Exact maximum expected differential and linear probability for 2-round Kuznyechik (Extended Abstract)

Vitaly Kiryukhin

JSC «InfoTeCS», Moscow, Russia
Vitaly.Kiryukhin@infotecs.ru

Abstract

This paper presents the complete description of the best differentials and linear hulls in 2-round Kuznyechik. We proved that 2-round MEDP $= 2^{-86.66\ldots}$, MELP $= 2^{-76.739\ldots}$. A comparison is made with similar results for the AES cipher.

Keywords: Kuznyechik, LSX, MDS codes, differential cryptanalysis, linear cryptanalysis, MEDP, MELP.

1 Introduction

This paper presents the results of the development of low-complexity algorithms, that will allow to find the complete description of the best differential trails, differentials, linear characteristics, linear hulls and *exact* values of maximum expected differential and linear probability (MEDP, MELP) for 2-round Kuznyechik.

We proved that 2-round MEDP $= 2^{-86.66\ldots}$, MELP $= 2^{-76.739\ldots}$.

A comparison is made with similar cryptanalysis results for the AES cipher [1].

The main focus will be on the differential method. The results of the search for linear characteristics will be obtained in a similar way, due to the existence well-known duality between differential cryptanalysis and linear cryptanalysis [2].

2 Basic information

Kuznyechik block cipher [3] consists of a sequence of 9 rounds and a post-whitening key addition. Each round contains three operations:

X – modulo 2 addition of an input block with an iterative key;

S – parallel application of a fixed bijective substitution to each byte of the block;

L – linear transformation which is defined as a LFSR over $GF(2^8)$. It can be represented as multiplication by the matrix $\mathbb{L}$ over $GF(2^8)$.

The block size is 128 bits ($n = 16$ bytes).

A 2-round differential trail can be represented as the following scheme:

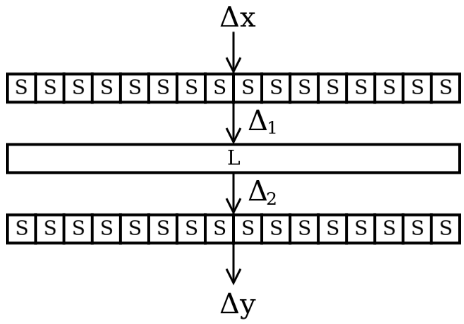

Figure 1: 2-round differential trail

$\Delta x = (x_1, ..., x_n)$ – the difference of input blocks in byte representation,

$\Delta_1 = (\alpha_1, ..., \alpha_n)$ – the difference of blocks after the nonlinear transformation on the first round,

$\Delta_2 = (\beta_1, ..., \beta_n) = (\alpha_1, ..., \alpha_n)\mathbb{L}$ – the difference of blocks after the linear transformation (matrix multiplication in row-by-row representation),

$\Delta y = (y_1, ..., y_n)$ – the difference of blocks after the nonlinear transformation on the second round.

Note that due to «linearity» and «invertibility» the linear transformation on the second round can be omitted without loss of generality.

The nonlinear transformation of each S-box is characterized by a matrix of transition probabilities (Differential Distribution Table). DDT is the set of local difference characteristics:

$$P(\alpha \to \beta) = \Pr(S(\chi \oplus \alpha) \oplus S(\chi) = \beta),\ \alpha, \beta, \chi \in \{0,1\}^8, \qquad (1)$$

where χ is a uniformly distributed random variable. S-box with nonzero input difference $\alpha \neq 0$ is called active.

2-round differential trail $\Delta x \to \Delta_1 \to \Delta_2 \to \Delta y$ is a random variable, that has a probability (*EDCP* [1])

$$P(\Delta x \to \Delta_1 \to \Delta_2 \to \Delta y) = \left(\prod_{i=1}^{n} P(x_i \to \alpha_i)\right)\left(\prod_{i=1}^{n} P(\beta_i \to y_i)\right). \quad (2)$$

The best differential trail has probability

$$P_{best}^{trail} = P_{best}(\Delta x \to \Delta_1 \to \Delta_2 \to \Delta y) =$$
$$= \max_{(\Delta x, \Delta_1, \Delta_2, \Delta y) \setminus (0,0,0,0)} P(\Delta x \to \Delta_1 \to \Delta_2 \to \Delta y).$$

Differential is the set of all differential trails that have the same Δx and Δy.

Differential is characterized by the probability (*EDP* [1])

$$P(\Delta x \to \Delta y) = \sum_{i=1}^{T}\left(\left(\prod_{j=1}^{n} P(x_j \to \alpha_j^{(i)})\right)\left(\prod_{j=1}^{n} P(\beta_j^{(i)} \to y_j)\right)\right), \quad (3)$$

where T is the number of the differential trails in the differential.

The best differential has probability (*MEDP* [1]):

$$P_{best}^{diff} = P_{best}(\Delta x \to \Delta y) = \max_{(\Delta x, \Delta y) \setminus (0,0)} P(\Delta x \to \Delta y)$$

Our first goal is to find the most probable differential trail – the best differential trail.

Matrix $\mathbb{L}$ is part of the matrix $\mathbb{G} = \mathbb{E}|\mathbb{L}$. $\mathbb{G}$ is the generator matrix of the MDS-code $(32, 16, 17)$ over $GF(2^8)$. Thus, the minimum possible total weight of vectors Δ_1 and Δ_2 is equal to the minimum code distance $d = 17$. We will start searching for the most probable differential trail by finding all minimum byte weight codewords in $\mathbb{G}$.

3 Algorithm for finding codewords with the smallest byte weight

Let (t, r) such, that $t + r = n + 1$, $t > 0$, $r > 0$. Fix $k_1, \dots, k_t, m_1, \dots, m_r$ – locations of non-zero elements in the vectors $\Delta_1 = (\alpha_1, ..., \alpha_n)$ and $\Delta_2 = (\beta_1, ..., \beta_n)$ accordingly. Let's present the transformation $\Delta_1\mathbb{L} = \Delta_2$ as a system of equations. Select the subsystem $\mathbb{S}_{n-r,t}$ in the system $\Delta_1\mathbb{L} = \Delta_2$: $(\alpha_{k_1}, \dots, \alpha_{k_t}) \cdot \mathbb{S}_{n-r,t} = (\underbrace{0, \dots, 0}_{n-r})$. Solve the subsystem $\mathbb{S}_{n-r,t}$. The set of solutions is $(\alpha^{(i)}_{k_1}, \ldots, \alpha^{(i)}_{k_t})$, $i = \overline{1, 255}$. Hence we have the set of $\Delta_1^{(i)}$ and the set of $\Delta_2^{(i)} = \Delta_1^{(i)}\mathbb{L}$, $i = \overline{1, 255}$.

Let's denote these sets of solutions

$$M^{(n+1)}(k_1, \ldots, k_t, m_1, \ldots, m_r) = (\alpha^{(i)}_{k_1}, \ldots, \alpha^{(i)}_{k_t}, \beta^{(i)}_{m_1}, \ldots, \beta^{(i)}_{m_r}),\ i = \overline{1, 255}. \tag{4}$$

The union of such sets is the set

$$M^{(n+1)} = \bigcup_{(k_1,\ldots,k_t,m_1,\ldots,m_r)} M^{(n+1)}(k_1, \ldots, k_t, m_1, \ldots, m_r)$$

of all code vectors of minimum weight $n + 1$. The cardinality of the set $M^{(n+1)}$ is equal to $255 \cdot \sum_{(t,r):t+r=n+1} \binom{n}{t}\binom{n}{r} = 255 \cdot \binom{2n}{n+1}$. Note, that the same expression for the number of codewords of minimal weight is obtained in [5].

Pseudocode of the algorithm is presented in Appendix E.

4 Algorithm for finding the best differential trail

In general, we consider differential trails for 2 rounds

$$\Delta x \to \Delta_1 \to \Delta_2 \to \Delta y.$$

We start with differential trails containing the minimum number of active S-boxes (minimal weight of Δ_1 and Δ_2).

To simplify the notation we denote $(\Delta_1, \Delta_2) = (\alpha_{k_1}, \ldots, \alpha_{k_t}, \beta_{m_1}, \ldots, \beta_{m_r})$, $t + r \geq d = 17$. Coordinates equal to zero are omitted in notation.

$$P_{max}(\Delta_1, \Delta_2) = \left(\prod_{j=1}^{t} \max_x P(x \to \alpha_{k_j})\right)\left(\prod_{j=1}^{r} \max_y P(\beta_{m_j} \to y)\right)$$

is the maximum probability of differential trail with a fixed vector (Δ_1, Δ_2). Then the most probable differential trail $\Delta x \to \Delta_1 \to \Delta_2 \to \Delta y$ has the probability:

$$P^{trail}_{best} = \max_{(\Delta_1,\Delta_2)\setminus(0,0)} P_{max}(\Delta_1, \Delta_2).$$

Let the vector (Δ_1, Δ_2) has a weight $n + 1$:

$$P^{trail}_{best} \geq \max_{(\Delta_1,\Delta_2)\in M^{(n+1)}} P_{max}(\Delta_1, \Delta_2).$$

Two sets of differential trails were found in $M^{(n+1)}$. Each trail in both sets has a maximum probability:

$$\max_{(\Delta_1,\Delta_2)\in M^{(n+1)}} P_{max}(\Delta_1,\Delta_2) = \left(\frac{8}{256}\right)^{13}\left(\frac{6}{256}\right)^{4} = 2^{-86.66\ldots}.$$

The trails in the set have the same inner part (Δ_1, Δ_2). There are no other trails that would have a maximum probability.

The found differential trails are presented in Appendix A.

Lemma 1. *Let $\Delta x \to \Delta_1 \to \Delta_2 \to \Delta y$ be the differential trail in 2-round Kuznyechik. Let $P(\Delta x \to \Delta_1 \to \Delta_2 \to \Delta y)$ be maximal among all trails. Then the weight (Δ_1, Δ_2) is equal to $n + 1 = 17$.*

Proof. One can see that the estimate

$$P(\Delta x \to \Delta_1 \to \Delta_2 \to \Delta y) \le \left(\max_{(\alpha,\beta)\setminus(0,0)} P(\alpha \to \beta)\right)^{w}. \tag{5}$$

is true for any differential trail $\Delta x \to \Delta_1 \to \Delta_2 \to \Delta y$, $\|\Delta_1\| + \|\Delta_2\| = w = t + r$.

In the case of Kuznyechik, $\max\limits_{(\alpha,\beta)\setminus(0,0)} P(\alpha \to \beta) = \left(\frac{8}{256}\right)$. Then for any $w \ge 18$ it holds that:

$$P(\Delta x \to \Delta_1 \to \Delta_2 \to \Delta y) \le \left(\frac{8}{256}\right)^{w} \le$$

$$\le \left(\frac{8}{256}\right)^{18} < \max_{(\Delta_1,\Delta_2)\in M^{(n+1)}} P_{max}(\Delta_1,\Delta_2) = 2^{-86.66\ldots}.$$

Hence $P^{trail}_{best} = 2^{-86.66\ldots}$. Lemma 1 is proved. □

5 Algorithm for finding the best differential

Suppose that the best differential will also be achieved on a configuration containing the minimum number $w = n + 1 = 17$ of active S-boxes.

Each subset $M^{(n+1)}(k_1, \ldots, k_t, m_1, \ldots, m_r)$ contains exactly 255 code vectors. The sets $k_1, \ldots, k_t$ and $m_1, \ldots, m_r$ specify the positions of active S-boxes. Hence the differential $\Delta x \to \Delta y$ contains trails from *only one* subset $M^{(n+1)}(k_1, \ldots, k_t, m_1, \ldots, m_r)$. Consequently, in expression (3) $T = 255$.

Consider an algorithm that allows you to get rid of the exhaustive search. It is based on the «pruning» of the branches of the search tree by using the constructed upper bounds.

In the previous paragraph, the exact value of the best differential trail is given $P_{best}^{trail} = 2^{-86.66\ldots}$. This probability is the lower bound for the probability of the best differential. It is always possible to construct a differential, consisting of one best trail $P_{best}^{diff} \geq P_{best}^{trail}$. We will use the probability $P_{est}^{diff} = P_{best}^{trail}$ as a threshold value.

5.1 Algorithm for calculating the upper bound of the differential

Let a subset of codewords (4) is given. Calculate the upper bound of the differential.

Fix $u \leq t$, $v \leq r$. Select $t-u$ coordinates α and $r-v$ coordinates β in the equation (4):

$$part^{(i)} = (\alpha_{k_1}^{(i)}, \ldots, \alpha_{k_{t-u}}^{(i)}, \beta_{m_1}^{(i)}, \ldots, \beta_{m_{r-v}}^{(i)}),\ i = \overline{1,255}.$$

For all $i = \overline{1,255}$ we obtain an easily computable upper bound for the «part» of the differential trail

$$P(\Delta x \to part^{(i)} \to \Delta y) \leq$$
$$\leq \left(\prod_{j=1}^{t-u} \max_x P(x \to \alpha_{k_j}^{(i)})\right)\left(\prod_{j=1}^{r-v} \max_y P(\beta_{m_j}^{(i)} \to y)\right).$$

Let's order these estimates in descending order.

We will construct for each x (and y) the sequence of transition probabilities. Let's use the S-box transition probability matrix (DDT):

$$P(x \to \alpha^{(1,x)}) \geq P(x \to \alpha^{(2,x)}) \geq \ldots \geq P(x \to \alpha^{(255,x)}),\ x = \overline{1,255}, \quad (6)$$

$$P(\beta^{(1,y)} \to y) \geq P(\beta^{(2,y)} \to y) \geq \ldots \geq P(\beta^{(255,y)} \to y),\ y = \overline{1,255}. \quad (7)$$

$$X^{(q)} = \max_x P(x \to \alpha^{(q,x)}),\ Y^{(q)} = \max_y P(\beta^{(q,y)} \to y). \quad (8)$$

Consider the differential (3). Let the summands be ordered in descending order. Then

$$P(\Delta x \to \Delta y) \leq \min_{u,v}\left(\sum_{q=1}^{255}\left(X^{(q)}\right)^u\left(Y^{(q)}\right)^v\left(P(\Delta x \to part^{(q)} \to \Delta y)\right)\right). \quad (9)$$

If the resulting upper bound (9) is less than the threshold P_{est}^{diff}, then the subset is no longer considered.

In practice, the values u and v are selected experimentally depending on the cipher substitution. For Kuznyechik $u = v = 2$ are close to optimal parameters. For such values, approximately $\frac{9}{10}$ subsets are excluded from being considered.

5.2 Algorithm for constructing the differential

Suppose that for some subset $M^{(n+1)}(k_1, \ldots, k_t, m_1, \ldots, m_r)$ the estimate is greater than the threshold value P_{est}^{diff}. Then the following estimate also holds

$$P(\Delta x \to \Delta y) \leq \sum_{i=1}^{255} \left(\prod_{j=1}^{t} \max_x P(x \to \alpha_{k_j}^{(i)}) \prod_{j=1}^{r} \max_y P(\beta_{m_j}^{(i)} \to y) \right). \quad (10)$$

We will sequentially search through possible non-zero values $x_{k_1}, \ldots, x_{k_t}$ and $y_{m_1}, \ldots, y_{m_r}$. The maximum values $\max\limits_x P(x \to \alpha_{k_j}^{(i)})$ (and $\max\limits_y P(\beta_{m_j}^{(i)} \to y)$) will be replaced by the immediate values $P(x_{k_j} \to \alpha_{k_j}^{(i)})$ ($P(\beta_{m_j}^{(i)} \to y_{m_j})$ accordingly). We will also use the pruning of the branches of the search tree.

Denote

$$P(a_1, a_2, \ldots a_s) = P(x_{k_1} = a_1, x_{k_2} = a_2, \ldots, x_{k_s} = a_s, x_{k_{s+1}}, \ldots, x_{k_t} \to \Delta y),$$

$$P(a_1, a_2, \ldots a_s) \leq$$

$$\leq \sum_{i=1}^{255} \left(\prod_{j=1}^{s} P(a_j \to \alpha_{k_j}^{(i)}) \prod_{j=s+1}^{t} \max_x P(x \to \alpha_{k_j}^{(i)}) \prod_{j=1}^{r} \max_y P(\beta_{m_j}^{(i)} \to y) \right).$$

In the estimate (10), we fix the first factor with the number k_1(the place of the first nonzero element). Let $x_{k_1} = 1$. Then we replace $\max\limits_x P(x \to \alpha_{k_1}^{(i)})$ by $P(1 \to \alpha_{k_1}^{(i)})$. After that we have the estimate $P(a_1 = 1)$. If the estimate $P(a_1 = 1)$ is less than the threshold value P_{est}^{diff}, then we perform a search among the elements $x_{k_1} = 2, 3, \ldots 255$. We will search until the element $x_{k_1} = a_1$, $P(a_1) \geq P_{est}^{diff}$ is found. If such x_{k_1} is not found, then the subset $M^{(n+1)}(k_1, \ldots, k_t, m_1, \ldots, m_r)$ is excluded from being considered.

Let such $x_{k_1} = a_1$ is found. We perform similarly search of the second factor. Consider the bytes $x_{k_2} = 1, 2, \ldots, a_2, \ldots, 255$. Substituting $P(a_2 \to \alpha_{k_2}^{(i)})$ instead of $\max\limits_x P(x \to \alpha_{k_2}^{(i)})$ into the estimate $P(a_1)$. Do this until $a_2 :$ $P(a_1, a_2) \geq P_{est}^{diff}$ is found. If such an element is not found then return to the previous step and try to accomplish this algorithm for the remaining bytes $x_{k_1} > a_1$.

We continue the recursive search. We replace the «s+1»-th factor in

$P(a_1, a_2, \dots a_s)$ with the value $P(a \to \alpha^{(i)}_{k_{s+1}})$, $a = 1, 2, \dots, 255$. Multipliers $\max\limits_y P(\beta^{(i)}_{m_j} \to y)$ are replaced by values $P(\beta^{(i)}_{m_j} \to b)$, $b = 1, 2, \dots, 255$.

If the algorithm substituted all the elements $a_1, \dots, a_t, b_1, \dots, b_r$ and did not reject the subset of codewords, then we obtained an exact estimate $P(a_1, \dots, a_t \to b_1, \dots, b_r)$ and the differential

$$P(\Delta x \to \Delta y) = \sum_{i=1}^{255} \left(\prod_{j=1}^{t} P(a_j \to \alpha^{(i)}_j) \right) \left(\prod_{j=1}^{r} P(\beta^{(i)}_j \to b_j) \right) \geq P^{diff}_{est}. \tag{11}$$

In this case, the value P^{diff}_{est} is updated. We return to the previous step of the algorithm and continue the search in the subset $M^{(n+1)}(k_1, \dots, k_t, m_1, \dots, m_r)$.

The last step of the algorithm: $P^{diff}_{best} = P^{diff}_{est}$.

It was shown that if the number of active substitutions is $n + 1 = 17$, then each best differential contains only one differential trail.

The best differential trails are presented in Appendix A. Pseudocodes of algorithms are presented in Appendix E.

Lemma 2. *Let $\Delta x \to \Delta y$ is the differential in 2-round Kuznyechik. Let $P(\Delta x \to \Delta y)$ be maximal among all differentials. Then the number of active S-boxes in $\Delta x \to \Delta y$ is equal to $n + 1 = 17$.*

The main idea of the proof is to construct an upper bound for the differential $\Delta x \to \Delta y$ containing $n + 2 = d + 1 = 18$ active S-boxes. The upper estimate is built by using: two majorants (8); the MDS code property (byte weight of the sum of codewords is not less than $n + 1$); the rearrangement inequality [6]. The proof of the Lemma is presented in Appendix D.

6 The comparison with AES

The comparison of the results given in this paper for Kuznyechik with the results of the AES cipher analysis is of particular interest [1].

Note the following differences between 2-round versions of the ciphers [3, 4].

Kuznyechik – one MDS-matrix 16×16; pseudorandom, non-analytical S-box; DDT and LAT do not have obvious patterns.

AES – byte permutation layer and four MDS-matrix 4×4; all nontrivial rows and columns in DDT (and LAT) have the same distribution of values.

Differences in linear and non-linear transformations lead to different approaches for calculating differential and linear characteristics.

In the case of AES the actual work is reduced to a single MDS-matrix 4×4. This allows you to construct the entire set of codewords. In the case of Kuznyechik, due to the use of the algorithm (3), only low-weight codewords are iterated over. After that, it is analytically shown that the differential on codewords of greater weight will be worse than the constructed one.

The best differential in AES consists of 75 differential trails. The estimate (6) is used in the construction of the differential. The estimate (10) will be the same for any subset of code words and is therefore not used. MEDP $= 2^{-28.272...}$, MELP $= 2^{-27.287...}$.

The best differential in Kuznyechik consists of a *single* differential trail, but the best linear hull consists of 37 linear characteristics. Due to the algorithm 5.1 it is shown that for the majority of considered subsets of codewords the best differential on them is not achieved. For the remaining subsets, an attempt is made to construct the best differential (algorithm 5.2). This is due to a sequence of transitions from the estimate (10) to the exact value (11). We got: MEDP $= 2^{-86.66...}$, MELP $= 2^{-76.739...}$.

7 Conclusion

The article presented: the algorithm for finding codewords with the small byte weight; algorithms for finding the complete description of the best differential trails (linear characteristics), differentials (linear hulls) in 2-round Kuznyechik.

The best differentials (linear hulls) and their probabilities were found. It was shown that the best differential contains one differential trail; the best linear hull contains 37 linear characteristics (Appendix A and B). We proved that 2-round MEDP $= 2^{-86.66...}$, MELP $= 2^{-76.739...}$. The estimate (5) for a differential trail (linear characteristic) is not achieved for 2-round Kuznyechik.

For any LSX cipher, the N-round MEDP (MELP) is the upper bound for $(N+1)$-round MEDP (MELP). Therefore, the 2-round MEDP (MELP) of Kuznyechik is the upper bound for any larger number of rounds. Obtaining a more precise upper bounds is the subject of further research.

Acknowledgments

The author is very grateful to Igor Arbekov and Anton Naumenko for valuable comments and suggestions on the text of the article.

References

[1] Keliher, L., Sui, J.: *Exact maximum expected differential and linear probability for two-round advanced encryption standard.* IET Information Security 1(2), 53–57 (2007), https://doi.org/10.1049/iet-ifs:20060161

[2] E. Biham, *On Matsui's linear cryptanalysis*, Advances in Cryptology – EUROCRYPT'94, in: Lecture Notes in Comput. Sci., Vol. 950, Springer, Berlin, 1995, pp. 341-355

[3] GOST R 34.12-2015 - National standard of the Russian Federation – Information technology – Cryptographic data security – Block ciphers, 2015

[4] National Institute of Standards and Technology. Advanced Encryption Standard (AES) (FIPS PUB 197), 2001

[5] F.J.MacWilliams, N.J.A.Sloane. *The Theory of Error-Correcting Codes.* North Holland, Amsterdam, 1977

[6] Hardy G.H., Littlewood J.E., Pólya G. *Inequalities, Cambridge Mathematical Library (2. ed.)*, Cambridge: Cambridge University Press, 1952

Appendix

A The best differentials

0 8 0 0 0 0 8 0 8 8 8 0 6 0 0 8	$P(\Delta x \to \Delta_1) \cdot 256$
0019000000002d00b8b8950072000028	Δ_1
2a00000d2337f74d0082a80000009d1b	Δ_2
8 0 0 8 8 8 8 6 0 8 6 0 0 0 6 8	$P(\Delta_2 \to \Delta y) \cdot 256$

Table 1: First optimal internal part$(\Delta_1 \to \Delta_2)$. It generates 2 best differentials.

0 8 8 6 0 8 8 8 0 8 8 0 0 8 0 8	$P(\Delta x \to \Delta_1) \cdot 256$
00a5def70085853700ec0300009c005a	Δ_1
0068ea0d00f700dd006d000000000090	Δ_2
0 6 6 8 0 8 0 6 0 8 0 0 0 0 0 8	$P(\Delta_2 \to \Delta y) \cdot 256$

Table 2: Second optimal internal part$(\Delta_1 \to \Delta_2)$. It generates 24 best differentials.

B Application to Linear Cryptanalysis

There is a certain duality between differential and linear cryptanalysis [2]. It allows us to apply the algorithms described above to calculate linear characteristics.

We make the appropriate substitutions.

Differential probability (1), are replaced by linear probability. DDT is replaced by Linear Approximation Table (LAT). Input/output differences α and β are replaced by input/output masks α' and β' correspondingly.

$$P(\alpha' \to \beta') = (2\mathrm{Pr}(\alpha' \bullet \chi = \beta' \bullet S(\chi)) - 1)^2,\ \alpha', \beta', \chi \in \{0,1\}^8,$$

where $\bullet$ is the inner product over $\{0,1\}$.

By analogy with the differential trail a linear characteristic for 2 rounds is introduced:

$$\mathbf{a} \to \mu_1 \to \mu_2 \to \mathbf{b}.$$

Its probability (by analogy with (2)) is equal to

$$P(\mathbf{a} \to \mu_1 \to \mu_2 \to \mathbf{b}) = \left(\prod_{j=1}^{n} P(\mathbf{a}[j] \to \mu_1[j])\right) \left(\prod_{j=1}^{n} P(\mu_2[j] \to \mathbf{b}[j])\right),$$

where $[j]$ is j-th coordinate of the corresponding vector.

The linear hull (similar to differential) is the set of all linear characteristics having input mask $\mathbf{a}$ and output mask $\mathbf{b}$.

$$(\mathbf{a} \to \mathbf{b}) = \{\mathbf{a} \to \mu_1^{(i)} \to \mu_2^{(i)} \to \mathbf{b},\ i = \overline{1, T}\}.$$

The probability of the linear hull $(\mathbf{a} \to \mathbf{b})$ is equal to:

$$P(\mathbf{a} \to \mathbf{b}) = \sum_{i=1}^{T} \left(\left(\prod_{j=1}^{n} P(\mathbf{a}[j] \to \mu_1^{(i)}[j]) \right) \left(\prod_{j=1}^{n} P(\mu_2^{(i)}[j] \to \mathbf{b}[j]) \right) \right),$$

where T is the number of linear characteristics.

You need to replace all formulas in the sections 4 and 5 according to the above analogies.

The maximum probability of the local linear characteristic of Kuznyechik is

$$P_{max}(\alpha' \to \beta') = \max_{(\alpha',\beta')\setminus(0,0)} P(\alpha' \to \beta') =$$
$$= \left(2\left(\frac{128+28}{256}\right) - 1\right)^2 = \left(\frac{56}{256}\right)^2.$$

The trivial estimate of the two-round linear characteristic is

$$\max_{(\mathbf{a},\mu_1,\mu_2,\mathbf{b})\setminus(0,0,0,0)} P(\mathbf{a} \to \mu_1 \to \mu_2 \to \mathbf{b}) \le \left(\frac{56}{256}\right)^{2\cdot 17} = 2^{-74.549...}.$$

The following results are obtained by executing the algorithms.

The best linear characteristic has a probability equal to

$$\max_{(\mathbf{a},\mu_1,\mu_2,\mathbf{b})\setminus(0,0,0,0)} P(\mathbf{a} \to \mu_1 \to \mu_2 \to \mathbf{b}) =$$
$$= \left(\frac{56}{256}\right)^{2\cdot 10} \left(\frac{52}{256}\right)^{2\cdot 4} \left(\frac{48}{256}\right)^{2\cdot 3} = 2^{-76.739...}.$$

The linear hull $(\mathbf{a} \to \mathbf{b})$ has a nontrivial form and (unlike the differential method) contains 37 linear characteristics $\mathbf{a} \to \mu_1^{(i)} \to \mu_2^{(i)} \to \mathbf{b}$, $i = \overline{1, 37}$. The exact probability of the linear hull is

$$\max_{(\mathbf{a},\mathbf{b})\setminus(0,0)} P(\mathbf{a} \to \mathbf{b}) = 2^{-76.739...} \cdot \left(1 + 2^{-61.407}\right).$$

00	28	00	28	28	00	00	26	00	28	00	00	00	24	00	00	$256 \cdot \frac{\sqrt{P(\mathbf{a} \to \mu_1)}}{2}$
00	6a	00	97	55	00	00	06	00	2f	00	00	00	9a	00	00	μ_1
9f	23	45	ba	5a	b8	00	00	00	00	41	00	4c	87	87	0d	μ_2
24	24	26	26	28	28	00	00	00	00	28	00	28	28	28	26	$256 \cdot \frac{\sqrt{P(\mu_2 \to \mathbf{b})}}{2}$

Table 3: Optimal internal part$(\mu_1 \to \mu_2)$.

The optimal inner part $(\mu_1 \to \mu_2)$ generates the best linear hull.

00	41	00	de	48	00	00	c6	00	5a	00	00	00	9f	00	00	$\mathbf{a}$
9a	38	e8	a2	2f	69	00	00	00	00	6a	00	a7	ab	ab	4b	$\mathbf{b}$

Table 4: The best linear hull $(\mathbf{a}, \mathbf{b})$

The best linear hull $(\mathbf{a}, \mathbf{b})$ consist of 37 linear characteristics $\mathbf{a} \to \mu_1^{(i)} \to \mu_2^{(i)} \to \mathbf{b}$, which are listed below (Table 5 and 6).

i	$\mu_1^{(i)}$	$\mu_2^{(i)}$	$\log_2 P(\mathbf{a} \to \mu_1^{(i)} \to \mu_2^{(i)} \to \mathbf{b})$
1	000800153d0000ef00e2000000020000	7e3ceaad70f7000000005f0048c2c217	-160.980...
2	00200056f40000bc008b000000080000	f9f3abb6c1dd000000007c002209095e	-150.676...
3	003f009b580000d7000d0000000f0000	aea232db819600000000a8003cf1f194	-157.973...
4	0046005e0200002200b7000000110000	27f2f1f753e900000000cd0082adad4f	-158.150...
5	006900ee2000002b007f0000001a0000	752916773291000000000db002fd8d8f4	-155.551...
6	007100d06700001a00580000001c0000	f76c2981a288000000003a00f69e9ecc	-139.633...
7	008d00bd04000045006f000000230000	4ee5e2eea6d2000000009b00055b5b9e	-148.300...
8	00a2000d2600004c00a7000000280000	1c3e056ec7aa000000008d00a82e2e25	-154.032...
9	00bd00c08a00002700210000002f0000	4b6f9c0387e1000000005900b6d6d6ef	-141.656...
10	00cb00e30600006700d8000000320000	69171319f53b00000000560087f6f6d1	-152.336...
11	000a005072000054001a000000420000	61b31086ac4a0000000088009a323212	-160.721...
12	004b00fd9b00002c000c000000520000	935547ea2ff1000000007000df2121af	-148.862...
13	005b00d6e10000f200c9000000560000	6f2c92b1cf1f00000000ce004ea5a580	-156.558...
14	006300be5200007f0065000000580000	149a06f19edb000000005300b5eaeae6	-140.159...
15	007b00801500004e00420000005e0000	96df39070ec200000000b2006cacacde	-146.218...
16	00b000631300002900 9a0000006c0000	ffc82a1efbf900000000e400eb5a5a0f	-155.166...
17	00b70090f8000073003b0000006d0000	2adc8c852bab00000000d1002ce4e4fd	-150.862...
18	00e600166b0000d500e8000000790000	24430eb248fe000000009700f873736f	-147.574...
19	001400a1e40000a90034000000850000	c367200d589500000000110035656524	-152.616...
20	005200ffe600008b0083000000940000	e495d1fa0b7c00000000dc00b7c8c86b	-151.329...
21	006a00975500000 6002f0000009a0000	9f2345ba5ab80000000041004c87870d	-76.7396...
22	007d004fc4000082004b0000009f0000	b64e367a6a0400000000ca001abdbdd0	-143.772...
23	00a60087b800003b0056000000a90000	2320f0387fd10000000022000ccfcf2e	-154.113...
24	00c8009a7300004a0088000000b20000	831d40d49d1200000000cc00e4a9a928	-164.757...

Table 5: Linear characteristics included in the best linear hull. $i = \overline{1, 24}$

i	$\mu_1^{(i)}$	$\mu_2^{(i)}$	$\log_2 P(\mathbf{a} \to \mu_1^{(i)} \to \mu_2^{(i)} \to \mathbf{b})$
25	00d800b109000094004d000000b60000	7f64958f7dfc000000007200752d2d07	-156.587...
26	00f700012b00009d0085000000bd0000	2dbf720f1c8400000000006400d85858bc	-161.616...
27	00ff001416000072006700000bf0000	538398a26c73000000003b00909a9aab	-158.417...
28	003600b25f0000ae0047000000cd0000	251b719045f500000000ba00c59c9c7f	-148.417...
29	003900548900001b0004000000ce0000	8e333da6e55000000000d0004ae0e09a	-141.692...
30	003e00a76200004100a5000000cf0000	5b279b3d350200000000e5008d5e5e68	-143.470...
31	004f00770500005b00fd000000d30000	ac4bb2bc978a00000000df007bc0c0a4	-149.774...
32	005f005c7f00008500380000000d70000	503267e7776400000000006100ea44448b	-154.862...
33	00670034cc0000080094000000d90000	2b84f3a726a000000000fc00110b0bed	-150.264...
34	006f0021f10000e70076000000db0000	55b8190a565700000000a30059c9c9fa	-140.721...
35	007000ec5d00008c00f0000000dc0000	02e98067161c000000007700473131 30	-162.535...
36	00ac00d7ca00006f004c000000eb0000	4293e0bed39b00000000aa0096fdfd3c	-156.627...
37	00fa00a2b2000093003e000000fe0000	9918c412609c00000000d90085d4d45c	-174.676...

Table 6: Linear characteristics included in the best linear hull. $i = \overline{25, 37}$

C Codewords with minimum binary weight

Let $\mathbb{G} = \mathbb{E}|\mathbb{L}$ is a linear binary code, codeword length – 256 bits, infoword length – 128 bits.

$\mathbb{L}$ is 128×128 binary matrix, which defines the linear transformation of Kuznyechik.

It is shown (algorithm of the section (3)) that in a linear binary code $\mathbb{G}$ there are no codewords of binary weight 17, 18, 19, 20.

Two codewords with binary weight equal to 29 are found.

0 2 0 2 2 0 0 0 0 2 1 0 1 2 0 1	w
009000a0030000000000901000109000 4	x
15040009010001090000000003a00090	$y = x\mathbb{L}$
3 1 0 2 1 0 1 2 0 0 0 0 2 2 0 2	w

Table 7: The codeword with a binary weight equal to 29

2 0 2 2 0 0 0 0 2 1 0 1 2 0 1 3	w
9000a003000000000901000109000415	x
0400090100010900000000003a0009000	$y = x\mathbb{L}$
1 0 2 1 0 1 2 0 0 0 0 2 2 0 2 0	w

Table 8: Another codeword with a binary weight equal to 29

D The proof of Lemma 2

Lemma 2. *Let $\Delta x \to \Delta y$ is the differential in 2-round Kuznyechik. Let $P(\Delta x \to \Delta y)$ is maximal among all differentials. Then the number of active S-boxes in $\Delta x \to \Delta y$ is equal to $n + 1 = 17$.*

Proof Denote P_{best}^{diffA} the best differential with A active S-boxes.

It is shown that among differentials containing trails of weight $n+1 = 17$, the best probability is

$$P_{best}^{diff17} = \left(\frac{8}{256}\right)^{13} \left(\frac{6}{256}\right)^{4} = 2^{-86.660...}.$$

We will show that

$$P_{best}^{diff} = P_{best}^{diff17} > P_{best}^{diffA},\ n+2 \le A \le 2n.$$

Consider an arbitrary differential $\Delta x \to \Delta y$ with 18 active S-boxes. The differential consists of trails of the form $\Delta x \to \Delta_1 \to \Delta_2 \to \Delta y$. The difference Δx and all the Δ_1 differences have the same set of active S-boxes. $(k_1, \ldots, k_t)$ is the set of their positions. Similarly for Δy and Δ_2, let's denote the positions of active S-boxes $(m_1, \ldots, m_r)$, $t + r = 18$.

Using the algorithm (3), you can find all pairs (Δ_1, Δ_2) corresponding to this set of active S-boxes. All differential trails $\Delta x \to \ldots \to \Delta y$ can only pass through these pairs. During the algorithm execution the system of equations with $18 - n = 2$ free variables will be solved. The number of solutions, and accordingly the number of pairs (Δ_1, Δ_2), will not exceed $255^{18-n} = 255^2$.

Let's present the set of pairs found as a table $\boldsymbol{D}$. Table size is equal to $255^2 \times 18$. Each row corresponds to a pair $(\Delta_1^{(i)}, \Delta_2^{(i)}) = (\alpha_{k_1}^{(i)}, \ldots, \alpha_{k_t}^{(i)}, \beta_{m_1}^{(i)}, \ldots, \beta_{m_r}^{(i)})$, $i \le 255^2$, and each column corresponds to the active S-box.

By definition, the probability of a differential with 18 active S-boxes is:

$$P(\Delta x \to \Delta y) = \sum_{i=1}^{T} \left(\left(\prod_{j=1}^{t} P(x_{k_j} \to \alpha_{k_j}^{(i)}) \right) \left(\prod_{j=1}^{r} P(\beta_{m_j}^{(i)} \to y_{m_j}) \right) \right),$$
$$T \leq 255^2, \ t + r = 18.$$

Let the Δx and Δy are fixed. Then each element of the table can be matched with the probability $P(x_{k_j} \to \alpha_{k_j}^{(i)})$ (or $P(\beta_{m_j}^{(i)} \to y_{m_j})$). Let us denote this probability $P_{i,j}$, then the probability of the differential is:

$$P(\Delta x \to \Delta y) = \sum_{i=1}^{T} \prod_{j=1}^{r+t} P_{i,j}. \tag{12}$$

We give an upper bound of the (12).

Note that there are no more than 255 identical bytes in each column of the table $\boldsymbol{D}$. Otherwise, there are rows with a pair of identical bytes. This corresponds to the existence of a codeword with a weight less than $n+1 = 17$. It contradicts the MDS-code definition.

Let the input x_{k_j} or output y_{m_j} bytes are fixed. Then the same bytes in the table column match the same probabilities.

Denote $p_8 = \frac{8}{256}$, $p_6 = \frac{6}{256}$, $p_4 = \frac{4}{256}$, $p_2 = \frac{2}{256}$.

Let's use the majorants (8). They take the following values:

$$X = p_8, \underbrace{p_6, \ldots, p_6}_{5}, \underbrace{p_4, \ldots, p_4}_{21}, \underbrace{p_2, \ldots, p_2}_{87}, \underbrace{0, \ldots, 0}_{141}; \tag{13}$$

$$Y = p_8, p_8, \underbrace{p_6, \ldots, p_6}_{7}, \underbrace{p_4, \ldots, p_4}_{27}, \underbrace{p_2, \ldots, p_2}_{92}, \underbrace{0, \ldots, 0}_{127}. \tag{14}$$

You can see that Y is always greater than X. To get the highest estimate we consider the case when 2 columns of the table are estimated using X (and 16 columns – Y).

The number of nonzero elements in the majorant X is $v = 114$. This allows us to refine the maximum number of differential trails in the differential $T \leq v^2 = 12996$. And also refine the values of majorants:

$$X = \underbrace{p_8, \underbrace{p_6, \ldots, p_6}_{5}, \underbrace{p_4, \ldots, p_4}_{21}, \underbrace{p_2, \ldots, p_2}_{87}}_{v=114}; \tag{15}$$

$$Y = \underbrace{p_8, p_8, \underbrace{p_6, \ldots, p_6}_{7}, \underbrace{p_4, \ldots, p_4}_{27}, \underbrace{p_2, \ldots, p_2}_{78}}_{v=114}. \tag{16}$$

We divide the columns of the table into two groups:

$$\sum_{i=1}^{T} \prod_{j=1}^{r+t} P_{i,j} = \sum_{i=1}^{T} \underbrace{(P_{i,1} \cdot P_{i,2})}_{\mathbb{I}} \cdot \underbrace{\left(\prod_{j=3}^{18} P_{i,j} \right)}_{\mathbb{III}}. \tag{17}$$

We multiply the elements of the group $\mathbb{I}$ in pairs:

$$P_{i,1} \cdot P_{i,2} = P_i^{(I)}, \ \forall i = \overline{1, T}.$$

Arrange in each row of $\mathbb{III}$ all factors in non-increasing order.

Arrange the elements of each sequence $P_1^{(I)}, \ldots, P_T^{(I)}$, $P_{1,j}, \ldots, P_{T,j}$, $\forall j = \overline{3, 18}$ (columns in $\boldsymbol{D}$) in a non-increasing order. Denote the elements of the resulting sequences $\hat{P}_1^{(I)}, \ldots, \hat{P}_T^{(I)}$, $\hat{P}_{1,j}, \ldots, \hat{P}_{T,j}$, $\forall j = \overline{3, 18}$.

From the rearrangement inequality [6] it follows that

$$\sum_{i=1}^{T} \underbrace{(P_{i,1} \cdot P_{i,2})}_{\mathbb{I}} \cdot \underbrace{\left(\prod_{j=3}^{18} P_{i,j} \right)}_{\mathbb{III}} \leq \sum_{i=1}^{T} \underbrace{\hat{P}_i^{(I)}}_{\mathbb{I}} \cdot \underbrace{\left(\prod_{j=3}^{18} \hat{P}_{i,j} \right)}_{\mathbb{III}}. \tag{18}$$

Let's estimate $\hat{P}_1^{(I)}, \ldots, \hat{P}_T^{(I)}$ using X (13). Knowing that all pairs in the first and second columns are different, we replace the elements of the sequence by the $X \times X$:

$$p_8^2, \ \underbrace{p_8 p_6, \ldots, p_8 p_6}_{10 \ \text{lines}}, \ \underbrace{p_6, \ldots, p_6}_{25 \ \text{lines}}, \ \underbrace{p_8 p_4, \ldots, p_8 p_4}_{42 \ \text{lines}}, \ \ldots. \tag{19}$$

Let's estimate the group $\mathbb{III}$.

We note that the following inequality holds:

$$\hat{P}_i = \hat{P}_i^{(I)} \cdot \left(\prod_{j=3}^{18} \hat{P}_{i,j} \right) \leq \hat{P}_{i+1} = \hat{P}_{i+1}^{(I)} \cdot \left(\prod_{j=3}^{18} \hat{P}_{i+1,j} \right), \ \forall i = \overline{1, T-1}. \tag{20}$$

Assume that the coordinates of all elements p_8 in $\mathbb{III}$ are known (Fig.2.a).

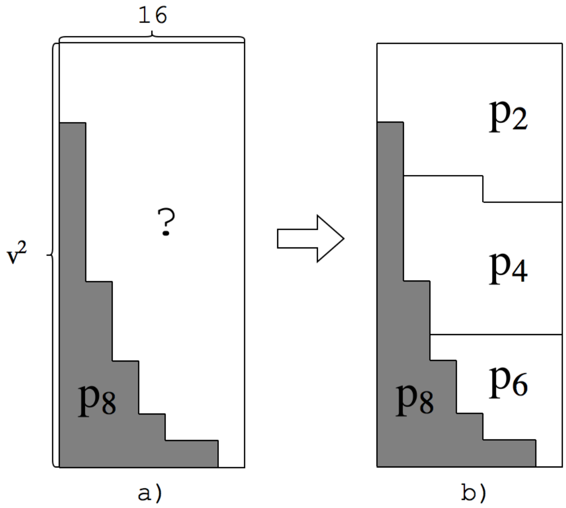

Figure 2: Reordering elements in $\mathbb{III}$.

We describe the *procedure for reordering* all elements p_6, p_4 and p_2 in $\mathbb{III}$.

1) Select the element in the first row $\hat{P}_{1,z} \neq p_8$, $z = \overline{3,18}$. Let z be the smallest (left column). If in the first row all elements are equal to p_8, we consider the second row, etc.

2) Find the maximum of all elements in $\mathbb{III}$, which have not been reordered before:

$$\hat{P}_{i',j'} = \max_{i,j} \hat{P}_{i,j},\ \ \hat{P}_{i,j} \neq p_8,\ \ i, i' = \overline{1,T},\ \ j, j' = \overline{3,18}.$$

3) We will exchange the values of the elements $\hat{P}_{1,z}$ and $\hat{P}_{i',j'}$. If $i' = 1$, then (18) does not change due to commutativity of multiplication. If $i' \neq 1$, then due to (20) then estimate (18) does not decrease. Note that after the exchange of elements can be broken inequalities (20).

4) Arrange the elements in columns $\hat{P}_1^{(I)}, \ldots, \hat{P}_T^{(I)}$, $\hat{P}_{1,j}, \ldots, \hat{P}_{T,j}$, $\forall j = \overline{3,18}$ by non-increasing. As a consequence of rearrangement inequality, (20) will be true. The value $\sum_{i=1}^{T} \hat{P}_i^{(I)} \cdot \left(\prod_{j=3}^{18} \hat{P}_{i,j} \right)$ will not decrease. The sequence $\hat{P}_1^{(I)}, \ldots, \hat{P}_T^{(I)}$, the coordinates of the elements p_8 and the value of the element with coordinates $(1, z)$ do not change.

The element with coordinates $(1, z)$ has been reordered.

We choose in the first row the next element not equal to p_8. We will perform the above steps 1 – 4.

Perform steps 1 – 4 sequentially for each element of the table not equal to p_8 and which has not been reordered before.

The result of the procedure will be the table $\widehat{\boldsymbol{D}}$. An exemplary view of the table $\widehat{\boldsymbol{D}}$ is shown in the figure 2.b. At each step of the procedure, the estimate (18) does not decrease. Suppose that there is a table $\widetilde{\boldsymbol{D}}$, which gives a greater estimate. If $\widetilde{\boldsymbol{D}}$ coincide with $\widehat{\boldsymbol{D}}$ within the accuracy of permutation of the same elements, then estimates (18) are the same, too. If $\widetilde{\boldsymbol{D}}$ does not coincide with $\widehat{\boldsymbol{D}}$, then apply the *reorder procedure* to the table $\widetilde{\boldsymbol{D}}$. Due to the steps that do not decrease the estimate (18), the table $\widehat{\boldsymbol{D}}$, will be built.

Thus it is proved that for a given arrangement of all elements p_8, the *reordering procedure* allows us to obtain the greatest estimate (18).

Let us now consider the possible arrangement of elements p_8 in the group $\mathbb{III}$.

The numbers of the elements p_8 in the tables $\boldsymbol{D}$ and $\widehat{\boldsymbol{D}}$ are the same. The number of rows containing the same number of elements p_8 also coincides.

Let w_i be the number of elements p_8 in the i-th row of the table $\widehat{\boldsymbol{D}}$, $w_i \geq w_{i+1}$, $i = \overline{1, T-1}$. Then

$$\sum_{i=1}^{T} w_i \leq v \cdot 16 \cdot 2 = 3648, \tag{21}$$

16 – the number of columns in the group $\mathbb{III}$, 2 – the number of elements p_8 in (16). Hence,

$$\left|\{i \ : w_i > 0,\ i = \overline{1, T}\}\right| \leq v \cdot 16 \cdot 2 = 3648. \tag{22}$$

The number of rows containing exactly 2 elements p_8 can be estimated as a $\binom{16}{2} \cdot 2^2$ – the number of pairs multiplied by the number of variants in the pair. Assume that the number of such pairs is greater. There are two different rows (two different codewords) that contain the same pair of bytes. Therefore, the sum of such codewords will give a codeword with a weight of 16 or less. It contradicts the MDS-code definition.

Let us estimate the number of rows with a greater number of elements. The maximum number of pairs is known – $\binom{16}{2} \cdot 2^2$. On the other hand, let i-th row contains w_i elements p_8, then this row contains $\binom{w_i}{2}$ different pairs of elements p_8. Then the number of rows containing exactly w elements p_8 is limited:

$$\left|\{i \ : w_i = w,\ i = \overline{1, T}\}\right| \leq \binom{16}{2} \cdot 2^2 / \binom{w}{2},\ 2 \leq w \leq 16. \tag{23}$$

And also:

$$\left|\{i \ : w_i \geq w,\ i = \overline{1, T}\}\right| \leq \binom{16}{2} \cdot 2^2 / \binom{w}{2},\ 2 \leq w \leq 16. \tag{24}$$

In addition, there should be a limit for the total number of pairs of elements p_8 in the table $\widehat{\boldsymbol{D}}$:

$$\sum_{i=1}^{T} \binom{w_i}{2} \leq \binom{16}{2} \cdot 2^2 = 480. \tag{25}$$

It is possible to show that the number of rows containing exactly $\omega = 8$ elements p_8, no more than $\rho \leq 5$. In each column of the table $\widehat{\boldsymbol{D}}$, no more than two different byte values correspond to the value of p_8. Any row must have at most one intersection (the same byte in the same column) with any other row. Initially, the number of bytes that were not selected is equal to $\nu = 2 \cdot 16 = 32$.

Choose the first row that contains exactly 8 elements p_8. Subtract $\omega = 8$ from ν.

Choose the second row that intersects the first row. Subtract $\omega - 1 = 7$ from ν.

Select the third row that intersects the first row and the second row. The minimum number that can be subtracted from ν is $\omega - 2 = 6$.

And so on:

$$\nu - (\omega \cdot \rho - \sum_{i=1}^{\rho-1} i) \geq 0,$$

$$\nu - \omega\rho + \frac{\rho(\rho-1)}{2} \geq 0,$$

$$\frac{1}{2}\rho^2 - (\omega + \frac{1}{2}) \cdot \rho + \nu \geq 0.$$

Then

$$\frac{1}{2}\rho^2 - (8 + \frac{1}{2}) \cdot \rho + 32 \geq 0 \tag{26}$$

Hence, $\rho \in \{0, 1, 2, 3, 4, 5\}$. If $\rho = 6$ then (26) less than zero.

Similarly, when $\omega = 9$ that $\rho \leq 4$. I.e. it is possible to show that the number of rows containing exactly 9 elements p_8, no more than 4. If $\omega = 10$ or $\omega = 11$ then $\rho \leq 3$. If $\omega \in \{12, 13, 14, 15, 16\}$ then $\rho \leq 2$.

Also, the following inequalities are true:

$$\begin{aligned} \left|\{i\ : w_i \geq 8,\ i = \overline{1,T}\}\right| &\leq 5 \\ \left|\{i\ : w_i \geq 9,\ i = \overline{1,T}\}\right| &\leq 4 \\ \left|\{i\ : w_i \geq 10,\ i = \overline{1,T}\}\right| &\leq 3 \\ \left|\{i\ : w_i \geq 12,\ i = \overline{1,T}\}\right| &\leq 2 \end{aligned} \tag{27}$$

$$w_i \leq \min\left(2 \cdot 16 - w_1 - (w_2 - 1) + 2,\ w_{i-1}\right),\ i = \overline{3, T} \tag{28}$$

Let $w_i > 2 \cdot 16 - w_1 - (w_2 - 1) + 2$. Then the i-th row must have at least two identical bytes with the first row or second row. It contradicts the MDS-code definition.

Let's iterate all possible sets w_i, $i = \overline{1, T}$. We will take into account the restrictions (21), (23), (25), (27), (28).

We choose the maximum estimate among all sets w_i, $i = \overline{1, T}$.

$$\sum_{i=1}^{T} \hat{P}_i^{(I)} \cdot \left(\prod_{j=3}^{18} \hat{P}_{i,j}\right) \leq 2^{-87.469\ldots} < P_{best}^{diff17} = 2^{-86.660\ldots}. \tag{29}$$

Note that it is possible to obtain more rough estimate without any additional search. We will not use restrictions (21), (25). Take the maximum values of the inequalities (24) and (27). The inequality (27) shows that the greatest $w_1, \ldots, w_5 = (16, 16, 11, 9, 8)$. Upper bounds in the inequality (24): exactly 7 elements p_8 – 17 rows, 6 elements – 10 rows, 5 elements – 16 rows, 4 elements – 32 rows, 3 elements – 80 rows, 2 elements – 320 rows, 1 element – 3168 rows.

$$\sum_{i=1}^{T} \hat{P}_i^{(I)} \cdot \left(\prod_{j=3}^{18} \hat{P}_{i,j}\right) \leq 2^{-87.012\ldots} < P_{best}^{diff17} = 2^{-86.660\ldots}. \tag{30}$$

The best estimate for a differential with 19 active S-boxes (P_{best}^{diff19}) cannot be greater than the best estimate for a differential with 18 active S-boxes (P_{best}^{diff18}).

$$P_{best}^{diff19} \leq \sum_{i=1}^{255} P(x_{k_1} \to \alpha_{k_1}^{(i)}) \cdot P_{best}^{diff18} =$$
$$= P_{best}^{diff18} \cdot \sum_{i=1}^{255} P(x_{k_1} \to \alpha_{k_1}^{(i)}) = P_{best}^{diff18} \cdot 1, \quad \forall k_1, x_{k_1}, \alpha_{k_1}.$$

Similarly for cases of $20, \ldots, 32$ active S-boxes.

Hence, the original lemma is proved:

$$P_{best}^{diff} = P_{best}^{diff17}.$$

Lemma 3. *Let* $(\mathbf{a} \to \mathbf{b})$ *is the linear hull in 2-round Kuznyechik. Let* $P(\mathbf{a} \to \mathbf{b})$ *be maximal among all linear hulls. Then the number of active S-boxes in*

$(\mathbf{a} \to \mathbf{b})$ *is equal to* $n+1=17$.

Proof The proof is analogous to the Lemma of the best differential.
p_8 is replaced by $p'_{28} = \left(\frac{2\cdot 28}{256}\right)^2$.
p_6, p_4, p_2 is replaced by $p'_{26} = \left(\frac{2\cdot 26}{256}\right)^2$, ... , $p'_2 = \left(\frac{2\cdot 2}{256}\right)^2$.
Majorants (13) and (14) are replaced by

$$X' = \underbrace{p'_{28},\ p'_{26},\ p'_{24}, p'_{24},\ p'_{22},\ p'_{20}, p'_{20}, p'_{20},\ p'_{18}, p'_{18}, p'_{18}, p'_{18},\ \ldots, \underbrace{p'_2, \ldots, p'_2}_{40}}_{242},\ \underbrace{0, \ldots, 0}_{13}$$

and

$$Y' = \underbrace{p'_{28}, p'_{28},\ p'_{24}, p'_{24},\ p'_{22}, p'_{22}, p'_{22},\ p'_{20}, p'_{20}, p'_{20}, p'_{20},\ \ldots, \underbrace{p'_2, \ldots, p'_2}_{7}}_{247},\ \underbrace{0, \ldots, 0}_{8}$$

correspondingly.
Estimate of $P(\mathbf{a}, \mathbf{b})$ similar to (29):

$$P(\mathbf{a}, \mathbf{b}) \le 2^{-77.310\ldots} < P^{lin}_{best} = 2^{-76.739\ldots}.$$

E Pseudocode of algorithms

Algorithm for finding codewords with the smallest byte weight

Algorithm 1 Algorithm for finding codewords with the smallest byte weight

Input: $k[1 \dots t]$ – nonzero x coordinates, $m[1 \dots r]$ – nonzero y coordinates,
$\mathbb{L}[1 \dots n, 1 \dots n]$, $t + r = n + 1$ // Matrix $\mathbb{L}$ in row-by-row representation
Output: $M^{(n+1)}(k_1, \dots, k_t, m_1, \dots, m_r)$

function *find_codewords*$(k[1 \dots t], m[1 \dots r], \mathbb{L}[1 \dots n, 1 \dots n])$
$m'[1...n-r] := \{i : \ i \notin m, \ 1 \le i \le n\}$ // zero y coordinates
$\mathbb{S}[1 \dots n-r, 1 \dots t]$
for $i := 1$ **to** $n - r$ **do**
  for $j := 1$ **to** t **do**
    $\mathbb{S}[i][j] := \mathbb{L}[m'[i]][k[j]]$
  end for
end for
$\mathbb{S}$:= identity_form($\mathbb{S}$)// Gauss method over $GF(2^8)$
// $\mathbb{S} = \begin{pmatrix} 1 & \cdots & 0 & c_1 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & c_{n-r} \end{pmatrix}$
codewords := {}
$\alpha[1 \dots t] := [0 \dots 0]$
for all e **in** $GF(2^8) \backslash 0$ **do**
  $\alpha[t] := e$
  for $i := 1$ **to** $t - 1$ **do**
    $\alpha[i] := e \times \mathbb{S}[i][t]$ // $\alpha_i = \alpha_t \times c_i$
  end for
  $\beta[1 \dots r] := L(\alpha)$ // zero coordinates are not specified
  codewords.add$((\alpha, \beta))$
end for
return codewords

The above algorithm could be easily generalized to finding small weight $w > n + 1$ codewords. In this case, the number of free variables in each subsystem $\mathbb{S}_{n-r,t}$ increases. Accordingly, the number of codewords generated by a single subsystem increases to 255^{w-n}. These codewords can include words that weigh less than w. This requires additional verification and increases the complexity of the algorithm.

The algorithm can be applied to an arbitrary MDS-code $(2n, n, n + 1)$ over any finite field $\mathbb{F}$.

We estimate the time complexity of the algorithm: Gaussian algorithm – $O(t^3)$; substitution of values – $O(ord(\mathbb{F})^{w-n})$; linear transformation – $O(n^2)$. The total complexity of the algorithm is $O(t^3 + ord(\mathbb{F})^{w-n} + n^2) = O(n^3 + ord(\mathbb{F})^{w-n})$.

One of the applications of this algorithm is the search in MDS-code codewords with small *binary* weight. The results are presented in Appendix B.

Algorithm for finding the best differential trail

Algorithm 2 Algorithm for finding the best differential trail

Input: $\mathbb{L}[1 \dots n, 1 \dots n]$, $\mathrm{DDT}[1 \dots 255, 1 \dots 255]$
  // $\mathrm{DDT}[\alpha_i, \beta_j] = P(\alpha_i \to \beta_j)$, $i, j = \overline{1, 255}$, $\alpha_i, \beta_j \in \{0,1\}^8 \backslash 0$
Output: best_diff_trails, P_{best}^{trail}

function *find_best_diff_trails*($\mathbb{L}[1 \dots n, 1 \dots n]$, $\mathrm{DDT}[1 \dots 255, 1 \dots 255]$)
best_diff_trails := {}
$P_{best}^{trail} := 0$
for $t := 1$ **to** n **do**
  $r := n + 1 - t$
  for all $k[1 \dots t]$ **in** combinations(n, t) **do**
    for all $m[1 \dots r]$ **in** combinations(n, r) **do**
      codewords := *find_codewords*($k[1 \dots t]$, $m[1 \dots r]$, $\mathbb{L}$)
      // codewords$[i]$ $=(\Delta_1^{(i)}, \Delta_2^{(i)})= (\alpha_{k_1}^{(i)} \dots \alpha_{k_t}^{(i)}, \beta_{m_1}^{(i)} \dots \beta_{m_r}^{(i)})$, $i = \overline{1, 255}$
      for all $\alpha[1 \dots t]$, $\beta[1 \dots r]$ **in** codewords **do**
        $P_{max}(\Delta_1, \Delta_2) :=$ *get_P_max*($\alpha[1 \dots t]$, $\beta[1 \dots r]$, DDT)
        if $P_{max}(\Delta_1, \Delta_2) = P_{best}^{trail}$ **then**
          best_diff_trails.add($(\alpha[1 \dots t], \beta[1 \dots r])$)
        end if
        if $P_{max}(\Delta_1, \Delta_2) > P_{best}^{trail}$ **then**
          $P_{best}^{trail} := P_{max}(\Delta_1, \Delta_2)$
          best_diff_trails := $\{(\alpha[1 \dots t], \beta[1 \dots r])\}$
        end if
      end for
    end for
  end for
end for
return best_diff_trails, P_{best}^{trail}

Time complexity of the algorithm 2 is

$$O\left(\underbrace{\sum_{t=1}^{n}\binom{n}{t}\binom{n}{n+1-t}}_{\text{all combinations}}\underbrace{\left(n^3 + \mathrm{ord}(\mathbb{F})\right)}_{\text{find\_codewords}} \cdot \underbrace{(n+1)}_{\text{get\_P\_max}}\right) =$$
$$= O\left(\binom{2n}{n+1} \cdot n^4\right) = O\left(\frac{2^{2n}}{\sqrt{n}} n^4\right)$$

Algorithm 3 Algorithm for calculating $P_{max}(\Delta_1, \Delta_2)$

function $get\_P\_max(\alpha[1 \ldots t], \beta[1 \ldots r], \mathrm{DDT}[1 \ldots 255, 1 \ldots 255])$
// $(\Delta_1, \Delta_2) = (\alpha_{k_1} \ldots \alpha_{k_t}, \beta_{m_1} \ldots \beta_{m_r})$
$P_{max}(\Delta_1, \Delta_2) := 1$
for $i := 1$ **to** t **do**
    $P_{max}(\Delta_1, \Delta_2) := P_{max}(\Delta_1, \Delta_2) \times \max\limits_x(\mathrm{DDT}[x][\alpha[i]])$
end for
for $j := 1$ **to** r **do**
    $P_{max}(\Delta_1, \Delta_2) := P_{max}(\Delta_1, \Delta_2) \times \max\limits_y(\mathrm{DDT}[\beta[j]][y])$
end for
// the values $\max\limits_x(\mathrm{DDT}[x][y])$, $\max\limits_y(\mathrm{DDT}[x][y])$ can easily be cached
return $P_{max}(\Delta_1, \Delta_2)$

The complexity of the algorithm is trivial – $O(t + r) = O(n)$

Algorithm for calculating the upper bound of the differential

Algorithm 4 Algorithm for calculating the upper bound of the differential

Input: $M^{(n+1)}(k_1, \ldots, k_t, m_1, \ldots, m_r)$, $\mathrm{DDT}[1 \ldots 255, 1 \ldots 255]$
Output: $P_{est} \geq P(\Delta x \to \Delta y)$
function get_upper_bound(codewords$[1 \ldots 255]$, DDT$[1 \ldots 255, 1 \ldots 255]$)
P_parts$[1 \ldots 255] := \{\}$
for $i := 1$ **to** 255 **do**
    $\alpha[1 \ldots t]$, $\beta[1 \ldots r]$:= codewords$[i]$
    P_parts$[i]$:= $get\_P\_max(\alpha[1 \ldots t-u], \beta[1 \ldots r-v], \mathrm{DDT})$// Let $u = v = 2$
end for
P_parts$[1 \ldots 255]$:= non_increasing_sort(P_parts$[1 \ldots 255]$)
$X[1 \ldots 255] := get\_majorant(\mathrm{DDT}[1 \ldots 255, 1 \ldots 255]$, input)
$Y[1 \ldots 255] := get\_majorant(\mathrm{DDT}[1 \ldots 255, 1 \ldots 255]$, output)
$P_{est} := 0$
for $i := 1$ **to** 255 **do**
    $P_{est} := P_{est} + X[i]^u \times Y[i]^v \times$ P_parts$[i]$
end for
return P_{est}

The values returned by the function *get_ majorant* can be cached. Therefore, the complexity of the algorithm 4 is equal to $O(ord(\mathbb{F}) \cdot n)$.

Algorithm 5 Algorithm for calculating X and Y

Input: DDT$[1\ldots255, 1\ldots255]$, input (X) or output (Y)
Output: $X[1\ldots255]$ or $Y[1\ldots255]$, 8

function *get_majorant*(DDT$[1\ldots255, 1\ldots255]$, input/output)
if output **then**
  DDT := transpose(DDT)
end if
for $i := 1$ **to** 255 **do**
  DDT$[i][1\ldots255]$:= non_increasing_sort(DDT$[i][1\ldots255]$) // sort rows
end for
majorant$[1\ldots255]$:= $[0,\ldots,0]$
for $i := 1$ **to** 255 **do**
  majorant$[i]$:= $\max_j$(DDT$[j][i]$) // select the maximum in the column
end for
// zero values can be removed
return majorant

Time complexity of the algorithm 5 is $O(ord(\mathbb{F})^2)$.

Algorithm for constructing the differential

Algorithm 6 Algorithm for constructing the differential

Input: $M^{(n+1)}(k_1,\ldots,k_t,m_1,\ldots,m_r)$, DDT$[1\ldots255, 1\ldots255]$, P_{est}^{diff}
Output: best_differentials, P_{est}^{diff}

function *construct_differentials*(codewords$[1\ldots255]$, DDT$[1\ldots255, 1\ldots255]$, P_{est}^{diff})
row_index := $\{1,\ldots,255\}$
row_est$[1\ldots255]$:= $[0,\ldots,0]$
for $i := 1$ **to** 255 **do**
  $\alpha[1\ldots t]$, $\beta[1\ldots r]$:= codewords$[i]$
  row_est$[i]$:= *get_P_max*($\alpha[1\ldots t]$, $\beta[1\ldots r]$, DDT)
end for
best_differentials := {}
external_bytes$[1\ldots t+r]$:= $[0,\ldots,0]$ // Δx and Δy
recursive_search(1, row_index, row_est)
return best_differentials, P_{est}^{diff}

The complexity of the algorithm 6 is determined by the complexity of algorithm 7.

Denote the complexity of the algorithm for constructing the differential as C_{diff}. In general case, algorithm 7 performs an exhaustive search of all inputs Δx and outputs Δy. In this case $C_{diff} = O(ord(\mathbb{F})^n)$. But in our practice, the average number of operations performed by the algorithm for constructing the differential is approximately equal to $ord(\mathbb{F})^2$. A more accurate estimate of the complexity is the subject of further research.

Algorithm 7 Recursive search of the differential

variables from Algorithm 6:
    codewords[$1 \dots 255$] // codewords[i]=codeword[$1 \dots t+r$], $i = \overline{1, 255}$
    DDT[$1 \dots 255, 1 \dots 255$]
    external_bytes[$1 \dots t+r$]
    P_{est}^{diff}
    best_differentials = {}
procedure *recursive_search*(column, row_index, row_est)
if column $> t+r$ **then**
    $\Delta x :=$ external_bytes[$1 \dots t$], $\Delta y :=$ external_bytes[$t+1 \dots t+r$]
    $P(\Delta x \to \Delta y) :=$ sum(row_est)
    if $P(\Delta x \to \Delta y) = P_{est}^{diff}$ **then**
        best_differentials.add($(\Delta x, \Delta y)$)
    end if
    if $P(\Delta x \to \Delta y) > P_{est}^{diff}$ **then**
        best_differentials = $\{(\Delta x, \Delta y)\}$
    end if
    return
end if
for $a := 1$ **to** 255 **do**
    external_bytes[column] $:= a$
    new_row_index := {}, new_row_est[$1 \dots 255$] := $[0, \dots, 0]$, $P_{est} := 0$
    for all i **in** row_index **do**
        codeword[$1 \dots t+r$] := codewords[i]
        $P_{trail} :=$ row_est[i]
        internal_byte := codeword[column]
        if column $\leq t$ **then**
            $P_{trail} := P_{trail} \times$ DDT[a][internal_byte]$/\max_x$(DDT[x][internal_byte])
        else
            $P_{trail} := P_{trail} \times$ DDT[internal_byte][a]$/\max_y$(DDT[internal_byte][y])
        end if
        if $P_{trail} > 0$ **then**
            $P_{est} := P_{est} + P_{trail}$
            new_row_index.add(i)
            new_row_est[i] $:= P_{trail}$
        end if
    end for
    if $P_{est} \geq P_{est}^{diff}$ **then**
        recursive_search(column+1, new_row_index, new_row_est)
    end if
end for

Algorithm for finding the best differential

Algorithm 8 Algorithm for finding the best differential

Input: $\mathbb{L}[1 \dots n, 1 \dots n]$, DDT$[1 \dots 255, 1 \dots 255]$, P_{best}^{trail}
Output: best_differentials, P_{best}^{diff}

function *find_best_differentials*($\mathbb{L}[1 \dots n, 1 \dots n]$, DDT$[1 \dots 255, 1 \dots 255]$)
best_differentials := {}
best_diff_trails, P_{best}^{trail} := *find_best_diff_trails*($\mathbb{L}$, DDT)
$P_{est}^{diff} := P_{best}^{trail}$
for $t := 1$ **to** n **do**
    $r := n + 1 - t$
    for all $k[1 \dots t]$ **in** combinations(n, t) **do**
        for all $m[1 \dots r]$ **in** combinations(n, r) **do**
            codewords := *find_codewords*($k[1 \dots t]$, $m[1 \dots r]$, $\mathbb{L}$)
            P_{est} := *get_upper_bound*(codewords, DDT)
            if $P_{est} < P_{est}^{diff}$ **then**
                continue
            end if
            differentials, P_{est} := *construct_differentials*(codewords, DDT, P_{est}^{diff})
            if $P_{est} = P_{est}^{diff}$ **then**
                best_differentials := best_differentials $\cup$ differentials
            end if
            if $P_{est} > P_{est}^{diff}$ **then**
                $P_{est}^{diff} := P_{est}$
                best_differentials := differentials
            end if
        end for
    end for
end for
$P_{best}^{diff} := P_{est}^{diff}$
return best_differentials, P_{best}^{diff}

Time complexity of the algorithm 8 is

$$O\left(\underbrace{\sum_{t=1}^{n}\binom{n}{t}\binom{n}{n+1-t}}_{\text{all combinations}}\left(\underbrace{n^3+\operatorname{ord}(\mathbb{F})}_{\text{find\_codewords}}+\underbrace{\operatorname{ord}(\mathbb{F})\cdot n}_{\text{get\_upper\_bound}}+\underbrace{C_{diff}}_{\text{construct\_differentials}}\right)\right)=$$

$$=O\left(\binom{2n}{n+1}\cdot C_{diff}\right)=O\left(\frac{2^{2n}}{\sqrt{n}}\cdot C_{diff}\right)$$