Success Probability of The First Attack on Full Round GOST

Orhun Kara, Ferhat Karakoç

***Abstract*—The GOST encryption function is the Russian encryption standard, which is a block cipher of 64-bit block and 256-bit key size. The first attack on the full round GOST is a kind of reflection attack given in [15]. This attack does not work for all the keys. It consists of two steps. The first step is the identification step that determines if the attack works for the given key. The weak key is identified by searching a special type of fixed points. If the encryption function possesses such fixed points then the second step is carried out. This step is the key recovery attack and implemented only if the key is identified as a weak key.**

In this paper, we analyze the success rate of the key recovery step of the attack given in [15]. We compute the success rate and come to a conclusion that the second step is successful with a probability of almost one half. In addition, we implement the second part of the attack for some small scale GOST toy ciphers and verify the theoretical results we come up with. Moreover, we introduce a new key recovery algorithm for the second step and improve the time complexity slightly.

***Index Terms*—block cipher, self-similarity, reflection attack, GOST, weak key**

I. Introduction

The GOST cipher, the Russian encryption standard, is a Feistel network of 64-bit block and 256-bit key size [27]. It has a relatively very simple key schedule. A given key is divided into 8 parts and each part is used as a subkey in a palindromic order. This is a similarity property of the cipher; the similarity due to the palindromic order of the subkeys occurs between round functions of the encryption and those of the decryption. However, self-similarity attacks such as slide attacks or related key attacks could not be mounted on the full round GOST since these attack techniques examine the similarity between round functions of the encryption direction.

One of the self-similarity attack, the reflection attack, exploits the similarities between the round functions of the encryption direction with the corresponding round functions of the decryption direction by means of unexpectedly many number of fixed points. These fixed points are due to the palindromic structure of the round keys [18], [15], [16], [17]. As one of the applications, it succeeded to be the first attack on the full round GOST [15]. The attack does not work for all the keys. It consists of two steps. The former step is the identification step. The key is identified as a weak key in this step. The latter step is the key recovery attack and implemented only if the key is identified as a weak key.

Manuscript received March 26, 2012.

Orhun Kara and Ferhat Karakoç are with TÜBİTAK BİLGEM Gebze, 41470 Kocaeli Turkey, e-mails: {orhun,ferhatk}@uekae.tubitak.gov.tr

The weak key identification process in [15] makes use of the special type of fixed points. In the attack, it has been expected that a special type of fixed points may occur due to weak keys and the second step of the attack has been mounted if the encryption function has possessed such fixed points. On the other hand, there is a probability that these fixed points may exist by chance also. However, this situation was not examined and hence the probability of recovering the key in the second step has been left out in [15]. In this paper, we analyze the success rate of the second step of the attack. We compute the success rate and come to a conclusion that the second step is successful with a probability of almost one half. In addition, we implement the second part of the attack for some small-scale GOST toy ciphers and verify the theoretical results we come up with. Furthermore, we introduce a new key recovery algorithm for the second step. This attack is also a guess and determine type attack. However, we guess the internal states of the cipher instead of subkey bits. We also take advantages of the meet-in-the middle technique while determining the key bits. Altogether, we gain a minor improvement in the time complexity.

The paper is organized as follows. We introduce the notation in the next section. Then we give a brief overview about self-similarity attacks in Section III. Section IV is about a description of GOST cipher. Then the reflection attack on full round GOST is given in Section V. We analyze the success rate of the attack in Section VI. Experimental results are given in Section VII. We introduce a new key recovery attack when the key is identified as a weak key in Section VIII. We conclude the paper with Section IX.

II. Notation

We follow the notation and definitions given in [15]. Let us denote the ith round function of an encryption function E_K with n-bit block length as F_{k_i} for $i = 1, ..., r$ where r is the number of rounds and $k_1, ..., k_r$ are the round keys. Let us define the partial encryption function $F_K[i, j]$ as the encryption starting from round i and finishing at round j. That is,

$$F_K[i, j] = F_{k_j} \circ \cdots \circ F_{k_i} \text{ for } 1 \leq i < j \leq r. \quad (1)$$

These functions are called *intermediate functions*. Let $U_K(i, j)$ be the set of fixed points of the function $F_K[i, j]$. More explicitly,

$$U_K(i, j) = \{x \in GF(2)^n : F_K[i, j](x) = x\}.$$

III. Self-Similarity Attacks

Self-similarity attacks exploit the similarities between building blocks of a cipher iterated several times such as round functions of a block cipher. There are three main kinds of self-similarity attacks: Slide attacks [7], [8], related key attacks [1], [23] and recently discovered reflection attacks [15], [16], [17], [18]. Related key attacks proposed by Biham [1] and independently by Knudsen [23] are based on the powerful assumption that the attacker knows a relation between several keys and can access encryption function with these related keys.

Reflection attack was first presented at a local symposium [16] and on the web [17]. Then, several applications were introduced in [18], [15]. Reflection attack differs from the previous self-similarly attacks in the sense that it exploits certain similarities of some round functions of the encryption direction with the corresponding round functions of the decryption direction. Therefore, it particularly works on some involutionary ciphers such as some Feistel networks and its assumptions can be much weaker than the assumptions of previous self-similarity attacks for some cases. This enables the reflection attack to be mounted on some ciphers even with complicated key schedules.

IV. Brief Description of GOST

GOST, the Russian encryption standard [27], is a 32 round 64 bit Feistel network with 256 bit key. It has a simple key schedule: 256 bit key is divided into eight 32 bit words $k_0, ..., k_7$ and the sequence of round keys is given as $k_0, ..., k_7, k_0, ..., k_7, k_0, ..., k_7, k_7, k_6, ..., k_1, k_0$. The round key is involved by the modular addition in the round function. We do not consider details of the round function. We only assume that it is bijective.

Denote the first eight rounds of GOST as $F_K[1,8]$. Note that $F_K[1,8]$ ends with a swap operation. Then, the GOST encryption function is given as

$$E_K(x) = F_K[8,1] \circ S \circ F_K^3[1,8](x),$$

where S is the swap operation of the Feistel network and $F_K[8,1]$ is the inverse of $F_K[1,8]$.

A. Security Status of GOST

The first single key recovery attack on full round GOST is the reflection attack [15]. Before the discovery of the reflection property of GOST, there was no known attack on a single key which is better than the exhaustive search for full-round GOST. A related key differential cryptanalysis is shown in [20]. The attack is impractical for properly chosen S-boxes. A slide attack has been mounted on 20 round GOST$\oplus$, a variant of GOST defined in [8]. This attack uses 2^{33} known plaintexts and 2^{65} memory space and the key is recovered in 2^{70} encryptions. Another related key differential attack given in [26] has been mounted on 21 round GOST. This attack has a data complexity of 2^{56} chosen plaintexts. A related key differential attack that uses the idea of Seki and Kaneko in [26], is mounted on GOST [22]. The attack is on full-round GOST and recovers 12 bits of the key with 2^{35} chosen plaintexts in 2^{36} steps. However, the attack is based on a powerful assumption that the attacker knows that the two related keys differ in only eight specific bits. Another study by Biham, Dunkelman and Keller is the improved slide attack mounted on GOST [2]. In this attack, they firstly recover the key for 24-round reduced GOST in 2^{63} steps and then extend the attack to 30-round GOST with a workload of 2^{254} encryptions by using almost the entire code book.

Using the reflection property in [15], Isobe et.al. recently proposed an attack on full-round GOST which works for all keys. This attack requires 2^{32} data, 2^{64} memory, and 2^{224} time [14]. Also, there is another recent attack on full-round GOST using a different technique with 2^{64} data, 2^{64} memory and 2^{226} time [9]. Dinur et.al. improved the attack done by Isobe reducing the time complexity to 2^{192} [10].

V. Reflection Attack on Full Round GOST

In this section, we recall the reflection attack on full-round GOST given in [15].

Assume there exists x such that x is a fixed point of both $F_K[1,8]$ and the swap operation S, i.e., $F_K[1,8](x) = x$ and $S(x) = x$. Then, x is also a fixed point of the encryption function E_K. This observation leads to the following attack: Encrypt all 2^{32} plaintexts whose left and right halves are equal and collect the fixed points in a set, say U_E. If U_E is empty, then the attack is not applicable. Otherwise, for any x in U_E solve the equation $F_K[1,8](x) = x$ for K. Note that there are 2^{192} solutions and each of the solutions may be obtained by guessing $k_0, k_1, ..., k_5$ and then determining k_6 and k_7.

Guessing $k_0, k_1, ..., k_5$, we construct a two-round Feistel network with unknown keys k_6 and k_7 and an input-output pair given as $(F_K[1,6](x), x)$. Then, solving the system for k_6 and k_7 is straightforward since the round functions F_{k_6} and F_{k_7} are bijective and their outputs are known. By taking the inverses of F_{k_6} and F_{k_7}, we obtain the inputs and then k_6 and k_7. Consequently, we obtain 2^{192} candidates for the key by solving $F_K[1,8](x) = x$. We recover the correct key by searching over all the candidates by roughly 2^{192} encryptions. We solve $F_K[1,8](x) = x$ for each $x \in U_E$. However, it is most likely that U_E is empty if there exists no fixed point of $F_K[1,8]$ with the equal halves. On the other hand, the expected number of fixed points is one and the probability that any arbitrary value is a fixed point of S is 2^{-32}. Hence, the number of keys satisfying that $\exists\, x$ such that $F_K[1,8](x) = x$ and $S(x) = x$ is roughly 2^{224}.

VI. Success Rate of The Attack

The reflection attack on full-round GOST is given in [15]. However, the success rate of the attack is left out. Indeed, if the function $F_K[1,8]$ has a fixed point of the form $x = (a,a)$ where the former half of x is equal to its latter half, it is definitely a fixed point of the encryption function E. Nevertheless, the attack makes use of the opposite direction of the statement. That is, the attack is mounted if there is a fixed point of E, assuming that it is (probably) a fixed point for also $F_K[1,8]$. In this paper, we examine this direction and find the probability that any fixed point of E is also a fixed point of

$F_K[1,8]$. Let us remark that the attack is not successful for those fixed points of E which are not fixed points of $F_K[1,8]$.

Let the set of the fixed points of the form $x=(a,a)$ of $F_K[1,8]$ be U_F. Recall that U_E is the set of the fixed points of E of the form $x=(a,a)$.

Then, the success rate, $\Pr(S)$, of the attack is given as the conditional probability that U_F is nonempty given that U_E is nonempty. That is,

$$\Pr(S)=\Pr(U_F\neq\emptyset|U_E\neq\emptyset).$$

The following statement gives the success rate explicitly.

Theorem 1: Assume the encryption function E behaves as a random permutation when the function F_K has no fixed point of the form $x=(a,a)$. Then the sucess probability is given as

$$\Pr(S)=\frac{1}{1+(1-2^{-64})^{2^{32}}}\approx\frac{1}{2-2^{-32}}.$$

Proof: We have

$$\Pr(S)=\Pr(U_F\neq\emptyset|U_E\neq\emptyset)$$

which leads to

$$\Pr(S)=\frac{\Pr(U_F\neq\emptyset)}{\Pr(U_E\neq\emptyset)}$$

since U_F is a subset of the set U_E. On the other hand

$$\Pr(U_F\neq\emptyset)=1-(1-2^{-64})^{2^{32}}$$

and $\Pr(U_E\neq\emptyset)$ is given as

$$\Pr(U_F\neq\emptyset)+\Pr(U_F=\emptyset)\Pr(U_E\neq\emptyset)|U_F=\emptyset).$$

Assuming that E is a random permutation when F_K has no fixed point of the form $x=(a,a)$, we have

$$\Pr(U_E\neq\emptyset)|U_F=\emptyset)=1-(1-2^{-64})^{2^{32}}.$$

Hence, we calculate $\Pr(U_E\neq\emptyset)$ as

$$1-(1-2^{-64})^{2^{32}}+(1-2^{-64})^{2^{32}}(1-(1-2^{-64})^{2^{32}}),$$

which yields to the probability

$$\Pr(S)=\frac{1-(1-2^{-64})^{2^{32}}}{(1-(1-2^{-64})^{2^{32}})(1+(1-2^{-64})^{2^{32}})}$$

concluding the proof. ■

Let us remark that the success probability given in Theorem 1 is very close to one half since the value $(1-2^{-64})^{2^{32}}$ is approximately $1-2^{32}2^{-64}$ which is almost one. One interesting result is that if the encryption function E_K is a random permutation when the function F_K has no fixed point of the form $x=(a,a)$, it is itself cannot be a random permutation in general. Because, the probability that U_E is not empty is twice as large as the probability for a random permutation. The following corollary states this phenomena formally.

Corollary 1:

$$\begin{aligned}\Pr(U_E\neq\emptyset) &= (1-(1-2^{-64})^{2^{32}})(1+(1-2^{-64})^{2^{32}})\\ &\approx 2(1-(1-2^{-64})^{2^{32}}).\end{aligned}$$

VII. Experimental Results

We deduce the result in Theorem 1 under the assumption that the encryption function E is a random permutation when the function F_K has no fixed point of the form $x=(a,a)$. We have tested this assumption by conducting some experiments on some shrunk versions of GOST. We have verified that the success rate is indeed around one half. Also we verify that the encryption function itself is not random since the probability that U_E is not empty is twice as high as expected.

We devise 4 versions of GOST with block sizes 16, 20, 24 and 28 bit lengths. The number of rounds is fixed to 32 for any block length and we use $n\times 4$ bit key for the n bit block length. The key is divided into 8 equal parts $k_0,...,k_7$ and incorporated into the round function as for the original GOST function. The keys are produced at random on a desktop PC. Then, they are used in the rounds in the order $k_0,...,k_7,k_0,...,k_7,k_0,...,k_7,k_7,k_6,...,k_1,k_0$.

We make $100\times 2^{n/2+4}$ number of experiments for the block length n, and count the number of keys where there is a fixed point of the form (a,a) and the number of the weak keys, that is, the keys where the attack works successfully. The results are depicted in Table 1.

The first column in Table 1 gives the block length of the GOST version. The second column is the number of experiment done. The third column is the number of keys where there is a fixed point of the form (a,a). The last column is the number of weak keys, that is, the number of the keys where there is a fixed point of the form (a,a) for the first 8-round of the related GOST version. For each block length, we adjust the expected number of fixed points of the form (a,a) for the encryption to be 3200 and the expected number of weak keys to be 1600 in the theory given in the previous section performing different number of experiment. The number of keys used in the experiment for the 16-bit block length is selected as 100×2^{12} to reach 1600 which is the number of keys which have a fixed point for the first 8 rounds because the probability of having a fixed point for the first 8 rounds for a key is $2^{-16}\times 2^8=2^{-8}$ and using 100×2^{12} different keys we expect to find $100\times 2^{12}\times 2^{-8}=1600$ weak keys. The number of experiment for other block length are selected in a similar way. As being seen from the table, the experimental results are as expected. That is, the success ratio is around 1600/3200, which is one half.

TABLE I
THE EXPECTED NUMBER OF FIXED POINTS AND WEAK KEYS ARE 3200 AND 1600 RESPECTIVELY

Block length	Number of keys used in experiment	Number of keys which have a fixed points for E_K	Number of weak keys	Success rate
16 bits	100×2^{12}	3153	1556	0.493
20 bits	100×2^{14}	3255	1618	0.497
24 bits	100×2^{16}	3193	1598	0.501
28 bits	100×2^{18}	3229	1588	0.492

The experimental results verify that the probability that the

GOST encryption function has a fixed point of the form (a, a) is approximately twice as large as a random permutation. For a random permutation, the number of keys producing fixed points of the form (a, a) is around 1600 for each block length in the experiments.

VIII. Another Attack

We slightly improve the attack given in Section V by means of guess and determine and meet-in-the-middle techniques together. In [15], the candidate key space is reduced to 2^{192}, utilizing a guess and determine type attack. Then, the brute force attack is mounted to find the correct key among the 2^{192} key candidates. Thus, the complexity is given as approximately 2^{192} encryptions. In this attack, we reduce the size of the candidate key space to 2^{192} as in the original attack. What is more is that, we are able to cluster the candidate lists for the first 5 and last 3 rounds keys. Thus, we can reduce the brute force complexity using the clustering. The details of the attack as follows.

We first introduce a 3-round meet-in-the-middle technique used in the attack. The technique is depicted as Algorithm 1. The time complexity of Algorithm 1 is 3×2^{32} F calculations

Algorithm 1 3-round meet-in-the-middle

1: One input-output pair for 3 rounds is given and a list of candidate keys of 3 rounds will be returned.
2: **for all** Guess of 1-st round key **do**
3: Calculate the right most 32-bit of the output of 1-st round and store it into Table C.
4: **end for**
5: **for all** Guess of 3-rd round key **do**
6: Calculate the left most 32-bit of the input of 3-th round and check the value in Table C. If there is a match calculate the 2-nd round key inverting F function and store 2^{32} candidate values for the triple (k_1, k_2, k_3) into the list L.
7: **end for**
8: Output L.

and we need 2^{32} memory for Table C and L used in the algorithm.

We mount an attack on full-round GOST given in Algorithm 2.

In the Algorithm after Step 5 there will be 2^{64} key candidates under one guess of the outputs of 3-th and 5-th rounds. Because of the 64-bit sieving in Step 11 the number of survived candidate key will be 1 on the average. The probability that the survived key will not be eliminated in the checking process in Step 12 is 2^{-192}. The number of all possible guess for the outputs is 2^{128}. Thus the number of candidate keys after the execution of the attack will be 2^{-64} which means that all wrong keys are eliminated and the correct key will be recovered.

The data complexity of the attack as follows. 2^{32} chosen plaintexts having equal left and right halves are used to find a fixed point for the cipher. In the case of finding a fixed point the attack given in Algorithm 2 is executed using the fixed

Algorithm 2 Attack on full-round GOST

1: The 64-bit value x such that $F_K[1,8](x) = x$ is given.
2: **for all** Guess of the outputs of 3-rd and 5-th rounds. **do**
3: Calculate k_3 and k_4 using the output of 3-rd and 5-th rounds.
4: Apply Algorithm 1 giving x and the output of 3-rd round. Store the output of the algorithm which is a list of 2^{32} candidates for the round keys (k_0, k_1, k_2) in Table A.
5: Apply Algorithm 1 giving the output of 5-th round and x. Store the output of the algorithm which is a list of 2^{32} candidates for the round keys (k_5, k_6, k_7) in Table B.
6: For a different input-output pairs of the Encryption algorithm do the followings.
7: **for all** Values of (k_0, k_1, k_2) in Table A **do**
8: Calculate the output of the 5-th and the input of 28-th rounds making 5-round partial encryptions and 5-round partial decryptions respectively.
9: **for all** Values of (k_5, k_6, k_7) in Table B **do**
10: Calculate the output of the 16-th and the input of 17-th rounds making 11-round partial encryptions and 11-round partial decryptions respectively.
11: **if** The calculated values of the output of the 16-th round and the input of the 17-th round are equal **then**
12: Check the key using 3 different input-output pairs of the cipher.
13: **end if**
14: **end for**
15: **end for**
16: **end for**

point and arbitrary 4 input-output pairs of the cipher. Here we assume that the fixed point for the cipher is also a fixed point for first 8 rounds with a probability of $1/2$. As a result, we use 2^{32} chosen plaintext to find a fixed point and the fixed point with arbitrary 4 input-output pairs of the cipher are used in the Algorithm.

The time complexity of the attack can be computed as follows. For a guess of the outputs of 3-th and 5-th rounds we do 2^{32} operations in step 4 and 5 because the complexity of Algorithm 1 is approximately 2^{32}. We perform 2^{32} times 10-round partial encryptions in step 8 using the keys in table A and k_3 and k_4 calculated in step 3. Also, we do 2^{64} times 22-round partial encryptions using the keys in Table A and B, k_3 and k_4 in Step 10. This gives an advantage provided by the clustering of the round key candidates during the brute force. There is a 64-bit sieving in Step 11, so there will be only one key candidate for a guess of the outputs of 3-th and 5-th rounds on the average. The number of operations in Step is 3 encryptions. Thus, the number of operations is approximately $2^{64} \times 22$ F calculations which costs roughly $2^{63.46}$ encryption operations for a guess of the outputs of 3-th and 5-th rounds. The number of all possible guess for the outputs is 2^{128}. As a result, the time complexity of Algorithm 2 is around $2^{191.46}$

encryption operations. We use approximately 2^{32} memory in the attack.

IX. Conclusion

We have computed the success rate of the key recovery step of the first attack mounted on the full GOST. We have seen that the success rate is around one half. In addition, we have conducted several experiments to verify the calculations. On the other hand, we have proved that the encryption function is not random when there is a fixed point of the form (a, a). Moreover, we have mounted another attack to recover the key whenever the key is identified as a weak key.

Acknowledgment

We would like to thank the anonymous reviewers and Mehmet Sabır Kiraz for their helpful comments.

References

[1] E. Biham. New Types of Cryptanalytic Attacks Using Related Keys. *J. of Cryptology*, Vol.7, pp.229-246, 1994.
[2] E.Biham, O. Dunkelman and N. Keller. Improved Slide Attacks, In *Proc. FSE'07*, LNCS 4593, pp. 153-166, Springer, 2007.
[3] E.Biham, O. Dunkelman and N. Keller. Related-Key Boomerang and Rectangle Attacks, In *Proc. of EUROCRYPT 2005*, LNCS 3494, pp. 507-525, Springer, 2005.
[4] E.Biham, O. Dunkelman and N. Keller. New Cryptanalytic Results on IDEA, *Proc. of ASIACRYPT 2006*, LNCS 4284, pp. 412-427, Springer, 2006.
[5] E.Biham, O. Dunkelman and N. Keller. A Simple Related-Key Attack on the Full SHACAL-1. In Proc. CT-RSA'07, LNCS 4377, pp. 20-30, Springer, 2007.
[6] E.Biham, O. Dunkelman and N. Keller. A Unified Approach to Related-Key Attacks. In *Proc FSE'08*, LNCS 5086, pp73-96, Springer, 2008.
[7] A. Biryukov and D. Wagner. Slide Attacks. In *Proc. FSE'99*, LNCS 1636, pp.245-259, Springer, 1999.
[8] A. Biryukov and D. Wagner. Advanced Slide Attacks. In *Proc. EUROCRYPT 2000*, LNCS 1807, pp.589-606, Springer, 2000.
[9] N. Courtois, G.V. Bard and D.Wagner. Algebraic and Slide Attacks on KeeLoq. In *Proc FSE'08*, LNCS 5086, pp73-96, Springer, 2008.
[10] I. Dinur, O. Dunkelman, and A. Shamir. Improved Attacks on Full GOST. IACR Cryptology ePrint Archive, 2011. http://eprint.iacr.org/2011/558.
[11] O. Dunkelman, N. Keller, J. Kim. Related-Key Rectangle Attack on the Full SHACAL-1. In Proc. SAC'06 LNCS 4356, pp. 28-44, Springer, 2006.
[12] S. Furuya. Slide Attacks with a Known-Plaintext Cryptanalysis. In Proc. Information and Communication Security 2001, LNCS 2288, pp. 214-225, Springer, 2002.
[13] S. Hong, J. Kim, G. Kim, S. Lee, B. Preneel. Related-Key Rectangle Attacks on Reduced Versions of SHACAL-1 and AES-192, In *Proc. FSE'05*, LNCS 3557, pp.368-383, Springer, 2005.
[14] T. Isobe. A Single-Key Attack on the Full GOST Block Cipher. In *Proc. FSE 2011*, LNCS 6733, pp. 290-305, Springer, 2011.
[15] O. Kara. Reflection Cryptanalysis of Some Ciphers. In *Proc. Indocrypt'08*, LNCS 5365, pp. 294-307, Springer, 2008.
[16] O. Kara. Self-Similarity Analysis of Block Ciphers. In Proc. II. National Cryptology Symposium, METU Ankara 2006.
[17] O. Kara. Reflection Attacks on Product Ciphers. IACR Cryptography e-Print Archive. `http://eprint.iacr.org/2007/043`, 2007.
[18] O. Kara and C. Manap. A new class of Weak Keys for Blowfish, In *Proc. FSE'07*, LNCS 4593, pp. 167-180, Springer, 2007.
[19] B.S. Kaliski, R.L. Rivest and T. Sherman. Is DES a Pure Cipher? (Results of More Cycling Experiments on DES). In *Proc. CRYPTO'85*, LNCS 218, pp. 212-222, Springer, 1985.
[20] J. Kelsey, B. Schneier and D. Wagner. Key-Schedule Cryptanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES. In *Proc. CRYPTO'96*, LNCS 1109, pp. 237-251, 1996.
[21] J. Kim, S. Hong, B. Preneel. Related-Key Rectangle Attacks on Reduced AES-192 and AES-256. In *Proc. FSE'07*, LNCS 4593, pp. 225-241, Springer, 2007.
[22] Y. Ko, S. Hong, W. Lee, S. Lee and J. Kang. Related Key Differential Attacks on 27 Rounds of XTEA and Full-Round GOST. In *Proc. FSE'04*, LNCS 3017, pp.299-316, Springer, 2004.
[23] L. Knudsen. Cryptanalysis of LOKI91. In *Proc. of AUSCRYPT'92*, LNCS 718, pp 196-208, Springer, 1993.
[24] J.H. Moore and G.J. Simmons. Cycle Structure of the DES with Weak and Semi-Weak Keys. In *Proc. CRYPTO'86*, LNCS 263, pp.9-32, Springer, 1986.
[25] J.H. Moore and G.J. Simmons. Cycle Structure of the DES for Keys Having Palindromic (or Antipalindromic) Sequences of Round Keys. *IEEE Transactions on Software Engineering*, pp. 262-273,No 13,1987.
[26] H. Seki and T. Kaneko. Differential Cryptanalysis of Reduced Rounds of GOST. In *Proc. SAC 2000*, LNCS 9743, pp. 315-323, Springer, 2000.
[27] I.A. Zabotin, G.P. Glazkov and V.B. Isaeva. Cryptographic Protection for Information Processing Systems. Cryptographic Transformation Algorithm. In *Government Standard of the USSR, GOST 28147-89*, 1989.