Journal of Discrete Mathematical Sciences & Cryptography
ISSN 0972-0529 (Print), ISSN 2169-0065 (Online)
Vol. 28 (2025), No. 6, pp. 2309–2322
DOI : 10.47974/JDMSC-2230

On retrieval of secret key parameters from symmetric block cipher O'zDST 1105:2009 using modified SQUARE cryptanalysis

Alisher Ikramov
Department of Mathematics
New Uzbekistan University
National University of Uzbekistan
Tashkent 100100
Uzbekistan

Abstract

In this article, the application of SQUARE cryptanalysis to the Uzbek standard of symmetric block encryption is investigated. We have constructed an efficient attack on the 1st and 2nd rounds of the Uzbek algorithm, allowing to retrieve several bytes of the secret key. We found that the constructed attack is almost surely successful.

This article highlights the importance of studying the interaction of different transformations, as the success of the devised attack relies on using operations in the ring modulo 256, rather than in a finite field.

Subject Classification (2020): *11T71, 13C13.*

Keywords: *Cryptanalysis, SQUARE, Symmetric block cipher.*

1. Introduction

This paper investigates the symmetric block encryption algorithm known as O'zDSt 1105:2009, which was officially adopted as Uzbekistan's encryption standard on September 28, 2009 [1]. The algorithm was designed to resist linear and differential cryptanalysis. Although it is utilized for protecting confidential information, it is not sanctioned for encrypting state secrets. For encrypting classified data, the Soviet encryption standard GOST 28147-89 [2] remains in effect.

Abdurakhimov et al. [3] conducted a study examining algebraic attacks on the O'zDSt 1105:2009 encryption algorithm, estimating the attack's complexity to be around 2^{73} operations.

ORCID-ID: `0000-0002-4302-6791`

E-mail: `a.ikramov@newuu.uz`

SQUARE, an encryption algorithm developed by Joan Daemen and Vincent Rijmen, is known for its robustness against linear and differential analysis [4]. However, Lars Knudsen created a novel method to recover secret round keys using vulnerabilities of this cipher, termed SQUARE cryptanalysis [4].

We adopt a similar method to uncover the parameters calculated on the secret key and used in some round keys as well.

When developing new encryption algorithms, it is essential to consider existing cryptanalysis techniques to ensure resistance against them. SQUARE cryptanalysis is particularly relevant for block encryption algorithms with a byte structure. This approach is demonstrated in works such as [5], [6], [7]. Employing bit-level operations to disrupt the byte structure can circumvent SQUARE cryptanalysis due to its complexity [8], [9].

2. The Description of O'zDSt 1105:2009 Algorithm

2.1 *Overview of the Design*

Uzbek symmetric block encryption algorithm uses an 8×4 matrix as its internal state. Unless stated otherwise, all calculations are performed modulo 256.

The initialization of the Algorithm requires Key expansion. It uses two 256-bit secret keys. The Algorithm comprises 8 rounds, each consisting of the steps below:

- Adding internal state with the round key using XOR
- Internal state is divided into two square matrices, each of them is multiplied by a corresponding Diamatrix. Bytes from the expanded key are used to form both diamatrices (Aralash)
- Permutation of positions of the elements (Sur)
- Non-linear operation using S-boxes that are calculated on the secret parameters from the expanded key (B_a)

After 8 rounds, the Algorithm adds the next round key and applies Aralash.

2.2 *Key Expansion Procedure*

The Algorithm requires two secret keys k and k_f each of length of 256 bits. The last 192 bits of k_f are used to form k'.

$$k_v = k + k' \times (1 + k_f \times k)$$

Then the Algorithm forms the expanded key k_{se} taking the leftmost 672 bits of k_v. 8 rightmost bytes of k_{se} are values $(d_1, R_1, L_1, C_1, d_2, R_2, L_2, C_2)$, which are used to calculate the substitution tables for the non-linear operation.

Bytes 40 to 21 from the right form two secret diamatrices for the Aralash operation.

The round key is formed from the first 256 leftmost bits of k_{se} for each round. k_{se} is then cyclically shifted 83 bits to the left (this procedure is repeated after retrieving the current round key).

2.3 *Linear operation Aralash and formation of Diamatrices*

To multiply diamatrices $C = A \otimes B$, the formula is as follows:

$$C_{i,i} = A_{i,i} \sum_{u=0}^{3} B_{u,i} - \sum_{u=0,u\neq i}^{3} B_{u,i} A_{u,u} \pmod{256}$$

$$C_{i,j} = A_{i,j} \sum_{u=0}^{3} B_{u,j} + B_{i,j} \sum_{u=0}^{3} A_{u,j} - \sum_{u=0,u\neq i}^{3} B_{u,i} A_{i,u} \pmod{256},$$

$$i \neq j$$

Based on the formulas provided above authors a diadeterminant (DIADET) of a specific diamatrix where some elements are the same was defined as follows:

$$D = \begin{pmatrix} d_7 & d_0 & d_1 & d_2 \\ d_8 & d_7 & d_8 & d_8 \\ d_9 & d_3 & d_7 & d_9 \\ d_4 & d_5 & d_6 & d_7 \end{pmatrix} \quad (1)$$

We get numbers $d_0, \ldots, d_9$ from the expanded key (for the top diamatrix bytes from 40 to 31 from right, for the bottom diamatrix bytes from 30 to 21 from right). Thus, any information on them also reveals information on some round keys.

$$\begin{aligned} DIADET(D) = \; & d_7 \times (d_7 + d_1 + d_8 + d_9 + d_6) \times \\ & (d_7 + d_0 + d_8 + d_3 + d_5) \times \\ & (d_7 + d_2 + d_8 + d_9 + d_4) \pmod{256} \end{aligned} \quad (2)$$

The operations modulo 256 require diadeterminant of a diamatrix to be odd in order to have an inverse diamatrix with the defined formulas of multiplication. Thus, each clause in (2) must be equal to an odd number. The diamatrix $D^{\backslash -1}$ is denotement for an inverse diamatrix to D.

A specific *correction procedure* was described by the authors of the Algorithm to fix non-invertible diamatrices in a deterministic way, ensuring both parties get the same secret:

- if any number d_i is zero for i from 0 to 9, then we replace it with 255;
- if a diagonal element d_7 is even, then we subtract 1 from it modulo 256;
- if the term $d_7 + d_0 + d_8 + d_3 + d_5$ is even, then we change $d_5 = d_5 - 1 \pmod{256}$;
- if the term $d_7 + d_1 + d_8 + d_9 + d_6$ is even, then we change $d_6 = d_6 - 1 \pmod{256}$;
- if the term $d_7 + d_2 + d_8 + d_9 + d_4$ is even, then we change $d_4 = d_4 - 1 \pmod{256}$.

For each block of the input text, we represent it and its intermediate values as an 8×4 matrix. Using the procedures described above, we form two diamatrices D_1, D_2 from different positions of the expanded key. Each time we call Aralash function in the Algorithm, we split the internal state into two square matrices: the upper part consists of the first four rows, and the lower part has of

the next four rows of the internal state. The function Aralash then multiplies each square matrix by the corresponding secret diamatrix (D_1, D_2). Then we replace the old internal state with new values keeping the same order of the parts.

2.4 *Parametric Multiplication and formulas for the Substitution Tables*

Let $R_k \in \mathbb{Z}_{257}$ (value from 0 to 256). For any two numbers $a, b \in \mathbb{Z}_{257}$ we define the parametric multiplication (with R_k) in the following way:

$$a \otimes_{R_k} b(\mathrm{mod} p) = (a \times b \times R_k + a + b)(\mathrm{mod}\ 257) \quad (3)$$

The introduced operation is both commutative and associative as it uses regular multiplication and addition in Abelian finite field. Moreover, we get a commutative monoid in case of parametric multiplication modulo prime number. The ability to reverse this parametric operation is crucial to form a substitution table. We extract the group component of the newly formed monoid solely from its non-zero values. The inverse operation for any element a in the monoid, concerning parametric multiplication with R_k, is represented as $a_{R_k}^{\backslash -1}$.

Algorithm uses exactly two S-boxes B_1, B_2. Each of them is formed from four numbers R_k, d_k, L_k, C_k, $k \in \{1,2\}$ taken from the expanded key using following procedure [1]:

1. FOR $t = 0$ TO 255 WITH STEP = 1
2. $\quad a[t] = (t + L_k)(\mathrm{mod}\ 256) + 1$
3. $\quad B_i[t] = a[t]_{R_k}^{\backslash d_k}(\mathrm{mod}\ 256)$
4. END FOR
5. FOR $t = 0$ TO 255 WITH STEP = 1
6. $\quad$ IF ($t > 0$ AND $|B_k[t] - B_k[t-1]| < 8$) OR $B_k[t] == t$
7. $\quad\quad$ SWAP ($B_k[t]$, $B[(t - C_k)\ (\mathrm{mod}\ 256)]$)
8. $\quad$ END IF
9. END FOR

Algorithm calls the function $B_a(e, H)$ with S-box B_1 in odd rounds and S-box B_2 in even rounds. It subsequently replaces each element within the internal state according to the selected S-box, represented as $H[i,j] = B_k(H[i,j])$ with $k = (r(\mathrm{mod}2)) + 1$.

According to the design [1], the respective parameter R must be no less than 1, and parameter d should satisfy the condition $d \equiv 3(\mathrm{mod}\ 4)$. If $R = 0$ then we have linear substitution tables. All other values of d result in non-reversible S-boxes.

2.5 Sur to change positions of bytes in the internal state

The function $Sur(H)$ executes a set of cyclic shifts for the rows indexed by $i \in \{0,1,2,3\}$ and the columns indexed by $j \in \{0,1,2, \ldots, 7\}$, as described below:

- First, i-th column of the internal state is cyclically shifted by $(i+1)(\mathrm{mod}8)$ rows DOWN;
- Then, j-th row of the internal state is cyclically shifted by $(j+1)(\mathrm{mod}4)$ columns RIGHT.

3. Method

The SP-system in algorithm O'zDST 1105:2009 shares similarities with the SQUARE algorithm [4]. In this section, we delve into the algebraic properties of the O'zDST 1105:2009 encryption process.

Transformation Components

- **Substitution Tables (B_a):** these tables represent only non-linear operations in the algorithm.
- **Multiplication by Secret Matrices (Aralash):** This linear operation influences each output byte based on 6 or 7 input bytes. The transformation relies on two secret diamatrices calculated on values from the expanded key. Although the coefficients are non-fixed, there exists a fixed dependency due to the multiplication of diamatrices. The internal state H of shape 8×4 is divided onto two matrices H_1, H_2 (first 4 rows of H are in H_1, next 4 rows are in H_2). Then $H_{1,new} = H_{1,old} \otimes KD_1$ and $H_{2,new} = H_{2,old} \otimes KD_2$ are calculated. The results are joined into 8×4 new internal state H_{new} (first 4 rows are from $H_{1,new}$, next 4 rows are from $H_{2,new}$).
- **XOR with Key:** A linear operation that modifies the current state using round keys.
- **Byte Permutation (Sur):** Another linear operation that rearranges the bytes but does not change their values.

Byte-Oriented Operations:

B_a, XOR, and Sur operate at the byte level. Each byte's value depends solely on a corresponding byte from the input text. Consequently, distinct input texts yield distinct output values.

The *SQUARE cryptanalysis* technique involves the following steps:

1. **Plaintext Selection.** We choose plaintexts that differ in only one byte (indexed as i) while keeping other bytes identical. This results in a set of 256 input texts.
2. **Several Rounds of Encryption.** Apply fixed number of rounds of the O'zDST 1105:2009 encryption algorithm to each input text.
3. **Observation of Output Values.** For each position, count the number of different values across all output texts. If all output texts have the same value at a position, we label it as *inactive.* If distinct values appear at a position across all output texts, we label it as *active.*
4. **Analysis of Output Positions.** Some positions exhibit more than one but fewer than 256 distinct values after one round of O'zDST 1105:2009

encryption. This behavior arises due to multiplications within the ring modulo 256.

4. Theoretical and Practical Results

Initially, we will demonstrate the formula for calculating exponentiation using the parametric multiplication through conventional arithmetic methods. Then we will prove some properties that lead to significant dependence between many S-boxes produced by these formulas.

Theorem 1:

$$y_{R_k}^{\backslash m} = y\sum_{i=1}^{m}\binom{m}{i}(R_k y)^{i-1} \tag{4}$$

Proof: Proof by induction. It is obvious, that $y_{R_k}^{\backslash 1} = y$. Hence, we consider the formula (4) to be true for m. We can now analyze it for $m+1$:

$$\begin{aligned}
y_{R_k}^{\backslash m+1} &= R_k \times y_{R_k}^{\backslash m} \times y + y + y_{R_k}^{\backslash m} \\
&= R_k y^2 \sum_{j=1}^{m}\binom{m}{j}(R_k y)^{j-1} + y + y\sum_{j=1}^{m}\binom{m}{j}(R_k y)^{j-1} \\
&= y\sum_{j=1}^{m}\binom{m}{j}(R_k y)^{j-1} + y + y\sum_{j=1}^{m}\binom{m}{j}(R_k y)^{j} \\
&= y\sum_{j=1}^{m}\binom{m}{j}(R_k y)^{j-1} + y + y\sum_{j=2}^{m+1}\binom{m}{j-1}(R_k y)^{j-1} \\
&= y\sum_{j=2}^{m}\left[\binom{m}{j} + \binom{m}{j-1}\right](R_k y)^{j-1} + y(m+1) \\
&+(R_k y)^m y = y(m+1) + (R_k y)^m y \\
&+y\sum_{j=1}^{m}\binom{m+1}{j}(R_k y)^{j-1} = y\sum_{j=1}^{m+1}\binom{m+1}{j}(R_k y)^{j-1}
\end{aligned}$$

Hence, the formula (4) is correct.

Theorem 2: *Let* $R_k \neq 0$. *Then*

$$y_{R_k}^{\backslash m} = \frac{((yR_k+1)^m - 1)}{R_k} \tag{5}$$

Proof: Using the formula from Theorem 1, we can write $y_{R_k}^{\backslash m} = y\sum_{j=1}^{m}\binom{m}{j}(R_k y)^{j-1} = \frac{\sum_{j=1}^{m}\binom{m}{j}(R_k y)^{j}}{R_k} = \frac{\left(\sum_{j=0}^{m}\binom{m}{j}(R_k y)^{j} - 1\right)}{R_k} = \frac{((yR_k+1)^m - 1)}{R_k}$.

Theorem 3: *For any* $x \in \mathbb{Z}_{257}$ *and any* $R \neq 0(\mathrm{mod}257)$

$$R \times x_R^{\backslash 128} = \begin{cases} 0 \\ 255 \\ 256 \end{cases} \pmod{257}$$

Proof: According to Fermat's theorem $\phi(257) = 257 - 1 = 256$. So, for any $a \in [1, \ldots, 256]$ we have $a^{256} \equiv 1(\mathrm{mod}257)$ as 257 is prime number. Hence,

$a^{128} = \begin{cases} 1 \\ -1 \\ 0 \end{cases} (\text{mod } 257)$. Using the results of Theorem 2, we have $R \times x_R^{\backslash 128} = (R \times x + 1)^{128} - 1 (\text{mod} 257) = \begin{cases} 1-1 \\ -1-1 \\ 0-1 \end{cases} (\text{mod } 257) = \begin{cases} 0 \\ 255 \\ 256 \end{cases} (\text{mod } 257)$.

Theorem 4: *Let* $R \neq 0(mod 257)$. *Then for any* $x \in \mathbb{Z}_{257}$ *and any whole number* d

$$x_R^{\backslash d+128} = \begin{cases} -\frac{1}{R} \\ -\frac{2}{R} - x_R^{\backslash d} \\ x_R^{\backslash d} \end{cases} (\text{mod } 257).$$

Proof: Using the results of Theorem 3

$$x_R^{\backslash d+128} = \left(x_R^{\backslash d} + x_R^{\backslash 128} + R \times x_R^{\backslash d} \times x_R^{\backslash 128}\right)$$

$$(\text{mod } 257) = \begin{cases} x_R^{\backslash d} + 0 \\ x_R^{\backslash d} + \frac{255}{R} + R \times x_R^{\backslash d} \times \frac{255}{R} \\ x_R^{\backslash d} + \frac{256}{R} + R \times x_R^{\backslash d} \times \frac{256}{R} \end{cases}$$

$$(\text{mod } 257) = \begin{cases} x_R^{\backslash d} \\ \frac{255}{R} - x_R^{\backslash d} \\ \frac{255}{R} \end{cases} \quad (6)$$

$$(\text{mod } 257) = \begin{cases} -\frac{1}{R} \\ -\frac{2}{R} - x_R^{\backslash d} \\ x_R^{\backslash d} \end{cases} (\text{mod } 257).$$

Note: we get $x_R^{\backslash d+128} = -\frac{1}{R} (\text{mod } 257)$ only when $Rx + 1 = 0(\text{mod } 257)$, or $x = -\frac{1}{R} (\text{mod } 257)$.

Theorem 5: *Let* $R \neq 0(mod\ 257)$, $d \equiv 3(mod\ 4)$. *Then for any* $x \in \mathbb{Z}_{257}$ *such that* $x_R^{\backslash d+128} = -\frac{2}{R} - x_R^{\backslash d} (mod\ 257)$ *the following is true:*

$$x_R^{\backslash d+128} = \left(-\frac{2}{R} - x\right)_R^{\backslash d} (\text{mod} 257).$$

Proof: Using the results of Theorems 2 and 4

$$x_R^{\backslash d+128} = -\frac{2}{R} - x_R^{\backslash d} \quad (\text{mod } 257)$$

$$= -\frac{2}{R} - \frac{(Rx+1)^d - 1}{R} \quad (\text{mod } 257) \quad (7)$$

$$= -\frac{2}{R} - \frac{(Rx+1)^d}{R} + \frac{1}{R} \pmod{257}$$
$$= -\frac{(Rx+1)^d+1}{R} \pmod{257}$$

Similarly, if we take $-\frac{2}{R} - x$:

$$(-\frac{2}{R} - x)_R^{\backslash d} = \frac{(R\left(-\frac{2}{R}-x\right)+1)^d-1}{R} \pmod{257}$$
$$= \frac{1}{R}((-2 - Rx + 1)^d - 1) \pmod{257} \quad (8)$$
$$= \frac{1}{R}((-Rx - 1)^d - 1) \pmod{257}.$$

As 257 is a prime number, $\phi(257) = 256$ is even, and d is odd, we get $(-a)^d = -(a)^d (\mathrm{mod} 257)$. Hence,

$$(-\frac{2}{R} - x)_R^{\backslash d} = \frac{1}{R}\ (-(Rx + 1)^d - 1) \pmod{257}$$
$$= -\frac{1}{R}\ ((Rx + 1)^d + 1) \pmod{257} \quad (9)$$

As right sides of Equations (7) and (9) are equal, the Theorem is true.

Now, we construct the modified SQUARE attack on one round of O'zDST 1105:2009 algorithm.

1. $M_i = [i, 0,0, \dots ,0] \in [0,255]^{32}$, $i \in [0,255]$.
2. $E_i = OneRound(M_i, Key)$, we use the same Key as given in [1].
3. Create sets $F_j = \{E_{0,j}, E_{1,j}, E_{2,j}, \dots , E_{255,j}\}$ for $j \in [0, 31]$, where $E_{i,j}$ is byte in position j of E_i.
4. Calculate $c_j = |F_j|$. If c_j is 1, then byte in position j is inactive. If $c_j = 256$, then byte in position j is active but of no interest.
5. Find all j such that $1 < c_j < 256$. Put all such j into V.
6. Define

$$\psi(R, d, j) = \max_{i_0 \in F_j} |\{i \in F_j : B_{R,d}^{-1}(E_{i,j}) = B_{R,d}^{-1}(E_{i_0,j}) \pmod 2\}|$$

7. Find all pairs (R, d) that provide maximum of ψ on all $j \in V$. If such pair is unique, it is the correct pair of secret parameters. Otherwise, change position of i in M_i and repeat the process.

Theorem 6: *Let* $d \equiv 3(mod4), R \neq 0(mod256), t \in \{1,2,3,4,5,6,7\}$, *and* $S_{y,t,d,R} = \{((x \times 2^t + y)(mod256))_R^{\backslash d}(mod257): x \in \{0,1,2, \dots ,255\}\}$. *Then*

$$\forall y(y \in \{0,1,2, \dots ,255\}) \rightarrow (|S_{y,t,d,R}| = 2^{8-t}).$$

Proof: Take any x such that $0 \le x < 2^{8-t}$. Then $2^t \times (x + k \times 2^{8-t})$ $(\mathrm{mod}256) = 2^t \times x + 2^t \times k \times 2^{8-t}(\mathrm{mod}256) = 2^t \times x + k \times 2^8(\mathrm{mod}256)$ $= 2^t \times x$ for any $k \in [0,1,2,\dots,2^t - 1]$. Hence, there are 2^t elements in each equivalency class of x in multiplication by 2^t, there are 2^8 numbers in total modulo 256, so, there are $2^8 \div 2^t = 2^{8-t}$ different values in $S_{y,t,d,R}$.

We ran a computational experiment to test when solution (R, d) is unique in providing maximum of ψ. The results are shown in the Table 1. For $t > 4$ in 2^t we have more than 2 solutions for some values of R.

We used 16 specific matrices, in which only one value was non-zero and equal to one. We multiplied such matrices by diamatrix and recorded all possible coefficients. They included $d_0, d_1, d_2, d_3, d_4, d_5, d_6, d_8, d_9$ (with positive or negative signs) and non-trivial coefficients. All observed non-trivial coefficients are listed in (10).

Table 1
Non-unique solutions of (R, d) to the designed attack on distribution of values of $B_a[2^t \times i + b]$ for all possible i depending on t

Value of t	Values of R	d and number of solutions
1	2 and 255	$d, d + 128$: 2 solutions
2	2, 37, 51, 85, 172, 206, 220, 255	$d, d + 128$: 2 solutions
3	2, 15, 17, 37, 51, 85, 98, 99, 113, 114, 116, 117, 123, 124, 133, 134, 140, 141, 143, 144, 158, 159, 172, 206, 220, 240, 242, 255	$d, d + 128$: 2 solutions
4	2, 15, 17, 37, 44, 51, 85, 98, 99, 113, 114, 116, 117, 123, 124, 133, 134, 140, 141, 143, 144, 158, 159, 172, 206, 213, 220, 240, 242, 255	$d, d + 128$: 2 solutions

Non-trivial:

$$\begin{array}{ll} c_1 = d_1 + d_8 + d_7 + 2d_6 - d_9, & c_2 = 2d_0 + d_7 + d_3 + d_5 - d_8 \\ c_3 = 2d_1 + d_8 + d_7 + d_6 - d_9, & c_4 = 2d_2 + d_8 + d_9 + d_7 - d_4 \\ c_5 = d_7 + 2d_8 + d_9 + d_4 - d_0, & c_6 = d_0 + d_7 + d_3 + d_5 \\ c_7 = d_1 + 2d_8 + d_7 + d_6 - d_3, & c_8 = d_2 + 2d_8 + d_9 + d_7 - d_5 \\ c_9 = d_7 + d_8 + 2d_9 + d_4 - d_1, & c_{10} = d_0 + d_7 + 2d_3 + d_5 - d_8 \\ c_{11} = d_1 + d_8 + d_7 + d_6, & c_{12} = d_2 + d_8 + 2d_9 + d_7 - d_6 \\ c_{13} = d_2 + d_8 + d_9 + d_7, & c_{14} = d_0 + d_7 + d_3 + 2d_5 - d_8 \\ c_{15} = d_7 + d_8 + d_9 + d_4, & c_{16} = d_7 + d_8 + d_9 + 2d_4 - d_2 \end{array} \quad (10)$$

Now, we can consider the constructed attack *successful* if it finds at most two possible pairs of parameters (R, d). We can estimate the number of *unsuccessful* attacks as the number of all vectors $(d_0, d_1, \dots, d_9)$ that result in all above coefficients (including d_i) being either odd or equal to $32 \times k_i$ with integer k_i.

We conducted a computational experiment to find the value of the estimate for number of unsuccessful attacks. As we look for values that are either odd or equal to $0(\mathrm{mod}32)$, we only need to find values modulo 32. The set of possible values for d_7 is only odd numbers from 1 to 31, all other variables go through values 0 and all odd numbers from 1 to 31. Based on the constraints of non-zero diadeterminant, we have three extra conditions: $d_0 = 1 + d_7 + d_8 + d_3 + d_5(\mathrm{mod}\ 2)$, $d_1 = 1 + d_7 + d_8 + d_9 + d_6(\mathrm{mod}2)$, $d_2 = 1 + d_7 + d_8 + d_9 + d_4(\mathrm{mod}\ 2)$.

For each set of values we calculate coefficients and check whether they are odd or equal to $0\ (\mathrm{mod}\ 32)$. We found that the total amount of such solutions is 912.

Theorem 7: *Let the values $d_3, d_4, d_5, d_6, d_8, d_9$ be random independent and uniformly distributed in integers from 0 to 255. Let d_7 be random independent from others odd integer. Let x_0, x_1, x_2 be random independent and uniformly distributed even integers from 0 to 254. Let $d_0 = x_0 + (1 + d_7 + d_8 + d_3 + d_5(mod2)), d_1 = x_1 + (1 + d_7 + d_8 + d_9 + d_6(mod2)), d_2 = x_2 + (1 + d_7 + d_8 + d_9 + d_4(mod2))$. Then the probability of an unsuccessful attack of the modified SQUARE on one round of O'zDST 1105:2009 algorithm satisfies*

$$P(Attack = Unsuccessful) \leq \frac{57}{2^{42}}. \quad (11)$$

Proof: Using results of the computational experiment we can find the frequency of cases in which all coefficients of multiplication by the formed diamatrix are either odd or equal to $0(\mathrm{mod}32)$ as

$$\frac{8^{10} \times 912}{2^{76}} = \frac{2^{30} \times 2^4 \times 57}{2^{76}} = \frac{57}{2^{42}}$$

Here, each number $d_i = 32 \times k_i + a_i$, where $0 \le k_i \le 7, 0 \le a_i \le 31$. Thus, for each unique element of the diamatrix we get 8 cases of k_i. Our experiment calculated all possible vectors of $(a_0, a_1, \dots, a_9)$, so, we multiply the answer of the experiment by 8^{10} where 10 is the number of dimensions. As per denominator, we have 4 constraints on diadeterminant (all 4 multiplicands must be odd). Thus, the total number of different diamatrices with odd diadeterminants is $2^{80-4} = 2^{76}$.

Not all such cases correspond to unsuccessful attack, therefore, $P(Attack = Unsuccessful) \le \frac{57}{2^{42}}$.

We will analyze a reduced version of the algorithm, with the number of rounds limited to 1 and 2. To simplify the scenario, the parameter C_i was not used to calculate S-boxes in the first test.

The set of input texts is $\{P_j = (j, 0, \dots, 0) | j = \overline{0,255}\}$, the set of respective encrypted texts is $\{E_j^e\}$ where e is the number of rounds. Each input text contains 32 bytes, with the first byte being "active." After encryption, a frequency table is constructed, recording the number of times each value at each position occurs in all encrypted texts E_j^e. We focus on positions u where the corresponding number equals 2^t for q ranging from 1 to 7. "Active" positions exhibit all 256 possible values in SQUARE cryptanalysis. However, the algorithm here reduces the number of distinct values by at least half due to the parity preservation.

Now, we look at all determined "active" positions and try to find such pair of valid values of R and d that would produce S-box with similar properties: either all numbers that are mapped to the values we have in $\{E_j^e\}$ are even, or all odd. We count the number of successes for each pair of values of R and d, and select one with the highest match. If we did not use C_i in calculations, the largest value of matches will be equal to the number of different values recorded for this position among encrypted texts. We used the key from the Appendix of [1] and attacked exactly two rounds to retrieve secret parameters. The numbers below represent the amount of different values we got for each position:

$$\begin{pmatrix} 161 & 64 & 1 & 100 \\ 157 & 128 & 1 & 256 \\ 100 & 256 & 1 & 97 \\ 106 & 1 & 100 & 151 \\ 128 & 106 & 101 & 52 \\ 1 & 97 & 63 & 106 \\ 113 & 58 & 99 & 63 \\ 1 & 177 & 113 & 1 \end{pmatrix} \quad (12)$$

The correct values were $d = 19, R = 1$. We got them when the highest match was equal to the recorded frequency. In Table 2 Other cases reflect the reduction of values caused by the first round. Next, we applied the same approach to the case when we used parameter C_i to generate S-boxes.

Table 2
Calculation of the highest match for different S-boxes based on values R, d in two-round attack on the last round (no C_2 was used)

Coordinates	Match	Frequency	R value	d value
row 0, column 1	64	64	1	19
row 1, column 1	79	128	35	59
row 4, column 0	82	128	227	243

$$\begin{pmatrix} 163 & 64 & 1 & 102 \\ 159 & 128 & 1 & 256 \\ 102 & 256 & 1 & 100 \\ 106 & 1 & 102 & 151 \\ 128 & 106 & 96 & 54 \\ 1 & 100 & 64 & 102 \\ 112 & 58 & 106 & 62 \\ 1 & 161 & 112 & 1 \end{pmatrix} \quad (13)$$

Similarly, we successfully got secret values in 3. In Table 3

Table 3
Calculation of the highest match for different S-boxes based on values R, d in two-round attack on the last round (C_2 was used)

Coordinates	Match	Frequency	R value	d value
row 0, column 1	59	64	1	19
row 1, column 1	79	128	238	59
row 4, column 0	82	128	107	243
row 5, column 2	60	64	1	19

After we get the correct S-box for the second round (accurate to the parameter L_2 which is impossible to determine on this stage), we can proceed with the attack on the first round. We can now decrypt all texts using the chosen S-box. If we now choose one of the texts and subtract it from all others, we can eliminate linear value L_2 from consideration.

Next, we can apply inverse Sur and consider all values in the active position r as $q_r(v_j - v_0)$ where q_r is the coefficient of the diamatrix. Here, we look into all active positions where our approach from the last step was unsuccessful. Thus, coefficient q_r must be odd. Values v_j are the values in these positions that we get after adding key in round 2.

Let us choose a non-zero value among all texts after subtraction. We can now divide all other values by it to exclude value q_r from consideration. We will get following formula:

$$\frac{t_0 - t_j}{t_0 - t_1} = \frac{(y_0 \oplus k_r) - (y_j \oplus k_r)}{(y_0 \oplus k_r) - (y_1 \oplus k_r)} \quad (14)$$

Here, values t_i were obtained on deciphering B_2 in the position r of all encrypted texts, values y_i correspond to the results we get after the first round of the algorithm (before adding the round key). Now, we can use a brute force over y_1, y_0, k_r and in each case calculate matches for parameters R_1, d_1. In this case we are sure that all remaining active positions must provide us with the same pair for the highest match. It requires less than 2^{43} operations. This approach can be also used for three rounds of the algorithm but with more calculations.

5. Discussion and Conclusion

The cryptographic security of the analyzed algorithm should be revised according to the constructed attack. We proved that the attack is successful not only theoretically but also practically on a reduced algorithm. The simplest modification of the design requires use of finite field for multiplication of diamatrices instead of operations in a ring modulo 256.

Current Uzbekistan standard of encryption using block ciphers has two algorithms: O'zDSt 1105:2009 and GOST 28147-89 [2]. As a result of our investigation, the first one is not secure to be used for confidential information.

The probability of the constructed modified SQUARE attack being successful is very high according to Theorem 7. It is larger than $1 - 1.3 \times 10^{-11}$.

Conflict of interest

Authors declare no conflict of interest.

References

[1] The data encryption algorithm O'zDST 1105: State Standard of Uzbekistan (official publication), approved and put into effect by the decree of the Uzbek Agency for Standardization, Metrology and Certification, dated Sept. 28, 2009, No. 05-163 (2009). [Online]. Available: https://tace.uz/docs/OzDSt_1105.pdf

[2] I. Dinur, O. Dunkelman, and A. Shamir, "Improved attacks on full GOST," in Fast Software Encryption, Lecture Notes in Computer Science, vol. 7549, pp. 9–28 (2012). doi: 10.1007/978-3-642-34047-5_2.

[3] B. Abdurakhimov, Z. Khudoykulov, O. Allanov, and I. Boykuziev, "Algebraic cryptanalysis of O'zDSt 1105:2009 encryption algorithm," in Proc. Int. Conf. on Information Science and Communications Technologies (ICISCT), pp. 1–7 (2020).

[4] J. Daemen, L. Knudsen, and V. Rijmen, "The block cipher Square," in Fast Software Encryption — FSE 1997, Lecture Notes in Computer Science, vol. 1267, E. Biham, Ed. Springer, pp. 149–165 (1997).

[5] S. Shahapure, V. Sule, and R. D. Daruwala, "Variation and security enhancement of block ciphers by embedding," *J. Discrete Math. Sci. Cryptogr.,* vol. 22, no. 2, pp. 151–160 (2019). doi: 10.1080/09720529.2019.1576336.

[6] K. Achkoun, C. Hanin, and F. Omary, "SPF-CA: A new cellular automata-based block cipher using key-dependent S-boxes," *J. Discrete Math. Sci. Cryptogr.,* vol. 23, no. 8, pp. 1529–1544 (2019). doi: 10.1080/09720529.2019.1649031.

[7] M. Kumar, "Full-round differential attack on DoT block cipher," *J. Discrete Math. Sci. Cryptogr.,* pp. 1–13 (2022). doi: 10.1080/09720529.2021.1961905.

[8] A. Ikramov, M. Aripov, and G. Juraev, "SPONGE structure in the basis of a new stream cipher," AIP Conf. Proc., vol. 2781, no. 1, p. 020045 (2023). doi: 10.1063/5.0144775.

[9] A. Ikramov and G. Juraev, "The complexity of testing cryptographic devices on input faults," in Network and System Security (NSS 2021), Lecture Notes in Computer Science, vol. 13041, M. Yang, C. Chen, and Y. Liu, Eds. Cham: Springer (2021). doi: 10.1007/978-3-030-92708-0_12.

Received May, 2024