Abstract

This paper presents the complete description of the best differentials and linear hulls in 2-round Kuznyechik. We proved that 2-round \(\text{MEDP} = 2^{-86.666...}\), \(\text{MELP} = 2^{-76.739...}\). 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 \textit{exact} values of maximum expected differential and linear probability (MEDP, MELP) for 2-round Kuznyechik.

We proved that 2-round \(\text{MEDP} = 2^{-86.666...}\), \(\text{MELP} = 2^{-76.739...}\).

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 \( L \) 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:

\[
\begin{align*}
\Delta x &= (x_1, \ldots, x_n) - \text{the difference of input blocks in byte representation}, \\
\Delta_1 &= (\alpha_1, \ldots, \alpha_n) - \text{the difference of blocks after the nonlinear transformation on the first round}, \\
\Delta_2 &= (\beta_1, \ldots, \beta_n) = (\alpha_1, \ldots, \alpha_n)L - \text{the difference of blocks after the linear transformation (matrix multiplication in row-by-row representation)}, \\
\Delta y &= (y_1, \ldots, y_n) - \text{the difference of blocks after the nonlinear transformation on the second round}.
\end{align*}
\]

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:

\[
P(\alpha \rightarrow \beta) = \Pr(S(\chi \oplus \alpha) \oplus S(\chi) = \beta), \quad \alpha, \beta, \chi \in \{0, 1\}^8, \quad (1)
\]

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

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

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

The best differential trail has probability

\[ P_{\text{trail}}^{\text{best}} = P_{\text{best}}(\Delta x \rightarrow \Delta_1 \rightarrow \Delta_2 \rightarrow \Delta y) = \max_{(\Delta x, \Delta_1, \Delta_2, \Delta y) \neq (0,0,0,0)} P(\Delta x \rightarrow \Delta_1 \rightarrow \Delta_2 \rightarrow \Delta y). \]

Differential is the set of all differential trails that have the same \( \Delta x \) and \( \Delta y \).

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

\[ P(\Delta x \rightarrow \Delta y) = \sum_{i=1}^{T} \left( \left( \prod_{j=1}^{n} P(x_j \rightarrow \alpha_{j}^{(i)}) \right) \left( \prod_{j=1}^{n} P(\beta_{j}^{(i)} \rightarrow y_j) \right) \right), \quad (3) \]

where \( T \) is the number of the differential trails in the differential.

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

\[ P_{\text{diff}}^{\text{best}} = P_{\text{best}}(\Delta x \rightarrow \Delta y) = \max_{(\Delta x, \Delta y) \neq (0,0)} P(\Delta x \rightarrow \Delta y) \]

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 \( \Delta_1 \) and \( \Delta_2 \) 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 \( k_1, \ldots, k_t, \ m_1, \ldots, m_r \) – locations of non-zero elements in the vectors \( \Delta_1 = (\alpha_1, \ldots, \alpha_n) \) and \( \Delta_2 = (\beta_1, \ldots, \beta_n) \) 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 \( \Delta_1 \mathbb{L} = \Delta_2 \): \( (\alpha_{k_1}, \ldots, \alpha_{k_t}) \cdot \mathbb{S}_{n-r,t} = (0, \ldots, 0) \). Solve the subsystem \( \mathbb{S}_{n-r,t} \). The set of
solutions is \( (\alpha_{k_1}^{(i)}, \ldots, \alpha_{k_t}^{(i)}) \), \( i = 1, \ldots, 255 \). Hence we have the set of \( \Delta_1^{(i)} \) and the set of \( \Delta_2^{(i)} = \Delta_1^{(i)} \mathbb{I} \), \( i = 1, \ldots, 255 \).

Let’s denote these sets of solutions

\[
M^{(n+1)}(k_1, \ldots, k_t, m_1, \ldots, m_r) = (\alpha_{k_1}^{(i)}, \ldots, \alpha_{k_t}^{(i)}, \beta_{m_1}^{(i)}, \ldots, \beta_{m_r}^{(i)}), \quad i = 1, \ldots, 255.
\]

(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

\[
\Delta x \rightarrow \Delta_1 \rightarrow \Delta_2 \rightarrow \Delta y.
\]

We start with differential trails containing the minimum number of active S-boxes (minimal weight of \( \Delta_1 \) and \( \Delta_2 \)).

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.

\[
P_{\text{max}}(\Delta_1, \Delta_2) = \left( \prod_{j=1}^{t} \max_{x} P(x \rightarrow \alpha_{k_j}) \right) \left( \prod_{j=1}^{r} \max_{y} P(\beta_{m_j} \rightarrow y) \right)
\]

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

\[
P_{\text{trail \ best}} = \max_{(\Delta_1, \Delta_2) \neq (0,0)} P_{\text{max}}(\Delta_1, \Delta_2).
\]

Let the vector \( (\Delta_1, \Delta_2) \) has a weight \( n + 1 \):

\[
P_{\text{best \ trail}} \geq \max_{(\Delta_1, \Delta_2) \in M^{(n+1)}} P_{\text{max}}(\Delta_1, \Delta_2).
\]
Two sets of differential trails were found in $M^{(n+1)}$. Each trail in both sets has a maximum probability:

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

The trails in the set have the same inner part $(\Delta_1, \Delta_2)$. There are no other trails that would have a maximum probability.

The found differential trails are presented in Appendix A.

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

**Proof.** One can see that the estimate

$$P(\Delta x \rightarrow \Delta_1 \rightarrow \Delta_2 \rightarrow \Delta y) \leq \left( \max_{(\alpha, \beta) \setminus (0,0)} P(\alpha \rightarrow \beta) \right)^w.$$ 

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

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

$$P(\Delta x \rightarrow \Delta_1 \rightarrow \Delta_2 \rightarrow \Delta y) \leq \left( \frac{8}{256} \right)^w \leq \left( \frac{8}{256} \right)^{18} \leq \max_{(\Delta_1, \Delta_2) \in M^{(n+1)}} P_{\max}(\Delta_1, \Delta_2) = 2^{-86.66...}.$$ 

Hence $P_{\text{trail}}^{\text{best}} = 2^{-86.66...}$. 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 $\Delta x \rightarrow \Delta y$ 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 $P_{\text{trail}}^{\text{best}} = 2^{-86.66...}$. 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 $P_{\text{diff}}^{\text{best}} \geq P_{\text{trail}}^{\text{best}}$. We will use the probability $P_{\text{est}}^{\text{diff}} = P_{\text{trail}}^{\text{best}}$ 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 $\alpha$ and $r-v$ coordinates $\beta$ in the equation (4):

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

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

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

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):

$$P(x \rightarrow \alpha^{(1,x)}) \geq P(x \rightarrow \alpha^{(2,x)}) \geq \ldots \geq P(x \rightarrow \alpha^{(255,x)}), \ x = 1, 255, \ \text{(6)}$$

$$P(\beta^{(1,y)} \rightarrow y) \geq P(\beta^{(2,y)} \rightarrow y) \geq \ldots \geq P(\beta^{(255,y)} \rightarrow y), \ y = 1, 255. \ \text{(7)}$$

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

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

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

If the resulting upper bound (9) is less than the threshold $P_{\text{est}}^{\text{diff}}$, 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 $M^{(n+1)}(k_1, \ldots, k_l, m_1, \ldots, m_r)$ the estimate is greater than the threshold value $P_{est}^{diff}$. Then the following estimate also holds

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

We will sequentially search through possible non-zero values $x_{k_1}, \ldots, x_{k_l}$ and $y_{m_1}, \ldots, y_{m_r}$. The maximum values $\max_x P(x \to \alpha_{k_j}^{(i)})$ (and $\max_y P(\beta_{m_j}^{(i)} \to y)$) 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

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

$$P(a_1, a_2, \ldots a_s) \leq \sum_{i=1}^{255} \left( \prod_{j=1}^{s} P(a_j \to \alpha_{k_j}^{(i)}) \prod_{j=s+1}^{t} \max_x P(x \to \alpha_{k_j}^{(i)}) \prod_{j=1}^{r} \max_y P(\beta_{m_j}^{(i)} \to y) \right).$$

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 $\max_x P(x \to \alpha_{k_1}^{(i)})$ 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 $x_{k_2} = a_1$, $P(a_1) \geq P_{est}^{diff}$ is found. If such $x_{k_1}$ is not found, then the subset $M^{(n+1)}(k_1, \ldots, k_l, m_1, \ldots, m_r)$ 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 $\max_x P(x \to \alpha_{k_2}^{(i)})$ 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
\[ P(a_1, a_2, \ldots, a_s) \] with the value \( P(a \rightarrow \alpha_{k_{s+1}}^{(i)}) \), \( a = 1, 2, \ldots, 255 \). Multipliers \( \max P(\beta_{m_j}^{(i)} \rightarrow y) \) are replaced by values \( P(\beta_{m_j}^{(i)} \rightarrow b) \), \( b = 1, 2, \ldots, 255 \).

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

\[
P(\Delta x \rightarrow \Delta y) = \sum_{i=1}^{255} \left( \prod_{j=1}^{t} P(a_j \rightarrow \alpha_{j}^{(i)}) \right) \left( \prod_{j=1}^{r} P(\beta_{j}^{(i)} \rightarrow b_j) \right) \geq P_{est}^{diff}.
\]

(11)

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

The last step of the algorithm: \( P_{best}^{diff} = P_{est}^{diff} \).

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

The main idea of the proof is to construct an upper bound for the differential \( \Delta x \rightarrow \Delta y \) 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 \( 16 \times 16 \); pseudorandom, non-analytical S-box; DDT and LAT do not have obvious patterns.

**AES** – byte permutation layer and four MDS-matrix \( 4 \times 4 \); 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 $4 \times 4$. 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. $\text{MEDP} = 2^{-28.272...}$, $\text{MELP} = 2^{-27.287...}$.

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: $\text{MEDP} = 2^{-86.66...}$, $\text{MELP} = 2^{-76.739...}$.

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 $\text{MEDP} = 2^{-86.66...}$, $\text{MELP} = 2^{-76.739...}$. The estimate (5) for a differential trail (linear characteristic) is not achieved for 2-round Kuznyechik.

For any LSX cipher, the $N$-round $\text{MEDP}$ ($\text{MELP}$) is the upper bound for $(N + 1)$-round $\text{MEDP}$ ($\text{MELP}$). Therefore, the 2-round $\text{MEDP}$ ($\text{MELP}$) 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


Appendix

A The best differentials

<table>
<thead>
<tr>
<th>$\Delta_1$</th>
<th>$\Delta_2$</th>
<th>$P(\Delta x \rightarrow \Delta_1) \cdot 256$</th>
<th>$P(\Delta_2 \rightarrow \Delta_2) \cdot 256$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 8 0 0 0 0 8 0 8 8 0 6 0 0 8</td>
<td>0 0 19 0 0 0 0 0 0 0 2 d 0 0 b 8 8 9 5 0 0 7 2 0 0 0 0 2 8</td>
<td>$\Delta_1$</td>
<td>$\Delta_2$</td>
</tr>
<tr>
<td>2 a 0 0 0 0 0 d 2 3 3 7 f 7 4 d 0 0 8 2 a 8 0 0 0 0 0 0 9 d 1 b</td>
<td>8 0 0 8 8 8 8 6 0 8 6 0 0 0 6 8</td>
<td>$P(\Delta_2 \rightarrow \Delta y) \cdot 256$</td>
<td>$P(\Delta_2 \rightarrow \Delta y) \cdot 256$</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>$\Delta_1$</th>
<th>$\Delta_2$</th>
<th>$P(\Delta x \rightarrow \Delta_1) \cdot 256$</th>
<th>$P(\Delta_2 \rightarrow \Delta_2) \cdot 256$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 8 8 6 0 8 8 0 8 8 0 0 8 8</td>
<td>0 0 a 5 d e f 7 0 0 8 5 8 5 3 7 0 0 e c 0 3 0 0 0 0 9 c 0 0 5 a</td>
<td>$\Delta_1$</td>
<td>$\Delta_2$</td>
</tr>
<tr>
<td>0 0 6 8 e a 0 d 0 0 f 7 0 0 d d 0 0 6 d 0 0 0 0 0 0 0 0 0 9 0</td>
<td>0 6 6 8 0 8 0 6 0 8 0 0 0 0 0 0 8</td>
<td>$P(\Delta_2 \rightarrow \Delta y) \cdot 256$</td>
<td>$P(\Delta_2 \rightarrow \Delta y) \cdot 256$</td>
</tr>
</tbody>
</table>

Table 2: Second optimal internal part ($\Delta_1 \rightarrow \Delta_2$). 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 $\alpha$ and $\beta$ are replaced by input/output masks $\alpha'$ and $\beta'$ correspondingly.

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

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

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

$$a \rightarrow \mu_1 \rightarrow \mu_2 \rightarrow b.$$

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

$$P(a \rightarrow \mu_1 \rightarrow \mu_2 \rightarrow b) = \left(\prod_{j=1}^{n} P(a[j] \rightarrow \mu_1[j])\right) \left(\prod_{j=1}^{n} P(\mu_2[j] \rightarrow b[j])\right).$$
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 \(a\) and output mask \(b\).

\[
(a \rightarrow b) = \{ a \rightarrow \mu_1^{(i)} \rightarrow \mu_2^{(i)} \rightarrow b, \ i = 1, T \}.
\]

The probability of the linear hull \((a \rightarrow b)\) is equal to:

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

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

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

The trivial estimate of the two-round linear characteristic is

\[
\max_{(a,\mu_1,\mu_2, b) \in (0,0,0,0)} P(a \rightarrow \mu_1 \rightarrow \mu_2 \rightarrow b) \leq \left( \frac{56}{256} \right)^{2.17} = 2^{-74.549...}.
\]

The following results are obtained by executing the algorithms.

The best linear characteristic has a probability equal to

\[
\max_{(a,\mu_1,\mu_2, b) \in (0,0,0,0)} P(a \rightarrow \mu_1 \rightarrow \mu_2 \rightarrow b) = \left( \frac{56}{256} \right)^{2.10} \left( \frac{52}{256} \right)^{2.4} \left( \frac{48}{256} \right)^{2.3} = 2^{-76.739...}.
\]

The linear hull \((a \rightarrow b)\) has a nontrivial form and (unlike the differential method) contains 37 linear characteristics \(a \rightarrow \mu_1^{(i)} \rightarrow \mu_2^{(i)} \rightarrow b, \ i = 1, 37\).

The exact probability of the linear hull is

\[
\max_{(a,b) \in (0,0)} P(a \rightarrow b) = 2^{-76.739...} \cdot (1 + 2^{-61.407}).
\]
The optimal inner part \((\mu_1 \rightarrow \mu_2)\) generates the best linear hull.

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

<table>
<thead>
<tr>
<th>(i)</th>
<th>(\mu_1^{(i)})</th>
<th>(\mu_2^{(i)})</th>
<th>(\log_2 P(a \rightarrow \mu_1^{(i)} \rightarrow \mu_2^{(i)} \rightarrow b))</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>000800153d0000ef00e2000000020000</td>
<td>7e3ceead70f700000005f0048c2c17</td>
<td>-160.980...</td>
</tr>
<tr>
<td>2</td>
<td>02002005f40000bc008b00000080000</td>
<td>f9f3abb6c1d400000007c02209095e</td>
<td>-150.676...</td>
</tr>
<tr>
<td>3</td>
<td>002f009b580000d70000000000f000</td>
<td>aea232db819600000a8003c1f194</td>
<td>-157.973...</td>
</tr>
<tr>
<td>4</td>
<td>0046005e0200002200b70000101000</td>
<td>27f2f1f753e9000000cd0082adad4f</td>
<td>-158.150...</td>
</tr>
<tr>
<td>5</td>
<td>006900ee2000002b007f0000001a000</td>
<td>752916773291000000b002f0dd8f4</td>
<td>-155.551...</td>
</tr>
<tr>
<td>6</td>
<td>007100b20600001a00580000001c000</td>
<td>f76c2981a2b80000003a00f69e9ec</td>
<td>-139.633...</td>
</tr>
<tr>
<td>7</td>
<td>008d00bd04000045006f00000023000</td>
<td>4ee5e2eaa6d20000009b005bb5b9e</td>
<td>-148.300...</td>
</tr>
<tr>
<td>8</td>
<td>00a200d42600004c0a700000028000</td>
<td>1c3e056ec7aa00000000d1002ce4e4f</td>
<td>-154.032...</td>
</tr>
<tr>
<td>9</td>
<td>00bd00c8a00002700210000002f000</td>
<td>4b6f9c0387e100000590066d6d6e</td>
<td>-141.656...</td>
</tr>
<tr>
<td>10</td>
<td>00cb00e30600067008000000032000</td>
<td>69171319f53b000000560087f6f6d1</td>
<td>-152.336...</td>
</tr>
<tr>
<td>11</td>
<td>000a005070000054000a00000042000</td>
<td>61b310b4ac400000080009a323212</td>
<td>-160.721...</td>
</tr>
<tr>
<td>12</td>
<td>004b00fd9b00002c00000000052000</td>
<td>935547e2ff10000007000df212a1</td>
<td>-148.862...</td>
</tr>
<tr>
<td>13</td>
<td>006b00d6e10000f200c900000056000</td>
<td>6f2c92b1cf1000000tec045ba5b80</td>
<td>-156.558...</td>
</tr>
<tr>
<td>14</td>
<td>006300be5200007f00650000058000</td>
<td>149a06f19deb0000005300b5eaeae6</td>
<td>-140.159...</td>
</tr>
<tr>
<td>15</td>
<td>007b00801500004e0042000005e000</td>
<td>96df39070ec2000002b006acacde</td>
<td>-146.218...</td>
</tr>
<tr>
<td>16</td>
<td>00b00063130002900a00000006e000</td>
<td>ffc82a1efbf9000000e400eb5a5a0f</td>
<td>-155.166...</td>
</tr>
<tr>
<td>17</td>
<td>00b70090f800007300b00000064000</td>
<td>2adc8c852ba000000d1002ce4e4f</td>
<td>-150.862...</td>
</tr>
<tr>
<td>18</td>
<td>006900f42600004c0a700000028000</td>
<td>24433e2b48f0000009700f873736f</td>
<td>-147.574...</td>
</tr>
<tr>
<td>19</td>
<td>001400a1e40000a9003400000085000</td>
<td>c367200d589500000011035656524</td>
<td>-152.616...</td>
</tr>
<tr>
<td>20</td>
<td>00b5006f600008b000000094000</td>
<td>e495d1fa07c0000000dc007c88686</td>
<td>-151.329...</td>
</tr>
<tr>
<td>21</td>
<td>006a009755000060200000094000</td>
<td>9f234ba6ab800000041004c87870d</td>
<td>-76.7396...</td>
</tr>
<tr>
<td>22</td>
<td>007d004c4e0000820040000009f000</td>
<td>b66a3676a04000000ca01ababdd0</td>
<td>-143.772...</td>
</tr>
<tr>
<td>23</td>
<td>00a60087b00003b005000000a90000</td>
<td>2320f387fd100000020000cccf2e</td>
<td>-154.113...</td>
</tr>
<tr>
<td>24</td>
<td>00c8009a7300004a008800000b2000</td>
<td>831d40d49d12000000cc0e4a9a928</td>
<td>-164.757...</td>
</tr>
</tbody>
</table>

Table 5: Linear characteristics included in the best linear hull. \(i = 1, 24\)
Table 6: Linear characteristics included in the best linear hull. \( i = 25, 37 \)

C Codewords with minimum binary weight

Let \( G = E|L \) is a linear binary code, codeword length – 256 bits, infoword length – 128 bits.

\( L \) 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 \( G \) there are no codewords of binary weight 17, 18, 19, 20.

Two codewords with binary weight equal to 29 are found.

\[
\begin{array}{cccccccccccccccccccc}
0 & 2 & 0 & 2 & 0 & 0 & 0 & 0 & 2 & 1 & 0 & 1 & 2 & 0 & 1 & \hline
009000a00030000000009010001090004 & w \\
1504000901000109000000000003a00090 & y = xL \\
3 & 1 & 0 & 2 & 1 & 0 & 1 & 2 & 0 & 0 & 0 & 2 & 2 & 0 & 2 & w \\
\end{array}
\]

Table 7: The codeword with a binary weight equal to 29
Table 8: Another codeword with a binary weight equal to 29

<table>
<thead>
<tr>
<th>2 0 2 2 0 0 0 0 2 1 0 1 2 0 1 3</th>
<th>w</th>
</tr>
</thead>
<tbody>
<tr>
<td>9000a00300000000901000109000415</td>
<td>x</td>
</tr>
<tr>
<td>040009010010900000000003a0009000</td>
<td>y = xL</td>
</tr>
<tr>
<td>1 0 2 1 0 1 2 0 0 0 2 2 0 2 0</td>
<td>w</td>
</tr>
</tbody>
</table>

D The proof of Lemma 2

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

Proof

Denote $P_{\text{best}}^{\text{diff}}$ 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

$$P_{\text{best}}^{\text{diff}17} = \left( \frac{8}{256} \right)^{13} \left( \frac{6}{256} \right)^{4} = 2^{-86.660...}.$$

We will show that

$$P_{\text{best}}^{\text{diff}} = P_{\text{best}}^{\text{diff}17} > P_{\text{best}}^{\text{diff}A}, \ n + 2 \leq A \leq 2n.$$

Consider an arbitrary differential $\Delta x \rightarrow \Delta y$ with 18 active S-boxes. The differential consists of trails of the form $\Delta x \rightarrow \Delta_1 \rightarrow \Delta_2 \rightarrow \Delta y$. The difference $\Delta x$ and all the $\Delta_1$ differences have the same set of active S-boxes. $(k_1, \ldots, k_t)$ is the set of their positions. Similarly for $\Delta y$ and $\Delta_2$, 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 $(\Delta_1, \Delta_2)$ corresponding to this set of active S-boxes. All differential trails $\Delta x \rightarrow \ldots \rightarrow \Delta y$ 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 $(\Delta_1, \Delta_2)$, will not exceed $255^{18-n} = 255^2$.

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

By definition, the probability of a differential with 18 active S-boxes is:
\[ P(\Delta x \rightarrow \Delta y) = \sum_{i=1}^{T} \left( \prod_{j=1}^{t} P(x_{k_j} \rightarrow \alpha_{k_j}^{(i)}) \right) \left( \prod_{j=1}^{r} P(\beta_{m_j}^{(i)} \rightarrow y_{m_j}) \right), \]

\[ T \leq 255^2, \ t + r = 18. \]

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

\[ P(\Delta x \rightarrow \Delta y) = \sum_{i=1}^{T} \prod_{j=1}^{r+t} P_{i,j}. \quad (12) \]

We give an upper bound of the \((12)\).

Note that there are no more than 255 identical bytes in each column of the table \( D \). 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:

\[ X = p_8, p_6, \ldots, p_6, p_4, \ldots, p_4, p_2, \ldots, p_2, 0, \ldots, 0; \quad (13) \]

\[ Y = p_8, p_8, p_6, \ldots, p_6, p_4, \ldots, p_4, p_2, \ldots, p_2, 0, \ldots, 0. \quad (14) \]

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:

\[ X = p_8, p_6, \ldots, p_6, p_4, \ldots, p_4, p_2, \ldots, p_2; \]

\[ y = 114 \]
\[ Y = p_8, p_8, p_6, \ldots, p_6, p_4, \ldots, p_4, p_2, \ldots, p_2. \]  
\[ v = 114 \]

We divide the columns of the table into two groups:

\[ \sum_{i=1}^{T} \prod_{j=1}^{r+t} P_{i,j} = \sum_{i=1}^{T} \left( P_{i,1} \cdot P_{i,2} \right) \cdot \left( \prod_{j=3}^{18} P_{i,j} \right). \]  

(17)

We multiply the elements of the group II in pairs:

\[ P_{i,1} \cdot P_{i,2} = P^{(I)}_i, \ \forall i = 1, T. \]

Arrange in each row of III all factors in non-increasing order.

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

From the rearrangement inequality [6] it follows that

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

(18)

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

\[ p^2_8, \ p_8 p_6, \ldots, p_8 p_6, \ p_6, \ldots, p_6, \ p_8 p_4, \ldots, p_8 p_4, \ \ldots \]  

(19)

Let’s estimate the group III.

We note that the following inequality holds:

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

(20)

Assume that the coordinates of all elements \( p_8 \) in III are known (Fig.2.a).
We describe the procedure for reordering all elements $p_6$, $p_4$ and $p_2$ in $\mathbb{II}$.

1) Select the element in the first row $\hat{P}_{1,z}$, $z \neq p_8$, $z = 3, 18$. 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 $\mathbb{II}$, which have not been reordered before:

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

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 $\hat{P}_{1}^{(I)}, \ldots, \hat{P}_{T}^{(I)}, \hat{P}_{1,j}, \ldots, \hat{P}_{T,j}$, $\forall j = 3, 18$ 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 $\hat{D}$. An exemplary view of the table $\hat{D}$ 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 $\hat{D}$, which gives
a greater estimate. If $\tilde{D}$ coincide with $\hat{D}$ within the accuracy of permutation of the same elements, then estimates (18) are the same, too. If $\tilde{D}$ does not coincide with $\hat{D}$, then apply the reorder procedure to the table $\tilde{D}$. Due to the steps that do not decrease the estimate (18), the table $\hat{D}$, 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 II.

The numbers of the elements $p_8$ in the tables $D$ and $\hat{D}$ 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 $\hat{D}$, $w_i \geq w_{i+1}$, $i = 1, T-1$. Then

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

16 – the number of columns in the group II, 2 – the number of elements $p_8$ in (16). Hence,

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

The number of rows containing exactly 2 elements $p_8$ can be estimated as a $\left( \binom{16}{2} \right) \cdot 2^2$ – 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 – $\left( \binom{16}{2} \right) \cdot 2^2$. 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:

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

And also:

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

In addition, there should be a limit for the total number of pairs of ele-
ments $p_8$ in the table $\hat{D}$:

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

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 $\hat{D}$, 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 $\nu$.

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

Select the third row that intersects the first row and the second row. The minimum number that can be subtracted from $\nu$ 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

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

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:

$$\left|\{i : w_i \geq 8, i = 1, T\}\right| \leq 5$$

$$\left|\{i : w_i \geq 9, i = 1, T\}\right| \leq 4$$

$$\left|\{i : w_i \geq 10, i = 1, T\}\right| \leq 3$$

$$\left|\{i : w_i \geq 12, i = 1, T\}\right| \leq 2 \quad (27)$$
\[ w_i \leq \min \left( 2 \cdot 16 - w_1 - (w_2 - 1) + 2, \ w_{i-1} \right), \ i = 3, T \]  
(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 \( w_i, \ i = 1, T \). We will take into account the restrictions (21), (23), (25), (27), (28).

We choose the maximum estimate among all sets \( w_i, \ i = 1, T \).

\[
\sum_{i=1}^{T} \hat{P}_i^{(j)} \cdot \left( \prod_{j=3}^{18} \hat{P}_{i,j} \right) \leq 2^{-87.469} < P_{\text{best}}^{\text{diff}17} = 2^{-86.660}.. 
\]  
(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.

\[
\sum_{i=1}^{T} \hat{P}_i^{(j)} \cdot \left( \prod_{j=3}^{18} \hat{P}_{i,j} \right) \leq 2^{-87.012} < P_{\text{best}}^{\text{diff}17} = 2^{-86.660}.. 
\]  
(30)

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

\[
P_{\text{best}}^{\text{diff}19} \leq \sum_{i=1}^{255} P(x_{k_1} \rightarrow \alpha_{k_1}^{(i)}) \cdot P_{\text{best}}^{\text{diff}18} =
\]

\[
= P_{\text{best}}^{\text{diff}18} \cdot \sum_{i=1}^{255} P(x_{k_1} \rightarrow \alpha_{k_1}^{(i)}) = P_{\text{best}}^{\text{diff}18} \cdot 1, \ \forall k_1, x_{k_1}, \alpha_{k_1}.
\]

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

Hence, the original lemma is proved:

\[
P_{\text{best}}^{\text{diff}} = P_{\text{best}}^{\text{diff}17}.
\]

**Lemma 3.** Let \((a \rightarrow b)\) is the linear hull in 2-round Kuznyechik. Let \( P(a \rightarrow b) \) be maximal among all linear hulls. Then the number of active S-boxes in
(a → b) is equal to $n + 1 = 17$.

**Proof** The proof is analogous to the Lemma of the best differential. $p_8$ is replaced by $p'_{28} = \left(\frac{2.28}{256}\right)^2$.

$p_6, p_4, p_2$ is replaced by $p'_{26} = \left(\frac{2.26}{256}\right)^2, \ldots, p'_{22} = \left(\frac{2.22}{256}\right)^2$.

Majorants (13) and (14) are replaced by

\[
X' = p'_{28}, p'_{26}, p'_{24}, p'_{22}, p'_{20}, p'_{20}, p'_{18}, p'_{18}, p'_{18}, \ldots, p'_{2}, \ldots, p'_{2}, 0, \ldots, 0
\]

and

\[
\]

correspondingly.

Estimate of $P(a, b)$ similar to (29):

\[
P(a, b) \leq 2^{-77.310...} < P_{\text{lin}}^{\text{best}} = 2^{-76.739...}.
\]
E Pseudocode of algorithms

Algorithm for finding codewords with the smallest byte weight

Algorithm 1 Algorithm for finding codewords with the smallest byte weight

\[ \text{Input:} \quad k[1 \ldots t] - \text{nonzero } x \text{ coordinates}, \quad m[1 \ldots r] - \text{nonzero } y \text{ coordinates}, \quad L[1 \ldots n, 1 \ldots n], \quad t + r = n + 1 \quad /\ Matrix \ L \text{ in row-by-row representation} \]

\[ \text{Output:} \quad M^{(n+1)}(k_1, \ldots, k_t, m_1, \ldots, m_r) \]

1: function find_codewords(k[1 \ldots t], m[1 \ldots r], L[1 \ldots n, 1 \ldots n])
2: \[ m'[1..n-r] := \{i : i \notin m, \ 1 \leq i \leq n\} \quad /\ zero \ y \ coordinates \]
3: \[ S[1..n-r, 1..t] \]
4: for i := 1 to n - r do
5: \quad for j := 1 to t do
6: \quad \quad S[i][j] := L[m'[i]][k[j]]
7: \quad end for
8: end for
9: \[ S := \text{identity_form}(S) \quad /\ Gauss \ method \ over \ GF(2^8) \]
10: \[ /\ S = \begin{pmatrix} 1 & \ldots & 0 & c_1 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \ldots & 1 & c_{n-r} \end{pmatrix} \]
11: codewords := {} \[ \]
12: \[ \alpha[1 \ldots t] := [0 \ldots 0] \]
13: for all e in GF(2^8) \[ \]
14: \[ \alpha[t] := e \]
15: for i := 1 to t - 1 do
16: \[ \alpha[i] := e \times S[i][t] \quad /\ \alpha_i = \alpha_t \times c_i \]
17: end for
18: \[ \beta[1 \ldots r] := L(\alpha) \quad /\ \text{zero coordinates are not specified} \]
19: codewords.add((\alpha, \beta))
20: end for
21: 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 \( S_{n-r,t} \) 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 – \( O(\text{ord}(\mathbb{F})^{w-n}) \); linear transformation – \( O(n^2) \). The total complexity of the algorithm is \( O(t^3 + \text{ord}(\mathbb{F})^{w-n} + n^2) = O(n^3 + \text{ord}(\mathbb{F})^{w-n}) \).
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

\begin{algorithm}
\caption{Algorithm for finding the best differential trail}
\begin{algorithmic}
\Require $\mathbb{L}[1 \ldots n, 1 \ldots n], \text{DDT}[1 \ldots 255, 1 \ldots 255]$ \Comment{DDT[$\alpha_i, \beta_j$] = $P(\alpha_i \rightarrow \beta_j)$, $i, j = 1, 255$, $\alpha_i, \beta_j \in \{0, 1\}^8$}
\Ensure best\_diff\_trails, $P_{\text{trail}}^\text{best}$
\Function{find\_best\_diff\_trails}{$\mathbb{L}[1 \ldots n, 1 \ldots n], \text{DDT}[1 \ldots 255, 1 \ldots 255]$} \Comment{$\text{DDT}[\alpha_i, \beta_j] = P(\alpha_i \rightarrow \beta_j)$, $i, j = 1, 255$, $\alpha_i, \beta_j \in \{0, 1\}^8$}
\State $\text{best\_diff\_trails} := \emptyset$
\State $P_{\text{trail}}^\text{best} := 0$
\For{$t = 1$ to $n$}
\State $r := n + 1 - t$
\ForAll{$[k_1 \ldots t]$ in combinations($n, t$)}
\ForAll{$m_1 \ldots r$ in combinations($n, r$)}
\State codewords := find\_codewords($k_1 \ldots t$, $m_1 \ldots r$, $\mathbb{L}$) \Comment{codewords[i] = $(\Delta_1^{(i)}, \Delta_2^{(i)}) = (\alpha_{k_1}^{(i)}, \ldots, \alpha_{k_t}^{(i)}, \beta_{m_1}^{(i)}, \ldots, \beta_{m_r}^{(i)}), i = 1, 255$}
\ForAll{$\alpha_1 \ldots t$, $\beta_1 \ldots r$ in codewords} \Comment{find\_codewords $\cdot$ \text{all combinations} ($n+1$)}
\State $P_{\text{max}}(\Delta_1, \Delta_2) := \text{get\_P\_max}(\alpha_1 \ldots t, \beta_1 \ldots r, \text{DDT})$
\If{$P_{\text{max}}(\Delta_1, \Delta_2) = P_{\text{trail}}^\text{best}$}
\State best\_diff\_trails.add((\alpha_1 \ldots t, \beta_1 \ldots r))
\EndIf
\If{$P_{\text{max}}(\Delta_1, \Delta_2) > P_{\text{trail}}^\text{best}$}
\State $P_{\text{trail}}^\text{best} := P_{\text{max}}(\Delta_1, \Delta_2)$
\State best\_diff\_trails := \{(\alpha_1 \ldots t, \beta_1 \ldots r)\}
\EndIf
\EndFor
\EndFor
\EndFor
\Return best\_diff\_trails, $P_{\text{trail}}^\text{best}$
\end{algorithmic}
\end{algorithm}

Time complexity of the algorithm 2 is

$$O \left( \sum_{t=1}^{n} \binom{n}{t} \binom{n}{n+1-t} \left( \frac{n^3 + \text{ord}({\mathbb{F}})}{\text{find\_codewords}} \right) \cdot \frac{(n+1)}{\text{get\_P\_max}} \right) = O \left( \binom{2n}{n+1} \cdot n^4 \right) = O \left( \frac{2n^2}{n} \right)$$
Algorithm 3 Algorithm for calculating $P_{\text{max}}(\Delta_1, \Delta_2)$

1: function $get\_P\_\text{max}(\alpha[1 \ldots t], \beta[1 \ldots r], \text{DDT}[1 \ldots 255, 1 \ldots 255])$
2: // $(\Delta_1, \Delta_2) = (\alpha_{k_1} \ldots \alpha_{k_t}, \beta_{m_1} \ldots \beta_{m_r})$
3: $P_{\text{max}}(\Delta_1, \Delta_2) := 1$
4: for $i := 1$ to $t$ do
5: $P_{\text{max}}(\Delta_1, \Delta_2) := P_{\text{max}}(\Delta_1, \Delta_2) \times \max_x (\text{DDT}[x][\alpha[i]])$
6: end for
7: for $j := 1$ to $r$ do
8: $P_{\text{max}}(\Delta_1, \Delta_2) := P_{\text{max}}(\Delta_1, \Delta_2) \times \max_y (\text{DDT}[\beta[j]][y])$
9: end for
10: // the values $\max_x (\text{DDT}[x][\alpha[i]])$, $\max_y (\text{DDT}[\beta[j]][y])$ can easily be cached
11: return $P_{\text{max}}(\Delta_1, \Delta_2)$

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

Algorithm 4 Algorithm for calculating the upper bound of the differential

Algorithm 4 Algorithm for calculating the upper bound of the differential

Input: $M^{(n+1)}(k_1, \ldots, k_t, m_1, \ldots, m_r)$, DDT[1 \ldots 255, 1 \ldots 255]

Output: $P_{\text{est}} \geq P(\Delta x \rightarrow \Delta y)$

1: function $get\_\text{upper\_bound}(\text{codewords}[1 \ldots 255], \text{DDT}[1 \ldots 255, 1 \ldots 255])$
2: $\text{P\_parts}[1 \ldots 255] := \{\}$
3: for $i := 1$ to $255$ do
4: $\alpha[1 \ldots t], \beta[1 \ldots r] := \text{codewords}[i]$
5: $\text{P\_parts}[i] := get\_P\_\text{max}(\alpha[1 \ldots t - u], \beta[1 \ldots r - v], \text{DDT}) // \text{Let } u = v = 2$
6: end for
7: $\text{P\_parts}[1 \ldots 255] := \text{non\_increasing\_sort}(\text{P\_parts}[1 \ldots 255])$
8: $X[1 \ldots 255] := get\_\text{majorant}(\text{DDT}[1 \ldots 255, 1 \ldots 255], \text{input})$
9: $Y[1 \ldots 255] := get\_\text{majorant}(\text{DDT}[1 \ldots 255, 1 \ldots 255], \text{output})$
10: $P_{\text{est}} := 0$
11: for $i := 1$ to $255$ do
12: $P_{\text{est}} := P_{\text{est}} + X[i]^u \times Y[i]^v \times \text{P\_parts}[i]$
13: end for
14: return $P_{\text{est}}$

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

25
Algorithm 5  Algorithm for calculating $X$ and $Y$

**Input:** DDT[1...255,1...255], input ($X$) or output ($Y$)

**Output:** $X[1...255]$ or $Y[1...255]$, 8

1: function get_majorant(DDT[1...255,1...255], input/output)
2: if output then
3: DDT := transpose(DDT)
4: end if
5: for $i := 1$ to 255 do
6: DDT[$i$][1...255] := non_increasing_sort(DDT[$i$][1...255]) // sort rows
7: end for
8: majorant[1...255] := [0,...,0]
9: for $i := 1$ to 255 do
10: majorant[$i$] := max$_j$(DDT[$j$][$i$]) // select the maximum in the column
11: end for
12: // zero values can be removed
13: return majorant

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

Algorithm 6  Algorithm for constructing the differential

**Input:** $M^{(n+1)}(k_1,\ldots,k_t,m_1,\ldots,m_r)$, DDT[1...255,1...255], $P^{\text{est}}$

**Output:** best_differentials, $P^{\text{diff}}$

1: function construct_differentials(codewords[1...255], DDT[1...255,1...255], $P^{\text{est}}$)
2: row_index := {1,...,255}
3: row_est[1...255] := [0,...,0]
4: for $i := 1$ to 255 do
5: $\alpha[1...t], \beta[1...r]$ := codewords[$i$]
6: row_est[$i$] := get_P_max($\alpha[1...t], \beta[1...r],$ DDT)
7: end for
8: best_differentials := {};
9: external_bytes[1...$t+r$] := [0,...,0] // $\Delta x$ and $\Delta y$
10: recursive_search(1, row_index, row_est)
11: return best_differentials, $P^{\text{diff}}$

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

**Algorithm 7** Recursive search of the differential

1: variables from Algorithm 6:
2:   codewords[1...255] // codewords[i]=codeword[1...t + r], i = 1,255
3:   DDT[1...255,1...255]
4:   external_bytes[1...t + r]
5:   $P_{\text{est}}^{\text{diff}}$
6:   best_differentials = {}
7: procedure recursive_search(column, row_index, row_est)
8: if column > t + r then
9:   $\Delta x := $ external_bytes[1...t], $\Delta y := $ external_bytes[t + 1...t + r]
10:   $P(\Delta x \to \Delta y) := \text{sum}(\text{row_est})$
11:   if $P(\Delta x \to \Delta y) = P_{\text{est}}^{\text{diff}}$ then
12:      best_differentials.add((\Delta x, \Delta y))
13: end if
14: if $P(\Delta x \to \Delta y) > P_{\text{est}}^{\text{diff}}$ then
15:      best_differentials = {((\Delta x, \Delta y))
16: end if
17: return
18: end if
19: for a := 1 to 255 do
20:   external_bytes[column] := a
21:   new_row_index := {}, new_row_est[1...255] := [0,...,0], $P_{\text{est}} := 0$
22: for all i in row_index do
23:   codeword[1...t + r] := codewords[i]
24:   $P_{\text{trail}} := \text{row_est}[i]$
25:   internal_byte := codeword[column]
26: if column ≤ t then
27:   $P_{\text{trail}} := P_{\text{trail}} \times \text{DDT}[a][\text{internal_byte}] / \text{max}(\text{DDT}[x][\text{internal_byte}])$
28: else
29:   $P_{\text{trail}} := P_{\text{trail}} \times \text{DDT}[\text{internal_byte}][a] / \text{max}(\text{DDT}[\text{internal_byte}][y])$
30: end if
31: if $P_{\text{trail}} > 0$ then
32:   $P_{\text{est}} := P_{\text{est}} + P_{\text{trail}}$
33:   new_row_index.add(i)
34:   new_row_est[i] := $P_{\text{trail}}$
35: end if
36: end for
37: if $P_{\text{est}} \geq P_{\text{est}}^{\text{diff}}$ then
38:   recursive_search(column + 1, new_row_index, new_row_est)
39: end if
40: end for
Algorithm for finding the best differential

Algorithm 8 Algorithm for finding the best differential

Input: $L[1 \ldots n, 1 \ldots n]$, DDT$[1 \ldots 255, 1 \ldots 255]$, $P_{\text{trail}}$
Output: best_differentials, $P_{\text{diff}}$

1: function find_best_differentials($L[1 \ldots n, 1 \ldots n]$, DDT$[1 \ldots 255, 1 \ldots 255]$
2: best_differentials := {} 
3: best_diff_trails, $P_{\text{trail}}$ := find_best_diff_trails($L$, DDT)
4: $P_{\text{est}}$ := $P_{\text{trail}}$
5: for $t := 1$ to $n$ do 
6: $r := n + 1 - t$
7: for all $k[1 \ldots t]$ in combinations($n, t$) do
8: for all $m[1 \ldots r]$ in combinations($n, r$) do
9: codewords := find_codewords($k[1 \ldots t]$, $m[1 \ldots r]$, $L$)
10: $P_{\text{est}}$ := get_upper_bound(codewords, DDT)
11: if $P_{\text{est}} < P_{\text{diff}}$ then 
12: continue
13: end if
14: differentials, $P_{\text{est}}$ := construct_differentials(codewords, DDT, $P_{\text{diff}}$)
15: if $P_{\text{est}} = P_{\text{diff}}$ then
16: best_differentials := best_differentials $\cup$ differentials
17: end if
18: if $P_{\text{est}} > P_{\text{diff}}$ then 
19: $P_{\text{diff}}$ := $P_{\text{est}}$
20: best_differentials := differentials
21: end if
22: end for
23: end for
24: end for
25: $P_{\text{diff}}$ := $P_{\text{est}}$
26: return best_differentials, $P_{\text{diff}}$

Time complexity of the algorithm 8 is

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

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