^n \times \mathcal{K}_n$. Sampling of an element (u,k) from the first distribution $\mathcal{D}_{\mathrm{UD},\mathcal{U}}$ is realized in the following way: $u \stackrel{\$}{\leftarrow} \{0,1\}^n$, $k \stackrel{\$}{\leftarrow} \mathcal{K}_n$. Sampling of an element (u,k) from the second distribution $\mathcal{D}_{\mathrm{UD},\mathcal{F}_n}$ is realized by sampling $k \stackrel{\$}{\leftarrow} \mathcal{K}_n$ and $x \stackrel{\$}{\leftarrow} \{0,1\}^n$, and then setting $u = f_k(x)$. The advantage of an adversary $\mathcal{A}$ against the UD of $\mathcal{F}_n$ is defined as the distinguishing advantage between these distributions:

$$\mathrm{Adv}^{\mathrm{UD}}_{\mathcal{F}_n}(\mathcal{A}) = \mathrm{Adv}_{\mathcal{D}_{\mathrm{UD},\mathcal{U}},\mathcal{D}_{\mathrm{UD},\mathcal{F}_n}}(\mathcal{A}). \tag{5}$$

Definition 5 (Undetectability)**.** *We call $\mathcal{F}_n$ undetectable, if the advantage of any adversary $\mathcal{A}$ against the UD property of $\mathcal{F}_n$ running in time $\leq t$ is negligible:*

$$\mathrm{InSec}^{\mathrm{UD}}(\mathcal{F}_n; t) \stackrel{\mathrm{def}}{=} \max_{\mathcal{A}}\{\mathrm{Adv}^{\mathrm{UD}}_{\mathcal{F}_n}(\mathcal{A})\} = \mathrm{negl}(n). \tag{6}$$

2.3 The W-OTS$^+$ signature scheme

Here we describe the construction of the W-OTS$^+$ signature scheme. First of all, we define basic parameters of the scheme. Let $n \in \mathbb{N}$ be the security parameter, and m be the bit-length of signed messages, that is $\mathcal{M} = \{0,1\}^m$. Let $w \in \mathbb{N}$ be so-called Winternitz parameter, which determines a base of the representation that is used in the scheme. Let us define the following constants:

$$l_1 = \left\lceil \frac{m}{\log(w)} \right\rceil, \quad l_2 = \left\lfloor \frac{\log(l_1(w-1))}{\log(w)} \right\rfloor + 1, \quad l = l_1 + l_2. \tag{7}$$

By using the described above function family $\mathcal{F}_n$, we define a chaining function $c_k^i(x, \mathbf{r})$ for $x \in \{0,1\}^n$, $\mathbf{r} = (r_1, \ldots, r_j) \in \{0,1\}^{n \times j}$, and $j \geq i \geq 0$ as follows:

$$c_k^0(x, \mathbf{r}) = x, \quad c_k^i(x, \mathbf{r}) = f_k(c_k^{i-1}(x, \mathbf{r}) \oplus r_i) \text{ for } i > 0. \tag{8}$$

In what follows $\mathbf{r}_{a,b}$ is a substiring $(r_a, \ldots, r_b)$ of $\mathbf{r}$ if $b > a$ or it is an empty string otherwise.

Now we are ready to define the basic algorithms of the W-OTS$^+$ scheme.

Key generation algorithm ($\mathsf{Kg}(1^n)$) consists of the following steps:

1. Sample the values

$$k \xleftarrow{\$} \mathcal{K}, \quad \mathbf{r} = (r_1, \ldots, r_{w-1}) \xleftarrow{\$} \{0,1\}^{n \times (w-1)}. \tag{9}$$

2. Sample the secret signing key

$$\mathsf{sk} = (\mathsf{sk}_1, \ldots, \mathsf{sk}_l) \xleftarrow{\$} \{0,1\}^{n \times l}. \tag{10}$$

3. Compute the public key as follows:

$$\mathsf{pk} = (\mathsf{pk}_0, \mathsf{pk}_1, \ldots, \mathsf{pk}_l) = ((\mathbf{r}, k), c_k^{w-1}(\mathsf{sk}_1, \mathbf{r}), \ldots, c_k^{w-1}(\mathsf{sk}_l, \mathbf{r})). \tag{11}$$

Signature algorithm ($\mathsf{Sign}(\mathsf{sk}, M, \mathbf{r})$) consists of the following steps:

1. Convert M to the base w representation: $M = (M_1, \ldots, M_{l_1})$ with $M_i \in \{0, \ldots, w-1\}$.
2. Compute the checksum $C = \sum_{i=1}^{l_1}(w - 1 - M_i)$ and its base w representation $C = (C_1, \ldots, C_{l_2})$.
3. Set $B = (b_1, \ldots, b_l) = M||C$ as the concatenation of the base w representations of M and C.
4. Compute the signature on M as follows:

$$\sigma = (\sigma_1, \ldots, \sigma_l) = (c_k^{b_1}(\mathsf{sk}_1, \mathbf{r}), \ldots, c_k^{b_l}(\mathsf{sk}_l, \mathbf{r})). \tag{12}$$

Verification algorithm ($\mathsf{Vf}(\mathsf{pk}, \sigma, M)$) consists of the following steps:

1. Compute $(b_1, \ldots, b_l)$ as it is described in steps 1-3 of the signature algorithm.
2. Do the following comparison:

$$\mathsf{pk}_i \stackrel{?}{=} c_k^{w-1-b_i}(\sigma_i, \mathbf{r}_{b_i+1, w-1}), \quad i \in \{1, \ldots, l\}. \tag{13}$$

If the comparison holds for all i, return 1, otherwise return 0.

We assume that the runtime of all three algorithm is determined by the evaluation of f_k, while time, which is required for other operations, in negligible. Thus, the upper bound on the runtime of Kg, Sign, Vf is given by the value of lw.

3 Security of W-OTS$^+$

3.1 Security proof

In this section we consider the security proof of the W-OTS$^+$ scheme. The general line of our proof coincides with the one from Ref. [8]. However there are important differences, which yield another expression for the resulting security value.

Theorem 1. *Let $n, w, m \in \mathbb{N}$ and $w, m = \mathrm{poly}(n)$. Let $\mathcal{F}_n = \{f_k : \{0,1\}^n \to \{0,1\}^n\}_{k \in \mathcal{K}_n}$ be a one-way, second preimage resistant, and undetectable function family. Then, the insecurity of the W-OTS$^+$ scheme against an EU-CMA attack is bounded by*

$$\begin{aligned}&\mathrm{InSec}^{\mathrm{EU-CMA}}(\text{W-OTS}^+(1^n, w, m); t, 1)\\ &\quad < lw \cdot \left(w \cdot \mathrm{InSec}^{\mathrm{UD}}(\mathcal{F}_n; \widetilde{t}) + \mathrm{InSec}^{\mathrm{OW}}(\mathcal{F}_n; \widetilde{t}) + w \cdot \mathrm{InSec}^{\mathrm{SPR}}(\mathcal{F}_n; \widetilde{t})\right) \quad (14)\end{aligned}$$

with $\widetilde{t} = t + 3lw + w - 2$, where time is given in number of evaluation function from $\mathcal{F}$.

Proof. The proof is by contrapositive. Suppose there exists an adversary $\mathcal{A}$ that can produce existential forgeries for W-OTS$^+(1^n, w, m)$ scheme by running an adaptive chosen message attack in time $\leq t$ with the success probability $\varepsilon_{\mathcal{A}} \equiv \mathrm{Succ}^{\mathrm{EU-CMA}}_{\text{W-OTS}(1^n, w, m)}(\mathcal{A})$.

Then we are able to construct an oracle machine $\mathcal{M}^{\mathcal{A}}$ that either breaks the OW or SPR of $\mathcal{F}_n$ using the adversary algorithm $\mathcal{A}$. Consider a pseudocode description of $\mathcal{M}^{\mathcal{A}}$ in Algorithm 9 and block scheme in Fig. 1(a).

The algorithm is based on the following idea. We generate a pair of W-OTS$^+$ keys, and then introduce OW and SPR challenges in the αth chain, where the index of the chain α, position of the OW challenge β, and position of the SPR challenge γ are picked up at random [see also Fig. 1(b)]. Then we submit a modified public key pk' to $\mathcal{A}$. The adversary can ask to provide a signature for some message M. If the element b_α calculated from M is less than β, that is it locates below our challenge y_c, then we are not able to generate a signature and we abort. Otherwise, we compute the signature σ with respect to our modified public key and give it to $\mathcal{A}$. Finally, we obtain

some forged message-signature pair (M', σ'), and if the forgery is valid then σ' eventually contains the solution for the one of our challenges. Otherwise $\mathcal{M}^{\mathcal{A}}$ return fail.

Algorithm 9: $\mathcal{M}^{\mathcal{A}}$

Input : Security parameter n, function key k, OW challenge y_c and SPR challenge x_c.
Output: A value x that is either a preimage of y_c (i.e. $f_k(x) = y$) or a second preimage for x_c under f_k (i.e. $f(x_c) = f(x)$ and $x \neq x_c$) or fail.

1 Generate W-OTS$^+$ key pair: $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Kg}(1^n)$
2 Choose random indices $\alpha \xleftarrow{\$} \{1, \ldots, l\}, \beta \xleftarrow{\$} \{1, \ldots, w-1\}$
3 **if** $\beta = w - 1$ **then**
4 set $\mathbf{r}' = \mathbf{r}$
5 **else**
6 Choose random index $\gamma \xleftarrow{\$} \{\beta + 1 \ldots w - 1\}$
7 Set $\mathbf{r}' = \mathbf{r}$ and replace r'_γ by $c_k^{\gamma-\beta-1}(y_c, \mathbf{r}_{\beta+1,w-1}) \oplus x_c$
8 Obtain modified public key pk' by setting $\mathsf{pk}'_0 = (\mathbf{r}', k)$, $\mathsf{pk}'_i = c_k^{w-1}(\mathsf{sk}_i, \mathbf{r}')$ for $1 \leq i \leq l, i \neq \alpha$, and $\mathsf{pk}'_\alpha = c_k^{w-1-\beta}(y_c, \mathbf{r}'_{\beta+1,w-1})$
9 Run $\mathcal{A}^{\mathsf{Sign}(\mathsf{sk},\cdot)}(\mathsf{pk}')$
10 **if** $\mathcal{A}^{\mathsf{Sign}(\mathsf{sk},\cdot)}(\mathsf{pk}')$ queries to sign message M **then**
11 Compute $B = (b_1, \ldots, b_l)$ which corresponds to M
12 **if** $b_\alpha < \beta$ **then**
13 **return** fail
14 Generate signature σ of M with respect to the modified public key:
 i. Run $\sigma = (\sigma_1, \ldots, \sigma_l) \leftarrow \mathsf{Sign}(M, \mathsf{sk}, \mathbf{r}')$
 ii. Set $\sigma_\alpha = c_k^{b_\alpha-\beta}(y_c, \mathbf{r}'_{\beta+1,w-1})$
15 Reply to the query using σ
16 **if** $\mathcal{A}^{\mathsf{Sign}(\mathsf{sk},\cdot)}(\mathsf{pk}')$ returns valid (σ', M') **then**
17 Compute $B' = (b'_1, \ldots, b'_l)$ which corresponds to M'
18 **if** $b'_\alpha \geq \beta$ **then**
19 **return** fail
20 **else if** $\beta = w - 1$ **or** $c_k^{\beta-b'_\alpha}(\sigma'_\alpha, \mathbf{r}'_{b'_\alpha+1,w-1}) = y_c$ **then**
21 **return** preimage $c_k^{w-1-b'_\alpha-1}(\sigma'_\alpha, \mathbf{r}'_{b'_\alpha+1,w-1}) \oplus r_\beta$
22 **else if** $x' = c_k^{\gamma-b'_\alpha-1}(\sigma'_\alpha, \mathbf{r}'_{b'_\alpha+1,w-1}) \oplus \mathbf{r}_\gamma \neq x_c$ **and** $c_k^{\gamma-b'_\alpha}(\sigma'_\alpha, \mathbf{r}'_{b'_\alpha+1,w-1}) = c_k^{\gamma-\beta}(y_c, \mathbf{r}_{\beta+1,w-1})$ **then**
23 **return** second preimage $x' = c_k^{\gamma-b'_\alpha-1}(\sigma'_\alpha, \mathbf{r}'_{b'_\alpha+1,w-1}) + r'_\gamma$.
24 **else**
25 **return** fail
26 **else**
27 **return** fail

We start with computing the success probability of $\mathcal{M}^{\mathcal{A}}$ in solving one of the challenges. Let $\widetilde{\epsilon}_{\mathcal{A}}$ be a probability that Algorithm 9 execution comes to the line 20. More formally, it can be written as follows:

$$\widetilde{\epsilon}_{\mathcal{A}} = \Pr[b_\alpha \geq \beta \wedge \text{“Forgery is valid”} \wedge b'_\alpha < b_\alpha], \tag{15}$$

where the event “Forgery is valid” stands for $(1 \leftarrow \mathsf{Vf}(\mathsf{pk}; \sigma'; M')) \wedge (M' \neq$

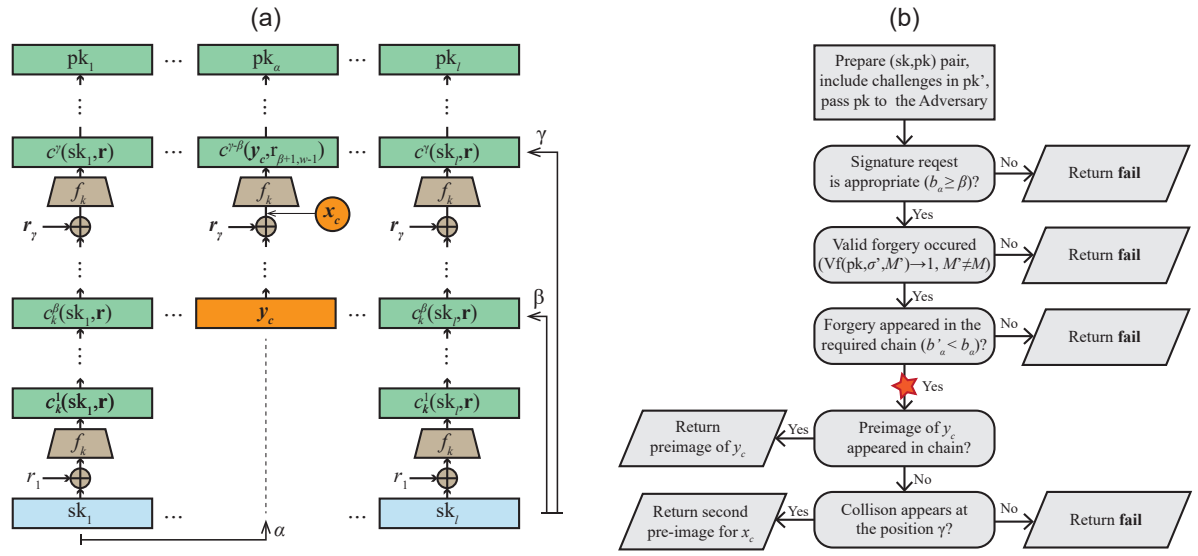


Figure 1: In (a) an introducing image and second pre-image challenges in the public key of the W-OTS$^+$ scheme is shown.
In (b) the block scheme of $\mathcal{M}^{\mathcal{A}}$ is depicted. A bullet marks a point for UD challenge.

M). We denote the whole event of Eq. (15) as "Forgery is fortunate".

We then can consider two mutually exclusive cases: either (i) $\beta = w - 1$ or the chain started from σ'_α come to y_c at the βth level, or (ii) $\beta < w-1$ and the chain started from σ'_α does not come to y_c at the βth level. Let these two case realizing with probabilities p and $(1-p)$ correspondingly conditioned by the event "Forgery is fortunate".

In the first case, the adversary $\mathcal{A}$ somehow found a preimage for the y_c. The total probability of this event is upper bounded by $\mathrm{InSec}^{\mathrm{OW}}(\mathcal{F}_n; \widetilde{t})$, so we can write

$$p \cdot \widetilde{\epsilon}_{\mathcal{A}} \leq \mathrm{InSec}^{\mathrm{OW}}(\mathcal{F}_n; \widetilde{t}). \tag{16}$$

The time $\widetilde{t} = t + 3lw + w - 2$ appears as the upper bound on the total running time of $\mathcal{A}$ plus each of the W-OTS$^+$ algorithms Kg, Sign, and Vf plus preparing αth chain in pk' (see line 8 in Algorithm 9).

In the second case, we have a collision somewhere between $(\beta+1)$th and $(w-1)$ level. If the collision appears at the level γ we obtain the second preimage of x_c. Since the SPR challenge was taken uniformly at random, the value of r'_γ remains to be a uniformly random variable, therefore there is no way for $\mathcal{A}$ to detect and intentionally avoid the position γ. Thus, we obtain the collision at the level γ with probability $(w-1-\beta)^{-1} > w^{-1}$ conditioned by the event "Forgery is fortunate". On the other hand, this probability is upper bounded by $\mathrm{InSec}^{\mathrm{SPR}}(\mathcal{F}_n; \widetilde{t})$. So we have

$$(1-p)\frac{\widetilde{\epsilon}_{\mathcal{A}}}{w} < \mathrm{InSec}^{\mathrm{SPR}}(\mathcal{F}_n; \widetilde{t}). \tag{17}$$

Again, the time $\widetilde{t} = t + 3lw + w - 2$ appears as the upper bound on the total running time of our algorithm.

By combining Eq. (16) and Eq. (17), we obtain the following expression:

$$\widetilde{\epsilon}_{\mathcal{A}} < \mathrm{InSec}^{\mathrm{OW}}(\mathcal{F}_n; \widetilde{t}) + w \cdot \mathrm{InSec}^{\mathrm{SPR}}(\mathcal{F}_n; \widetilde{t}). \tag{18}$$

In the remainder of the proof we derive a lower bound for $\widetilde{\epsilon}_{\mathcal{A}}$ as the function of $\epsilon_{\mathcal{A}}$. We note that in general $\mathcal{A}$ may behave in a 'nasty' way making $\widetilde{\epsilon}_{\mathcal{A}} \ll \epsilon_{\mathcal{A}}$ e.g. by always asking to sign 'bad' messages with $b_\alpha < \beta$ or avoiding forgeries in 'good' positions $b'_\alpha > b_\alpha$. In other words, the algorithm may avoid crossing the point shown in Fig. 1(a). This behaviour of $\mathcal{A}$ means that it can somehow reveal the challenge position from the modified public key pk'. We below consider the strategy of using this possible ability of $\mathcal{A}$ to break UD property.

Consider two distributions $\mathcal{D}_{\mathcal{M}}$ and $\mathcal{D}_{\mathsf{Kg}}$ over $\{1, \ldots, w-1\} \times \{0,1\}^n \times \{0,1\}^{n \times (w-1)} \times \mathcal{K}_n$. An element $(\beta, u, \mathbf{r}, k)$ is obtained from $\mathcal{D}_{\mathcal{M}}$ by generating all subelements β, u, $\mathbf{r}$, and k uniformly at random from the corresponding sets. At the same time, an element $(\beta, u, \mathbf{r}, k)$ is obtained from $\mathcal{D}_{\mathsf{Kg}}$ by generating β, $\mathbf{r}$, and k uniformly at random, but setting $u = c_k^{\beta}(x, \mathbf{r})$ with $x \xleftarrow{\$} \{0,1\}^n$. One can see that $\mathcal{D}_{\mathsf{Kg}}$ corresponds to the generation of elements in W-OTS$^+$ signature chain from the secret key element up to the βth level.

Consider a pseudocode of Algorithm 10 of a machine $\mathcal{M}'^{\mathcal{A}}$ taking the security parameter n and an element from either $\mathcal{D}_{\mathcal{M}}$ or $\mathcal{D}_{\mathsf{Kg}}$ as input. One can see that the operation of $\mathcal{M}'^{\mathcal{A}}$ is very similar to the operation of $\mathcal{M}^{\mathcal{A}}$.

Given an input $(\beta, u, \mathbf{r}, k)$ from $\mathcal{D}_{\mathcal{M}}$, $\mathcal{M}'^{\mathcal{A}}$ sets $y_c = u$ and then works exactly as $\mathcal{M}$ up to line 19 of the Algorithm 9. If the event "Forgery is fortunate" happens, then $\mathcal{M}'^{\mathcal{A}}$ returns 1. Otherwise, it returns 0. So given an input $(\beta, u, \mathbf{r}, k)$ from $\mathcal{D}_{\mathcal{M}}$, $\mathcal{M}'^{\mathcal{A}}$ outputs 1 with probability $\widetilde{\epsilon}_{\mathcal{A}}$.

Algorithm 10: $\mathcal{M}'^{\mathcal{A}}$

Input : Security parameter n, a sample $(\beta, u, \mathbf{r}, k)$.
Output: 0 or 1.

Generate W-OTS$^+$ key pair: $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Kg}(1^n)$ taking bitmasks from $\mathbf{r}$ and a function for chain f_k instead of random ones
Choose random index $\alpha \xleftarrow{\$} \{1, \ldots, l\}$
Obtain modified public key pk' by setting $\mathsf{pk}'_0 = (\mathbf{r}, k)$, $\mathsf{pk}'_i = c_k^{w-1}(\mathsf{sk}_i, \mathbf{r}')$ for $1 \le i \le l, i \ne \alpha$, and $\mathsf{pk}'_\alpha = c_k^{w-1-\beta}(u, \mathbf{r}'_{\beta+1,w-1})$
Run $\mathcal{A}^{\mathsf{Sign}(\mathsf{sk},\cdot)}(\mathsf{pk}')$
if $\mathcal{A}^{\mathsf{Sign}(\mathsf{sk},\cdot)}(\mathsf{pk}')$ queries to sign message M **then**
    Compute $B = (b_1, \ldots, b_l)$ which corresponds to M
    if $b_\alpha < \beta$ **then**
        return 0
    Generate signature σ of M with respect to the modified public key:
        i. Run $\sigma = (\sigma_1, \ldots, \sigma_l) \leftarrow \mathsf{Sign}(M, \mathsf{sk}, \mathbf{r}')$
        ii. Set $\sigma_\alpha = c_k^{b_\alpha - \beta}(y_c, \mathbf{r}'_{\beta+1,w-1})$
    Reply to the query using σ
if $\mathcal{A}^{\mathsf{Sign}(\mathsf{sk},\cdot)}(\mathsf{pk}')$ returns valid (σ', M') **then**
    Compute $B' = (b'_1, \ldots, b'_l)$ which corresponds to M'
    if $b'_\alpha \ge \beta$ **then**
        return 0
    else
        return 1

Let us consider the behavior of $\mathcal{M}'^{\mathcal{A}}$ given an input from $\mathcal{D}_{\mathsf{Kg}}$. In this case $\mathcal{A}$ obtains a fair W-OTS$^+$ public key. The probability that $\mathcal{M}'^{\mathcal{A}}$ outputs 1 is thus given by

$$\begin{aligned}\widehat{\epsilon}_{\mathcal{A}} &\equiv \Pr[b_\alpha \ge \beta \wedge \text{“Forgery is valid”} \wedge b'_\alpha < b_\alpha] \\ &= \epsilon_{\mathcal{A}} \cdot \Pr[b_\alpha \ge \beta \wedge b'_\alpha < b_\alpha | \text{“Forgery is valid”}] \\ &\ge \epsilon_{\mathcal{A}} \cdot \Pr[b_\alpha = \beta \wedge b'_\alpha < b_\alpha | \text{“Forgery is valid”}]. \quad (19)\end{aligned}$$

Here we used the fact that in the considered case $\Pr[\text{“Forgery is valid”}] = \epsilon_{\mathcal{A}}$. Then we can write

$$\begin{aligned}&\Pr[b_\alpha = \beta \wedge b'_\alpha < b_\alpha | \text{“Forgery is valid”}] \\ &= \Pr[b_\alpha = \beta | \text{“Forgery is valid”}] \cdot \Pr[b'_\alpha < b_\alpha | b_\alpha = \beta \wedge \text{“Forgery is valid”}] \quad (20)\end{aligned}$$

and consider each term of the RHS in detail. Let X be a random variable equal to a number of elements in the requested W-OTS$^+$ signature σ which lie above the zero level, which is conditioned by the fact the the forgery produced by $\mathcal{A}$ is valid (if one gives σ to $\mathcal{A}$). More formally we define X as

follows:

$$X = |\{i : 1 \le i \le l, b_i > 0\}| \quad \text{conditioned by "Forgery is valid".} \tag{21}$$

Since α and β are chosen at random from the sets $\{1, \ldots, l\}$ and $\{1, \ldots, w-1\}$ we have

$$\Pr[b_\alpha = \beta | \text{"Forgery is valid"}] = \frac{X}{l(w-1)} > \frac{X}{lw}. \tag{22}$$

Then, since the forged message M' has at least one element in its signature σ' which went down through its chain compared to the signature σ, and this element is certainly among X elements, we have

$$\Pr[b'_\alpha < b_\alpha | b_\alpha = \beta \wedge \text{"Forgery is valid"}] \ge \frac{1}{X}. \tag{23}$$

Taking together Eqs. (19), (22), (23), and putting the result into Eq. (19) we obtain

$$\widehat{\epsilon}_{\mathcal{A}} > \frac{\epsilon_{\mathcal{A}}}{lw}. \tag{24}$$

By the definition, the advantage of distinguishing $\mathcal{D}_{\mathcal{M}}$ $\mathcal{D}_{\mathsf{Kg}}$ by $\mathcal{M}'^{\mathcal{A}}$ is given by

$$\mathrm{Adv}_{\mathcal{D}_{\mathcal{M}},\mathcal{D}_{\mathsf{Kg}}}(\mathcal{M}'^{\mathcal{A}}) = |\widetilde{\epsilon}_{\mathcal{A}} - \widehat{\epsilon}_{\mathcal{A}}|. \tag{25}$$

Using the obtained bound (24) and expanding absolute value in Eq. (25) we come to the following upper bound on $\epsilon_{\mathcal{A}}$:

$$\epsilon_{\mathcal{A}} < lw \cdot \left(\mathrm{Adv}_{\mathcal{D}_{\mathcal{M}},\mathcal{D}_{\mathsf{Kg}}}(\mathcal{M}'^{\mathcal{A}}) + \widetilde{\epsilon}_{\mathcal{A}}\right). \tag{26}$$

The remaining step is to derive an upper bound of $\mathrm{Adv}_{\mathcal{D}_{\mathcal{M}},\mathcal{D}_{\mathsf{Kg}}}(\mathcal{M}'^{\mathcal{A}})$ using the maximal possible insecurity level of the UD property. For this purpose we employ the hybrid argument method. First, we note that

$$\mathrm{Adv}_{\mathcal{D}_{\mathcal{M}},\mathcal{D}_{\mathsf{Kg}}}(\mathcal{M}'^{\mathcal{A}}) = \sum_{\beta'=1}^{w-1} \frac{1}{w-1} \mathrm{Adv}_{\mathcal{D}_{\mathcal{M}}^{\beta=\beta'},\mathcal{D}_{\mathsf{Kg}}^{\beta=\beta'}}(\mathcal{M}'^{\mathcal{A}}), \tag{27}$$

where $\mathcal{D}_{\mathcal{M}}^{\beta=\beta'}$ and $\mathcal{D}_{\mathsf{Kg}}^{\beta=\beta'}$ denote distributions with fixed first subelement $\beta = \beta'$. Expression (27) leads to the fact that there must exist at least one value $\beta^\star$ such that

$$\mathrm{Adv}_{\mathcal{D}_{\mathcal{M}}^{\beta=\beta^\star},\mathcal{D}_{\mathsf{Kg}}^{\beta=\beta^\star}}(\mathcal{M}'^{\mathcal{A}}) \ge \mathrm{Adv}_{\mathcal{D}_{\mathcal{M}},\mathcal{D}_{\mathsf{Kg}}}(\mathcal{M}'^{\mathcal{A}}). \tag{28}$$

Then we define a sequence of distributions $\{\mathcal{H}_i\}_{i=0}^{\beta^\star}$ over $\{1, \ldots, w-1\} \times \{0,1\}^n \times \{0,1\}^{n\times(w-1)} \times \mathcal{K}_n$, such that an element $(\beta, u, \mathbf{r}, k)$ is generated from $\mathcal{H}_i$ by setting

$$\beta = \beta^\star, \quad x \xleftarrow{\$} \{0,1\}^n, \quad u = c_k^{\beta^\star - i}(x, \mathbf{r}_{j+1,w-1}), \tag{29}$$

and sampling $\mathbf{r}$ and k uniformly at random from the corresponding spaces. One can see that $\mathcal{H}_0$ and $\mathcal{H}_{\beta^\star}$ coincide with $\mathcal{D}_{\mathsf{Kg}}^{\beta=\beta^\star}$ and $\mathcal{D}_{\mathcal{M}}^{\beta=\beta^\star}$, correspondingly. So, Eq. (28) can be rewritten as follows:

$$\mathrm{Adv}_{\mathcal{H}_{\beta^\star},\mathcal{H}_0}(\mathcal{M}'^{\mathcal{A}}) \geq \mathrm{Adv}_{\mathcal{D}_{\mathcal{M}},\mathcal{D}_{\mathsf{Kg}}}(\mathcal{M}'^{\mathcal{A}}). \tag{30}$$

The triangular inequality yields the fact that there must exist two consecutive distributions $\mathcal{H}_{i^\star}$ and $\mathcal{H}_{i^\star+1}$ with $0 \leq i^\star < \beta^\star$ such that

$$\mathrm{Adv}_{\mathcal{H}_{i^\star},\mathcal{H}_{i^\star+1}}(\mathcal{M}'^{\mathcal{A}}) \geq \frac{1}{\beta^\star}\mathrm{Adv}_{\mathcal{D}_{\mathcal{M}},\mathcal{D}_{\mathsf{Kg}}}(\mathcal{M}'^{\mathcal{A}}) > \frac{1}{w}\mathrm{Adv}_{\mathcal{D}_{\mathcal{M}},\mathcal{D}_{\mathsf{Kg}}}(\mathcal{M}'^{\mathcal{A}}). \tag{31}$$

We are ready to construct our final machine $\mathcal{B}^{\mathcal{M}'^{\mathcal{A}}}$, shown in Algorithm 11, which employs $\mathcal{M}'^{\mathcal{A}}$ to break the UD property.

Algorithm 11: $\mathcal{B}^{\mathcal{M}'^{\mathcal{A}}}$

Input : Security parameter n, a sample (u, k).
Output: 0 or 1.

1 Generate $\mathbf{r} \xleftarrow{\$} \{0,1\}^{n(w-1)}$
2 Input n and $(\beta^\star, c_k^{\beta^\star-(i^\star+1)}(u, \mathbf{r}_{i^\star+1,w-1}), \mathbf{r}, k)$ into $M'^{\mathcal{A}}$
3 **return** the result from $M'^{\mathcal{A}}$

One can see that

$$\mathrm{Adv}_{\mathcal{D}_{\mathrm{UD},\mathcal{U}},\mathcal{D}_{\mathrm{UD},\mathcal{F}_n}}(\mathcal{B}^{\mathcal{M}'^{\mathcal{A}}}) = \mathrm{Adv}_{\mathcal{H}_{i^\star},\mathcal{H}_{i^\star+1}}(\mathcal{M}'^{\mathcal{A}}), \tag{32}$$

since the input to $\mathcal{M}'^{\mathcal{A}}$ with (u, k) from $\mathcal{D}_{\mathrm{UD},\mathcal{U}}$ is equivalent to a sample from $\mathcal{H}_{i^\star+1}$, while this input to $\mathcal{M}'^{\mathcal{A}}$ with (u, k) from $\mathcal{D}_{\mathrm{UD},\mathcal{F}_n}$ is equivalent to a sample from $\mathcal{H}_{i^\star}$. Indeed,

$$c_k^{\beta^\star-(i^\star+1)}(f_k(x), \mathbf{r}_{i^\star+1,w-1}) = c_k^{\beta^\star-i^\star}(x \oplus r_{i^\star}, \mathbf{r}_{i^\star,w-1}) \tag{33}$$

and $x \oplus r_{i^\star}$ is indistinguishable from the uniformly random string. At the same time, we have

$$\mathrm{Adv}_{\mathcal{D}_{\mathrm{UD},\mathcal{U}},\mathcal{D}_{\mathrm{UD},\mathcal{F}_n}}(\mathcal{B}^{\mathcal{M}'^{\mathcal{A}}}) \leq \mathrm{InSec}^{\mathrm{UD}}(\mathcal{F}_n; \widetilde{t}). \tag{34}$$

The runtime bound $\widetilde{t} = t + 3lw + w - 2$ is obtained as sum of time t required for $\mathcal{A}$, at most $3lw$ calculations of f_k required in Kg, Sign, and Vf used in $\mathcal{M}'^{\mathcal{A}}$, and at most $w-2$ calculations of f_k, while preparing input for $\mathcal{M}'^{\mathcal{A}}$ in $\mathcal{B}^{\mathcal{M}'^{\mathcal{A}}}$ (line 2 in Algorithm 11) and preparing αth chain in $\mathcal{M}'^{\mathcal{A}}$ (line 3 in Algorithm 10) (the total number of f_k evaluations is given by $w - 1 - (i^\star + 1) \leq w - 2$).

By combining together Eqs. (31), (32), and (34) we obtain

$$\mathrm{Adv}_{\mathcal{D}_{\mathcal{M}},\mathcal{D}_{\mathsf{Kg}}}(\mathcal{M}'^{\mathcal{A}}) < w \cdot \mathrm{InSec}^{\mathrm{UD}}(\mathcal{F}_n; \widetilde{t}). \tag{35}$$

Then putting this result into Eq. (26) we arrive at

$$\epsilon_{\mathcal{A}} < lw \cdot \left(w \cdot \mathrm{InSec}^{\mathrm{UD}}(\mathcal{F}_n; \widetilde{t}) + \widetilde{\epsilon}_{\mathcal{A}}\right) \tag{36}$$

Finally, taking into account Eq. (18) we obtain the desired upper bound. □

Remark 1. *One can note that the bound* $\widetilde{t} = t + 3lw + w - 2$ *can be tightened at least to* $\widetilde{t} = t + 3lw$ *by firstly choosing the value* α *and then removing calculation of* α*th chain within* Kg *used in* $\mathcal{M}^{\mathcal{A}}$ *and* $\mathcal{M}'^{\mathcal{A}}$*. However, it has almost no practical value since usually is assumed that* $t \gg 4lw$*.*

3.2 Difference from the previous version of the proof

Here we point out main differences between our security proof and the original proof from Ref. [8] that contains a slightly different security bound, namely:

$$\begin{aligned}\mathrm{InSec}^{\mathrm{EU-CMA}}(\text{W-OTS}^+(1^n, w, m); t, 1) \\ \leq wl \cdot \max\left\{\mathrm{InSec}^{\mathrm{OW}}(\mathcal{F}_n; t'), w \cdot \mathrm{InSec}^{\mathrm{SPR}}(\mathcal{F}_n; t')\right\} + \\ w \cdot \mathrm{InSec}^{\mathrm{UD}}(\mathcal{F}_n; t^\star),\end{aligned} \tag{37}$$

where $t' = t + 3lw$ and $t^\star = t + 3lw + w - 1$.

First of all, during the discussion of $\mathcal{M}^{\mathcal{A}}$, that is the same in both proofs, it was stated that $\Pr[b_\alpha = \beta] \geq \frac{1}{w}$, motivated by the fact that β is chosen uniformly at random (see p. 181 of [8]). However, as we discussed in our proof, $\mathcal{A}$ may reveal the chain containing challenges, and also may always ask to sign a message with $b_\alpha = 0$ thus making $\Pr[b_\alpha = \beta] = 0$.

In the proof of [8] it is stated that $\Pr[b'_\alpha < \beta | \text{“Forgery is valid”} \wedge b_\alpha = \beta] \geq l^{-1}$. This is also may not be correct if $\mathcal{A}$ is able to reveal the chain containing challenges and, e.g., make forgery only with $b'_\alpha = \beta$. Actually, accounting a possibility of hostile behavior of $\mathcal{A}$ forces us to introduce the “Forgery is fortunate” event and bound its probability by employing $\mathrm{InSec}^{\mathrm{UD}}(\ldots)$. We note that our treatment also gives a different factor before the term $\mathrm{InSec}^{\mathrm{UD}}(\ldots)$.

Moreover, in Ref. [8] the obtained bound contains $\max\{\mathrm{InSec}^{\mathrm{OW}}(\ldots), w\mathrm{InSec}^{\mathrm{SPR}}(\ldots)\}$ instead of $\mathrm{InSec}^{\mathrm{OW}}(\ldots) + w \cdot \mathrm{InSec}^{\mathrm{SPR}}(\ldots)$. Perhaps, it appeared by putting multiples p and $(1-p)$ on the opposite side of inequalities corresponded to Eq. (16) and Eq. (17) of the present paper.

Finally, the used different runtime bounds t' and $t^\star$ for breaking OW/SPR and UD of $\mathcal{F}_n$, however, as it is shown above they can be considered to be the same.

Anyway, as we demonstrate below, both expressions (14) and (37) provide close levels of security. Moreover, we note that the security level of W-OTS$^+$ used in the security proof of SPHINCS coincides with the derived expression (14) (see $\#_{ots}$ term on page 382 of [9]).

3.3 Security level

Given results of the Theorem 1, we are able to compute the security level against classical and quantum attacks. Following reasoning from Refs. [8, 19], we say the scheme has security level b if a successful attack is expected to require 2^{b-1} evaluations of functions from $\mathcal{F}_n$. We calculate lower bound on b by considering the inequality $\mathrm{InSec}^{\mathrm{EU-CMA}}(\text{W-OTS}^+(1^n, w, m); t, 1) \geq 1/2$. We assume that

$$\mathrm{InSec}^{\mathrm{OW}}(\mathcal{F}(n); t) = \mathrm{InSec}^{\mathrm{SPR}}(\mathcal{F}(n); t) = \mathrm{InSec}^{\mathrm{UD}}(\mathcal{F}(n); t) = \frac{t}{2^n} \qquad (38)$$

for brute force search attacks with classical computer [3], and

$$\mathrm{InSec}^{\mathrm{OW}}(\mathcal{F}(n); t) = \mathrm{InSec}^{\mathrm{SPR}}(\mathcal{F}(n); t) = \mathrm{InSec}^{\mathrm{UD}}(\mathcal{F}(n); t) = \frac{t}{2^{n/2}} \qquad (39)$$

for attack with quantum computer using Grover's algorithm [4]. We also assume that $t \gg 4lw$, so all runtime bounds used in (14) and (37) are the same: $\widetilde{t} \approx t' \approx t^\star \approx t$. The results of comparison are shown in Table 1. The new bound is smaller the previous one by $\log \frac{l(2w+1)}{lw+1} \approx 1$ bit for typical parameter values $w = 16$ and $l = 67$.

	Bound from [8]	Bound from present work
Classical attacks	$b > n - \log w - \log(lw + 1)$	$b > n - \log(lw) - \log(2w + 1)$
Quantum attacks	$b > \frac{n}{2} - \log w - \log(lw + 1)$	$b > \frac{n}{2} - \log(lw) - \log(2w + 1)$

Table 1: Comparison of security levels for the W-OTS$^+$ scheme.

4 Conclusion and outlook

Here we summarize the main results of our work. We have recapped the security analysis of the W-OTS$^+$ signature presented in Ref. [8], and pointed out some of its flaws. Although the updated security level almost coincides with the one from Ref. [8], we believe that our contribution is important for a fair justification of the W-OTS$^+$ security.

We note that a security analysis of the many-times stateless signature scheme SPHINCS$^+$, which uses W-OTS$^+$ a basic primitive and which was

submitted to NIST process [11], originally was based on another approach for evaluating the security level [10]. However, it was discovered that the employed security analysis has some critical flaws (see C.J. Peikert official comment on Round 1 SPHINCS$^+$ submission [20]).

Recently, a new approach for the security analysis of hash-based signature was introduced [14]. It suggests a novel property of hash functions, namely the decisional second-preimage resistance, and therefore requires an additional deep comprehensive study.

References

[1] Shor P.W., "Polynomial-time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer", *SIAM J. Comput.*, **26** (1997), 1484–1509.

[2] Bernstein D.J., Lange T., "Post-quantum Cryptography", *Nature*, **549** (2017), 188–194.

[3] Dods C., Smart N. P., Stam M., "Hash Based Digital Signature Schemes", *Cryptography and Coding*, 2005, 96–115.

[4] Grover L.K., "A Fast Quantum Mechanical Algorithm for Database Search,", Proceedings of 28th Annual ACM Symposium on the Theory of Computing (New York, USA), 1996, 212.

[5] Cooper D.A., Apon D.C., Dang Q.H., Davidson M.S., Dworkin M.J., Miller C.A., *Recommendation for Stateful Hash-Based Signature Schemes*, 2019, `https://doi.org/10.6028/NIST.SP.800-208-draft`.

[6] D. McGrew M. Curcio and S. Fluhrer, "Hash-based signatures", 2018, `https://datatracker.ietf.org/doc/draft-mcgrew-hash-sigs/`.

[7] A. Hülsing D. Butin S. Gazdag J. Rijneveld and A. Mohaisen, "XMSS: eXtended Merkle Signature Scheme", 2018, `https://datatracker.ietf.org/doc/rfc8391/`.

[8] Hülsing A., "W-OTS+ – Shorter Signatures for Hash-Based Signature Schemes", *Progress in Cryptology – AFRICACRYPT 2013*, 2013, 173–188.

[9] Bernstein D.J., Hopwood D., Hülsing A., Lange T., Niederhagen R., Papachristodoulou L., Schneider M., Schwabe P., Wilcox-O'Hearn Z., "SPHINCS: practical stateless hash-based signatures", EUROCRYPT 2015: Advances in Cryptology – EUROCRYPT 2015, 368–397.

[10] Hülsing A., Rijneveld J., Song F., "Mitigating Multi-target Attacks in Hash-Based Signatures", *Public-Key Cryptography – PKC 2016*, 2016, 387–416.

[11] Daniel J. Bernstein, Christoph Dobraunig, Maria Eichlseder, et al., "SPHINCS$^+$ Submission to the NIST post-quantum project", 2017, `https://sphincs.org/resources.html`.

[12] Aumasson J.-P., Endignoux G., "Gravity-SPHINCS", 2017, `https://github.com/gravity-postquantum/gravity-sphincs`.

[13] Bernstein D.J., Hülsing A., Kölbl, Niederhagen R., Rijneveld J., "The SPHINCS$^+$ Signature Framework", CCS '19: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, 2019, 2129–2146.

[14] Daniel J. Bernstein and Andreas Hülsing, "Decisional second-preimage resistance: When does SPR imply PRE?", 2019, `https://eprint.iacr.org/2019/492`.

[15] Merkle R. C., "A Digital Signature Based on a Conventional Encryption Function", *Advances in Cryptology — CRYPTO '87*, 1988, 369–378.

[16] Even S., Goldreich O., Micali S., "On-Line/Off-Line Digital Signatures", *Advances in Cryptology — CRYPTO' 89 Proceedings*, 1990, 263–275.

[17] Lamport L., "Constructing digital signatures from a one way function", 1979.

[18] Buchmann J., Dahmen E., Ereth S., Hülsing A., Rückert M., "On the Security of the Winternitz One-Time Signature Scheme", *Progress in Cryptology – AFRICACRYPT 2011*, 2011, 363–378.

[19] Lenstra A.K., "Key lengths", *Contribution to The Handbook of Information Security*, 2004.

[20] Christopher J. Peikert, "Official Comments - SPHINCS$^+$", 2018, `https://csrc.nist.gov/Projects/post-quantum-cryptography/Round-1-Submissions`.