The ISC Int'l Journal of Information Security

July 2019, Volume 11, Number 2 (pp. 145–158)

http://www.isecure-journal.org

New Fixed Point Attacks on GOST2 Block Cipher with Memory Complexity Improvements

Siavash Ahmadi [1], and Mohammad Reza Aref [1,*]

[1] *Information Systems and Security Lab (ISSL), Department of Electrical Engineering, Sharif University of Technology*

ARTICLE INFO.

Article history:
Received: 14 July 2018
Revised: 6 June 2019
Accepted: 26 June 2019
Published Online: 31 July 2019

Keywords:
Cryptanalysis, Fixed Point Attack, GOST2 Block Cipher, Meet in the Middle.

ABSTRACT

GOST block cipher designed in the 1970s and published in 1989 as the Soviet and Russian standard GOST 28147-89. In order to enhance the security of GOST block cipher after proposing various attacks on it, designers published a modified version of GOST, namely GOST2, in 2015 which has a new key schedule and explicit choice for S-boxes. In this paper, by using three exactly identical portions of GOST2 and fixed point idea, more enhanced fixed point attacks for filtration of wrong keys are presented. More precisely, the focus of the new attacks is on reducing memory complexity while keeping other complexities unchanged as well. The results show a significant reduction in the memory complexity of the attacks, while the time complexity slightly increased in comparison to the previous fixed point attacks. To the best of our knowledge, the lowest memory complexity for an attack on full-round GOST2 block cipher is provided here.

1 Introduction

Block ciphers are one of the most important building blocks of many security protocols and in some situations, they are known as the security cornerstone of communication or storage systems. Therefore, ensuring the security of block ciphers is one of the most important subjects in the designing phase of block ciphers. Coincident with block cipher designing, cryptanalysis of them is also an important issue. In fact, the security provided by each block cipher can be measured by applying different methods of cryptanalysis or attacks on it, each attack results in time, memory, and data complexities for extracting the master key. These attacks can be categorized into distinguishing-based attacks such as linear and differential attacks [1–4], or non-distinguishing-based attacks such as meet in the middle (MITM) and biclique attacks [5–8] in the single key model.

In this paper, by a combination of an enhanced meet in the middle attack and fixed point property, we focus on the cryptanalysis of the Russian block cipher, GOST2, which is published after some security threats were found in the previous version of this block cipher, namely GOST. According to the literature, GOST block cipher was designed by the Soviet Union in the 1970s as an alternative to DES block cipher and accepted as a Russian standard block cipher [11]. The particular aspect of GOST standardization is the S-boxes which were not included in the standard, and hence anyone can deploy different sets of S-boxes. However, numerous cryptanalysis for GOST block cipher enforced the designers to publish the modified version of this block cipher, namely GOST2, and propose a new standard called "Kuznyechik" [12].

A considerable amount of literature has been pub-

* Corresponding author.
Email addresses: s_ahmadi@ee.sharif.edu (S. Ahmadi), arefsharif.edu (M.R. Aref)
ISSN: 2008-2045

Table 1. Cryptanalysis results on full GOST2

Type of attack	Time	Data	Memory	Reference
Reflection†	2^{192}	2^{32}KP	$2^{68.58}$	[28]
Impossible Reflection‡	$2^{254.34}$	2^{63}CP	$2^{166.58}$	[28]
Impossible Reflection‡	$2^{255.34}$	2^{64}KP	$2^{166.58}$	[28]
Fixed Point	2^{237}	2^{64}KP	$2^{138.15}$	[28]
Fixed Point	$2^{233}\star$	2^{64}KP	2^{196}	[29]
Fixed Point (Attack 1)	$2^{241.2}$	2^{64}KP	2^{82}	Section 4.1
Fixed Point (Attack 2)	$2^{246.9}$	2^{64}KP	2^{67}	Section 4.2

† Attack on 2^{224} keys

‡ Attack on $2^{256} - 2^{224}$ keys

⋆ The corrected time complexity is more than 2^{239} encryptions

lished for cryptanalysis of GOST. For instance, there are some related key differential attacks [13], reflection cryptanalysis [14, 15], differential attacks [16–20], self-similarity and black-box reduction attacks [21–23], meet in the middle attacks [24, 25], and some innovative attacks on GOST [26, 27]. However, the results on GOST2 are more limited and summarized in Table 1, such as reflection, impossible reflection, and fixed point attacks [28, 29]. To the best of our knowledge, the most efficient attack on GOST2, thus far, is the fixed point attack which can be applied on the full round cipher. However, due to the high memory and data complexities of this attack, it is not considered as a serious threat for GOST2.

In [25], the concept of 2-dimensional partial filtering for GOST is provided. Moreover, a new weakness for the fixed point attack on GOST2 is proposed in [28]. Here, by combining the concept of partial filtering and the weakness found for the fixed point attack on GOST2, new 3-dimensional fixed point attacks on GOST2 block cipher are proposed. The main advantage of the proposed attacks is their high reduction in memory complexity (see Table 1). The main ideas used for the new fixed point attacks on GOST2 are as follows:

- Using small intermediate variables to reduce time and memory complexities of the attack;
- Providing 8 parallel filters and paths by guessing carries and some other related bits, and store them in some hash tables;
- Considering an efficient precomputation phase as a primary filter to take more wrong keys;
- Using proper indexing for hash tables to archive an efficient implementation of the filtering parts;
- Using smart integration of parallel filters and paths and the primary filter to reduce total time complexity.

The rest of the paper is organized as follows. In Section 2, the preliminaries for basics of the fixed point attack, description of the GOST2 block cipher, and notations are provided. The main ideas of previous fixed point attacks on GOST2 are described in Section 3. Section 4 begins by summarizing the new ideas which can be utilized in a new fixed point attack. Afterward, the details of two new fixed point attacks on GOST2 block cipher are introduced and some discussions are provided. Finally, the work is concluded in Section 5.

2 Preliminaries

2.1 Basics of the fixed point property

Definition of fixed point property is as follows:

Definition 1. [Fixed point property]Let $p : \{0,1\}^b \to \{0,1\}^b$ be a pseudorandom permutation. The fixed point property states that there is an input $x \in \{0,1\}^b$ by which $x = p(x)$, namely *fixed point*.

Suppose any reduced-round encryption/decryption starting from an arbitrary internal state of round i to another internal state of round j is called a portion of a block cipher. According to Definition 1, considering any portion of a block cipher with b bits block length as a pseudorandom permutation g, with respect to the fixed point property, there will be a specific value x, with high probability, such that $x = g(x)$. This property is useful when some successive portions of a block cipher are identical to each other. In such cases, if the input of the first portion is the fixed point, then the output of that portion would be also the fixed point which is the input to the next portion, and so on. Therefore, the fixed point goes through these portions without any change until the final portion.

In the case of block ciphers, according to the randomness of the master key, we can consider the output of partial encryption as a random permutation. Therefore, when a block cipher cryptosystem with an unknown master key is provided, there will be a fixed point with high probability. Although, since the input strings are of size b bits the probability of finding such an input is 2^{-b}. Hence, one must consider all the 2^b inputs to ensure, one of them is the fixed point. By using this idea, the fixed point attacks on GOST2 block cipher are proposed.

2.2 GOST2 block cipher description

GOST2 is a 32-round Feistel block cipher with 64-bit block and 256-bit key sizes. Every 64-bit state of GOST2 is divided into two 32-bit left and right words, and each round of GOST2 consists of a round function followed by a swap. The equations for one round GOST2 are:

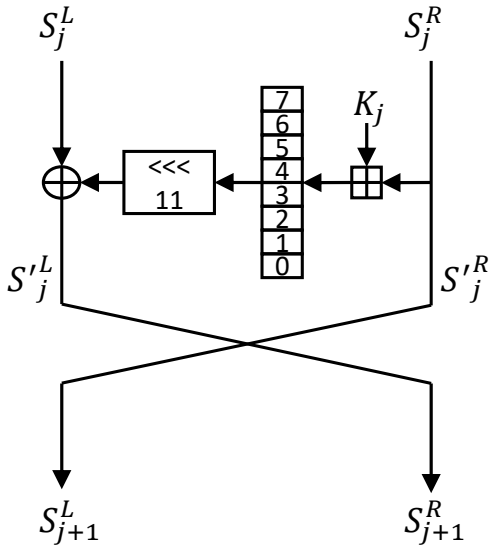

Figure 1. One round of GOST and GOST2 block cipher

$$
\begin{aligned}
&S_j^L||S_j^R = S_j \\
&S_{j+1}^R = S'^L_j = F(S_j^R, K_j) \oplus S_j^L \\
&S_{j+1}^L = S'^R_j = S_j^R \\
&S_{j+1} = S_{j+1}^L||S_{j+1}^R
\end{aligned} \tag{1}
$$

in which j is the round number starting from 0, S_j is the input state of the j^{th} round, S_j^R (S_j^L) is the right (left) 32-bit word of j^{th} round input state, S'^R_j (S'^L_j) is the right (left) 32-bit word of j^{th} round output state before the swap, F is the round function, K_j is the j^{th} round key, and $||$ denotes the concatenation. It should be noted that the last round does not have the swap. Also, the round function of GOST2 consists of a modular addition with the round key, passing through eight 4×4 S-boxes and an 11-bit left cyclic shift, which is shown in Figure 1.

GOST2 key schedule is simple. Consider the 256-bit master key $K = K^0||K^1||K^2||K^3||K^4||K^5||K^6||K^7$ with equal size of $K^i, 0 \leq i \leq 7$. Then, the key schedule of GOST2 is just a permutation of subkeys used in the master key as described in Table 2.

It is worthy to note that the S-boxes of GOST2 are concrete in contrast to the original GOST which left the choice of S-boxes open. Since our attack is independent of the structure of the S-boxes, we do not concentrate on their details. Interested readers can find details in [28].

2.3 Notations

The following notations are used in the rest of the paper:

- $X_j, (0 \leq j \leq 31)$ is utilized to emphasize that the round number of the variable X is j. In addition, X_{j_1,j_2} means (X_{j_1}, X_{j_2}). Also $X_{j_1-j_2}$ means $(X_{j_1}, X_{j_1+1}, ..., X_{j_2})$.
- MV_j is the matching variable placed in round j and it is equal to the right word of the round input state (or equivalently, stands for input of the j^{th} round function; $MV_j = S_j^R = S'^R_j$ as shown in Figure 1).
- $X[i_1-i_2]$ denotes i_1^{th} to i_2^{th} bits of the variable X, starting from zero and counting from right (LSB) to left (MSB). Hence, it is clear that the 11-bit left cyclic shift of the round function results in *incrementing* the values of bit numbers by 11. Also, $X[i_1 - i_2]$ means $X[i_1 - 31]$ and $X[0 - i_2]$, if $i_1 > i_2$. In addition, if i_1 *or* $i_2 > 31$, then it should be considered modulo 32.
- $X[i_1 - i_2]^{+w}$ denotes $X[(i_1 + w) - (i_2 + w)]$.
- $K^{i_1-i_2}$ denotes $(K^{i_1}, K^{i_1+1}, ..., K^{i_2})$.
- a_j^i is the input carry bit for i^{th} $(0 \leq i \leq 7)$ nibble/S-box in round j (see Figure 1). In addition, $a^i_{j_1,j_2}$ means $(a^i_{j_1}, a^i_{j_2})$, and $a^i_{j_1-j_2}$ means $(a^i_{j_1}, a^i_{j_2}, ..., a^i_{j_2})$. Also if $i > 7$ then it should be considered modulo 8. It is obvious that a_j^0 is always zero.

Table 2. Key schedule of GOST2

Round	Key	Round	Key	Round	Key	Round	Key
0	K^0	8	K^3	16	K^5	24	K^6
1	K^1	9	K^4	17	K^6	25	K^5
2	K^2	10	K^5	18	K^7	26	K^4
3	K^3	11	K^6	19	K^0	27	K^3
4	K^4	12	K^7	20	K^1	28	K^2
5	K^5	13	K^0	21	K^2	29	K^1
6	K^6	14	K^1	22	K^3	30	K^0
7	K^7	15	K^2	23	K^4	31	K^7

3 An overview of prior fixed point attacks on GOST2

In [28], a fixed point attack is proposed by Ashur *et al.* on GOST2 block cipher. This attack uses the following two main observations:

Observation 1. GOST2 rounds 10 to 15 are identical to the rounds 16 to 21 which allows an adversary to use fixed point property for the input internal states of rounds 10,16 and 22 (which are shown by S_{10}, S_{16}, and S_{22}).

Observation 2. Having input and output of a 3-round Feistel structure, an adversary can check if they match in the middle round to filter out the wrong keys.

The main ideas of this attack are precomputing some first and last part of the GOST2 in a hash table (first phase of the attack), and then continue by filtering some key bits between two respective fixed point, and finally, filtering the other key bits in the rest of the GOST2 encryption algorithm by using the precomputed table (second phase of the attack).

The total time complexity of the fixed point attack in [28] is 2^{237} as there are 2^{237} candidate keys left for exhaustive search in line 17 of Ashur's algorithm (see Appendix, Algorithm 1). However, the memory access of the attack hasn't been computed in [28]. Anyway, our investigations show that the memory access of the fixed point attack in [28] is not a dominant part for time complexity and it can be correctly ignored. The memory access (MA) method which is used in this paper is based on B-Tree implementation [30]. According to this implementation, for a table T with the number of n rows, the cost of access to a specific row is equal to $log_2(n)$.

Another fixed point attack on GOST2 is also proposed in [29] with similar observations. The goal was the reduction of the time complexity by some memory complexity penalties. However, it has not also taken the memory access of the attack into account. More precisely, there are two filtering parts in the attack (lines 32 and 34 of Algorithm 1 in [29], respectively) in which memory access cannot be ignored. According to our computations, the total memory access of these filterings is more than $2^{249}MAs$ ($2^{224} \times 2^8 \times 2^0 log_2(2^8) = 2^{236}MAs$ for the first filter and $2^{208} \times 2^{16} \times 2^8 \times 2^{13} log_2(2^{16}) = 2^{249}MAs$ for the second filter, assuming that the huge table T is implemented by B-Tree method and pre-sorted by S_3, S_{28}, $K^7[0-31]$, $K^2[0-7]$, $K^0[0-7]$, and $K^1[0-15]$). Hence, the total time complexity of the attack is at least 2^{233} encryptions and $2^{249}MAs$. The effect of this huge memory access on time complexity is reflected in the footnote of Table 1. In addition, the memory complexity of the attack is too much high (2^{196} bytes).

4 New fixed point attacks on GOST2

Here, we not only utilize the first observation in [28] but also add a new observation as follows: **Observation 3.** There are three 3-round portions with the exact same reduced-round encryption algorithm and subkeys. These portions are rounds 15 to 13, 2 to 0 and 28 to 30, all with three ordered subkeys of (K^2, K^1, K^0).

Suppose that any partial encryption/decryption starting from an arbitrary intermediate state of round i to another intermediate state of round j is called a parallel path. Therefore, a portion may be a union of distinct parallel paths. By this definition, according to the Observation 3 along with the others mentioned in [28], two fixed point attacks on GOST2 with similar procedures are expressed by the following main steps:

- Step 1: a simple precomputation (to make a primary filter);
- Step 2: building parallel filters;
- Step 3: making parallel paths by using parallel filters;
- Step 4: integration of parallel filters and paths with nested loops and filters by proper use of the precomputation phase, carries and other separator bits.

These attacks are slightly different in the precomputation and integration phases. Anyway and roughly, the master key in both attacks is recovered by the following procedure: during the first steps, a precomputation phase runs to build the primary filter, and 5 out of 8 subkeys are guessed (namely K^3 to K^7). To recover K^0, K^1, and K^2, the attacker uses 3 portions of the cipher where these keys are used: the first 3 rounds, rounds 13 to 15 (between two fixed points) and rounds 28 to 30. The main idea is to build tables (parallel filters and paths) corresponding to each nibble/S-box at a time, so that the time complexity is limited (only a few bits are guessed at a time) and so is the memory complexity (each table contains few elements). Since the key is incorporated by a modular addition, the nibbles entering the S-boxes are not completely independent and some bits corresponding to carries also have to be guessed. Once these tables are built for rounds 13 to 15 (building parallel filters), they are used with respect to rounds 0 to 2 and 28 to 30 to filter out the wrong keys by some nested loops and using the primary filter (the common bits are also checked) (making parallel paths by using parallel filters, and the integration phase) and finally the possible keys are obtained. These illustrations for GOST2 are summarized in Figure 2.

It should be noted that for both attacks, we guess the subkey K^7 for all of its 2^{32} values at the beginning. Also, we consider an encryption and a byte as the units of time and memory complexities, respectively. Hence, the computation of each S-box can be considered as a 2^{-8} encryption since each encryption of GOST2 has 256 S-boxes. In addition, as each S-box of GOST2 has a dimension of 4, each look-up for an S-box could be considered as $log_2(2^4) = 2^2 MA$ in look-up table implementation according to B-Tree method. So, each MA can also be considered as a 2^{-10} encryption.

The details of our attacks are explained in the fol-

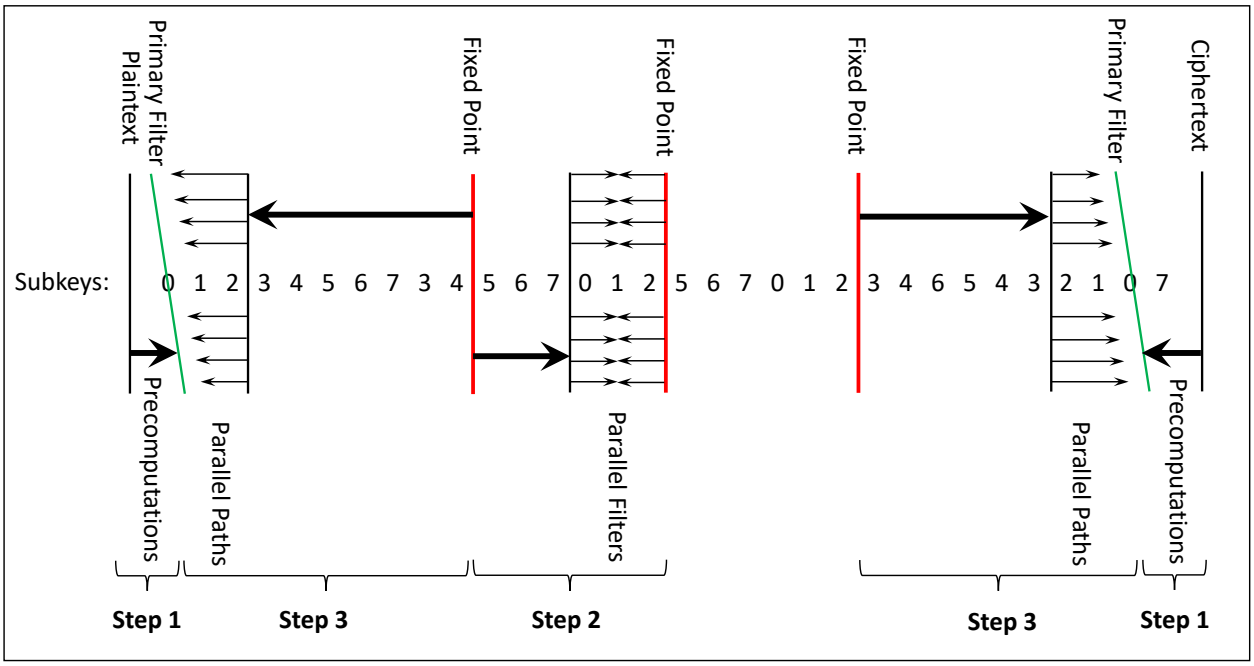

Figure 2. New fixed point attacks procedure on GOST2 ($\forall i \in \{0,1,...,7\}$, K^i is shown by i for simplicity. In addition, each round function of GOST2 is shown by only its subkey); step 1 shows the precomputation phase, step 2 shows building parallel filters, and step 3 shows making parallel paths by using parallel filters; the integration phase is going to be done after these steps at the green skewed lines.

lowing subsections and the exact algorithms are presented in Algorithm 2 and 3 of the Appendix.

4.1 Attack 1 details

Simple Precomputation Phase. At the first step, some precomputations must be performed from plaintext (resp. ciphertext) to certain intermediate states of early (resp. last) rounds. The chosen intermediate states for Attack 1 are $(P^R = S_0^R, S_1^R[11-26])$ for early rounds and $(S'^R_{29}[11-16], S'^R_{30})$ for last rounds. Therefore, assuming a known K^7, we should also guess 16 bits of $K^0[0-15]$ in addition to all pairs of (P, C) to obtain the chosen intermediate states. These intermediate states are a part of parallel paths ending points in the later steps (see step 1 and skewed lines in Figure 2). Hence, the intermediate states of $S^{R_1}[11-26]$, $S'^{R_{29}}[11-26]$ and $S'^{R_{30}}$ should be computed and saved together with P^R and $K^0[0-15]$ in a hash table of $\mathcal{U}$, namely primary filter, with a proper argumentation (the details are reflected in Algorithm 2, lines 2 to 6). The total number of arguments bits is chosen in a way that it will be equal to total guessed values for the primary filter. More precisely, the total number of arguments bits in $\mathcal{U}$ is equal to 80, and also there are exactly 2^{64} plaintext-ciphertext pairs and 2^{16} subkey bits of $K^1[0-15]$ that must be guessed. Therefore, it can be seen that each argument of $\mathcal{U}$ has one row on average. Indeed, the argumentation is utilized to reduce the memory complexity by choosing a pre-defined index for each row. As there are 16 S-boxes computations and a 32-bit saved value (see line 5 in Algorithm 2) for each guess in this step, the time and memory complexities to construct the table for a fixed K^7 are $\frac{16 \times 2^{80}}{256} = 2^{76}$ encryptions and $\frac{2^{80} \times 32}{8} = 2^{82}$ bytes, respectively.

Building Parallel Filters. After obtaining the primary filter, the next step is building parallel filters. Therefore, we should continue by guessing the fixed point value of $S_{10} = S_{16} = S_{22}$. After choosing the fixed point value, there are two portions that should be used for the attack: one is between two successive fixed points (see step 2 in Figure 2), and the other is between the first/last fixed points and $\{(S_0^L[27-10], S_0^R, S_1^R[11-26]), (S'^R_{29}[11-26], S'^L_{30}[27-10], S'^R_{30})\}$, the ending points of parallel paths (the green skewed lines in Figure 2). In this step, we consider the rounds between two successive fixed points for building parallel filters and then in the next step, we continue by other rounds for making parallel paths. As one can see in Table 2, the K^5 and K^6 subkeys are in both parts and so they should be guessed along with the fixed point. Hence, we should guess 128 bits of $S_{10} = S_{16} = S_{22}, K^5, K^6$. It should be mentioned that there are 160 bits with determined values until now, because of considering

a fixed value for K^7. Therefore, the time complexity of any computations from now on has an increasing factor of 2^{160}.

Here, we are looking to build parallel filters between any two successive fixed points, for example, S_{10} and S_{16}. In order to do that, first, we can easily compute 3-round states according to the known values of K^5, K^6, K^7, and S_{10}. Hence, we calculate S_{13} with time complexity of only 3-round encryption. Then, for rounds between S_{13} and S_{16}, as the subkeys are K^0, K^1 and K^2, we can build a 3-round parallel filter for i^{th} $(0 \le i \le 7)$ nibble of them by guessing three carries $(a^i_{13,15}, a^{i+3}_{14})$ and a related separator bit $(MV_{14}[15]^{+4i})$ (see Figure 3) in a new hash table $\mathcal{H}_i$ with proper indexes. The details are again reflected in Algorithm 2, lines 8 to 12. These hash tables are the results of building parallel filters step.

It can be seen that all of the hash tables $\mathcal{H}_i$ with $0 \le i \le 7$ are independent of each other by considering fixed values of carries and other separator bits. Due to the 8 bits guessed values of $(MV_{14}[11-15]^{+4i}, a^i_{13,15}, a^{i+3}_{14})$ for computing each $\mathcal{H}_i$, each hash table $\mathcal{H}_i$ has exactly 2^8 rows. Also, there are only 3 S-boxes computations (see Figure 3) and a 16-bit saved value (see lines 10 and 11 in Algorithm 2) for each guess in this step. Therefore, the time and memory complexities of building all eight hash tables $\mathcal{H}_i$ are $\frac{8 \times 2^8 \times 3}{256} \approx 2^{4.6}$ encryptions and $\frac{8 \times 2^8 \times 16}{8} = 2^{12}$ bytes, respectively.

Making Parallel Paths by using Parallel Filters. Now we go through the second part, from fixed points to ending points of parallel paths. In this step, the final goal is to extract a combination result for each parallel filter and the corresponding path in a new hash table $\mathcal{Q}$. In order to do that, first, we should calculate the starting points of parallel paths, which are S'_2 and S_{28}. For this purpose, the subkeys K^3 and K^4 are also required. So, they should be guessed. Therefore, any computations from now has an increasing factor of $2^{160} \times 2^{64} = 2^{224}$. Also, it is clear that the time complexity of calculating both S'_2 and S_{28} is 13-round encryption for each guess.

With having the starting points (it means S'_2, S_{28}), making each parallel path is possible. The hash tables $\mathcal{H}_i, 0 \le i \le 7$ should also be used to make them. Each parallel path made by its corresponding parallel filter is shown by a hash table $\mathcal{Q}_i$ which can be computed by all 2^8 rows of $\mathcal{H}_i$ and also 2^6 values of $a^i_{2,28}, a^{i+3}_{1,29}, MV_{1,29}[15]^{+4i}$ when i changes from 0 to 7 (see Figure 4). Hence, each hash table $\mathcal{Q}_i$ has 2^{14} rows. Again, the details are shown in Algorithm 2, lines 15 to 18. According to line 17 of Algorithm 2, the argumentation used for i^{th} parallel path consists

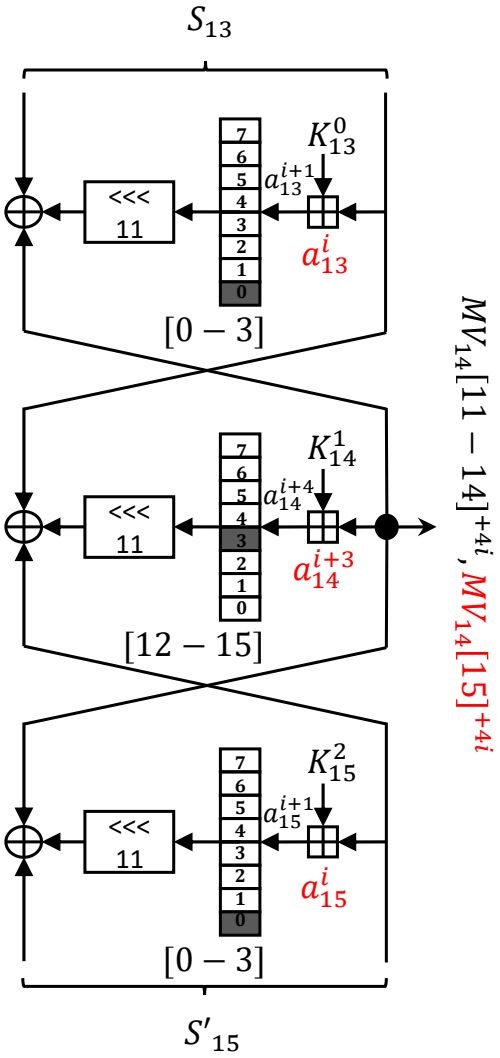

Figure 3. Building parallel filters by guessing three carries and another separator bit.

of 10 bits as $\mathcal{Q}_i[a^i_{2,13,15,28}, a^{i+3}_{1,14,29}, MV_{1,14,29}[11]^{+4i}]$. The reason of choosing this argumentation is returned to the variables stored in that hash table, which are $\{MV_{0,30}[23-26]^{+4i}, MV_{1,29}[12-15]^{+4i}, MV_{14}[15]^{+4i}, K^{0,2}[0-3]^{+4i}, K^1[12-15]^{+4i}, a^{i+1}_{2,13,15,28}, a^{i+4}_{1,14,29}\}$. It means that the outputs obtained from $\mathcal{Q}_i$ can be directly used as the inputs for $\mathcal{Q}_{i+1}$. This property will be used in the integration phase of the attack.

As there are only 4 S-boxes computations (see Figure 4) and a 36-bit saved value (see line 17 in Algorithm 2) for each guess in this step, the total time and memory complexities of making all eight hash tables $\mathcal{Q}_i$ are $\frac{8 \times 2^{14} \times 4}{256} = 2^{11}$ encryptions and $\frac{8 \times 2^{14} \times 36}{8} \approx 2^{19.2}$ bytes, respectively. In the next step, we have to integrate these hash tables $(\mathcal{Q}_i, 0 \le i \le 7)$ with nested loops and filters so that

we can find a candidate key.

Integration Phase. In order to integrate all of the hash tables $\mathcal{Q}_i, 0 \leq i \leq 7$, we need to consider all rows in each $\mathcal{Q}_i$ along with dependencies to the next hash table $\mathcal{Q}_{i+1}$. This results in some nested loops, which can be seen in the Algorithm 2, lines 19 to 36. Now, the exact integration phase is going to be illustrated.

The integration phase is begun with $\mathcal{Q}_0$. For this particular hash table, four bits of arguments are equal to zero which are $a^0_{2,13,15,28} = (0,0,0,0)$. As it is mentioned before, each hash table $\mathcal{Q}_i$ has 2^{14} rows. Hence, considering $a^0_{2,13,15,28} = (0,0,0,0)$, 2^{10} rows have been left for $\mathcal{Q}_0$. Now, according to the previous step, having any row of $\mathcal{Q}_i$, all input arguments of $\mathcal{Q}_{i+1}$ are also obtained. As $\mathcal{Q}_{i+1}$ has 10 argument bits, it can be concluded that for each row of $\mathcal{Q}_i$, there are only $2^{14-10} = 2^4$ chances for rows in $\mathcal{Q}_{i+1}$. This can be continued until $i = 3$. Hence, for the first loop, we need to consider 2^{10} rows of table $\mathcal{Q}_0$, while for the next three loops, there are only 2^4 rows for each table. After passing through four first loops (or equivalently integration of all $\mathcal{Q}_i, 0 \leq i \leq 3$), the primary filter $\mathcal{U}$ can be used to extract full intermediate states for ending points of parallel paths. Therefore, using the other hash tables $\mathcal{Q}_i, 4 \leq i \leq 7$ results in filtering the wrong keys until the final table $\mathcal{Q}_7$ will be considered. By performing an exact investigation on $\mathcal{Q}_4$ after applying the primary filter, it can be found that a 2^{-4} filtration is obtained by considering $\mathcal{Q}_4$. In addition, hash tables $\mathcal{Q}_5$ and $\mathcal{Q}_6$ have similar effects. But the hash table $\mathcal{Q}_7$ should be consistent with both $\mathcal{Q}_6$ and $\mathcal{Q}_0$. This leads to a 2^{-10} filtration of wrong guesses when the hash table $\mathcal{Q}_7$ is checked.

The time complexity of the integration phase has both types of MA and computations. Because here we only lookup tables and extract the proper values and finally check the candidate key. According to lines between 19 to 36 and the above mentioned, for the first four loops, we need to consider $2^{10} \times 2^4 \times 2^4 \times 2^4 = 2^{22}$ iterations. Afterward, the intermediate variables of endpoints of parallel paths are extracted from primary filter $\mathcal{U}$. It is worthy to be noted that by passing through each loop, 20 bits of indexes of table $\mathcal{U}$ are obtained. So, the memory access for line 23 of Algorithm 2 can be reduced to access a table with 2^{20} rows. After attaining the values of endpoints of the other parallel paths ($P^R[7-22]$ and $S'^R_{30}[7-22]$), looking ups on hash tables $\mathcal{Q}_4$ to $\mathcal{Q}_7$ result in filtering the wrong values. So, passing through each filter for the first three one results in 2^{-4} reduction of iterations and the final filter also reduce the iterations 2^{-10} times. Therefore, the total reduction of iterations is equal to $2^{-4} \times 2^{-4} \times 2^{-4} \times 2^{-10} = 2^{-22}$. Finally, it is clear that only one ($2^{22} \times 2^{-22} = 1$) candidate key $K = K^{0-7}$ left for trying to find the master key.

Table 3. Dominant memory access calculations for integration phase of the Attack 1

Line	Memory access
After line 21	$2^{18}(log_2(2^{20}))$ ⋆ Third partial memory access of primary filter $\mathcal{U}$
Line 22	$2^{18}(log_2(2^{10}) + 2^4)$ Memory access of $\mathcal{Q}_3$ and choosing 2^4 rows from that
Line 23	$2^{22}(log_2(2^{20}))$ Fourth (last) partial memory access of primary filter $\mathcal{U}$
Line 24	$2^{22}(log_2(2^{14}))$ Memory access of of $\mathcal{Q}_4$
Line 25	$2^{18}(log_2(2^{14}))$ Memory access of of $\mathcal{Q}_5$
Total MA	$MA \approx 2^{18}(log_2(2^{20}) + log_2(2^{10}) + 2^4)$ $+2^{22}(log_2(2^{20}) + log_2(2^{14})) + 2^{18}log_2(2^{14})$ $= 2^{18} \times 60 + 2^{22} \times 34 \approx 2^{27.2}$

⋆ First, second and third partial memory access of primary filter $\mathcal{U}$ are not shown in the Algorithm 2

According to these explanations, the time complexity for lines between 19 to 36 is only 1 encryption, while the dominant memory access of these lines is calculated in Table 3.

The memory complexity of this step is negligible and can be ignored. At the end of this step, a candidate key is checked and after performing all procedures, the master key should be extracted.

Total Attack Complexities. Up to now, all the steps of the Attack 1 are introduced. Here, we focus on computing the total time and memory complexities of the attack. For this, we should sum all the dominated complexities achieved by all steps. Hence, the total normalized complexities are:

$$\begin{aligned} C_{time} &\simeq 2^{224} \times 2^{11} = 2^{235} \text{ encryptions} \\ C_{MA} &\simeq 2^{224} \times 2^{27.2} = 2^{251.2} MA \\ C_{memory} &\simeq 2^{82} \text{ bytes} \end{aligned} \quad (2)$$

For better comparison of our result and the best previous one in [28], one can change memory access to encryption complexity. With this modification, Attack 1 complexities are $2^{235} + 2^{251.2} \times 2^{-10} \simeq 2^{241.2}$ encryptions and 2^{82} bytes, while the Ashur's attack complexities are $2^{237.5} + 2^{244.1} \times 2^{-10} \simeq 2^{237.6}$ encryptions and $2^{138.15}$ bytes. It means Attack 1 results

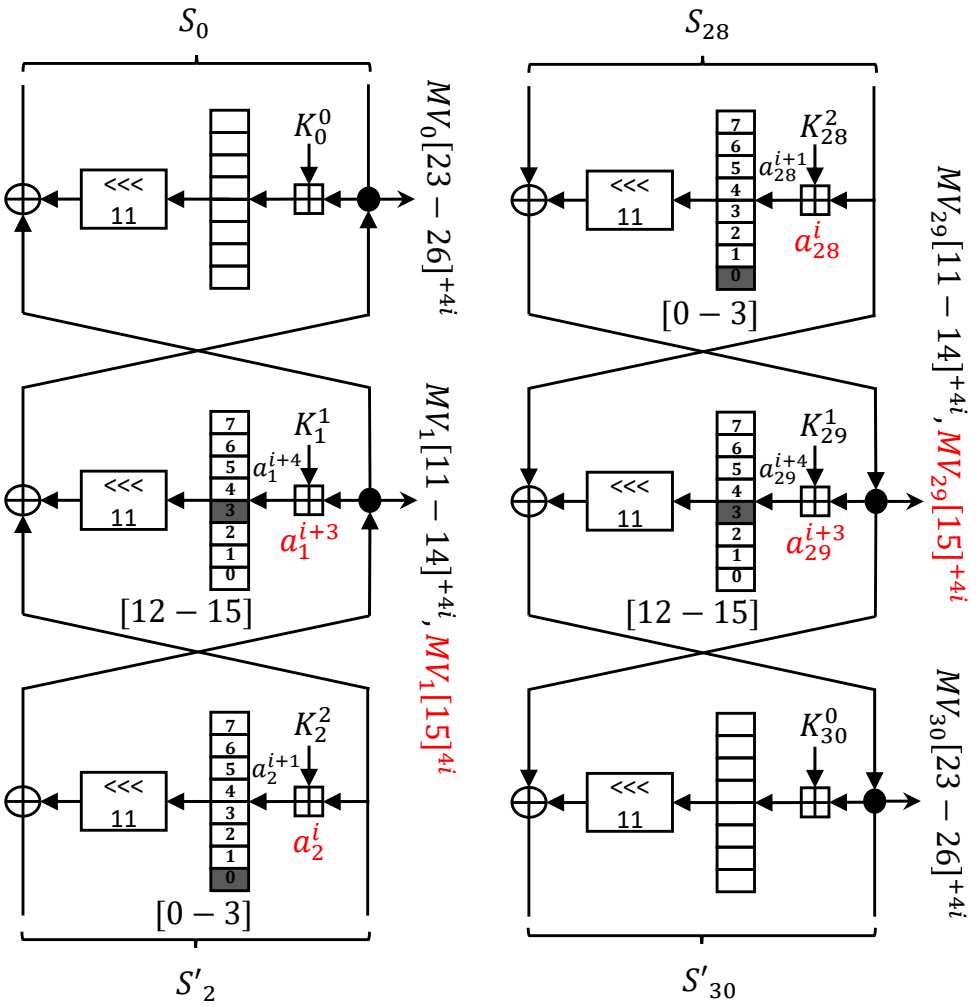

Figure 4. Making parallel paths

in increasing 4.2-bit in time complexity while reducing 56-bit in memory complexity. It should be noted that the data complexity of both attacks is 2^{64}.

4.2 Attack 2 details

As the general procedure of Attack 2 is similar to Attack 1, we just briefly explain the attack procedure and have more concentrate on the differences here.

Simple Precomputation Phase. Here, we only precompute internal states of S_0 and S'_{30} by known K^7 bits and all pairs of (P, C). Therefore, the precomputations are much simpler than Attack 1, while in order to show how to use these precomputations (or the primary filter), we used different argumentation for hash table $\mathcal{U}$ (see Algorithm 3, lines 2 to 5). The arguments of $\mathcal{U}$ are chosen in a way that could be simply utilized in the integration phase of the attack. According to this precomputation phase, the memory complexity is equal to $\frac{2^{64} \times 64}{8} = 2^{67}$ bytes, while the time complexity is negligible in comparison to the other steps of the attack.

Building Parallel Filters and Parallel Paths. Steps 2 and 3 are exactly the same as Attack 1, with equal complexities. The most important output of these steps are $\mathcal{Q}_i, 0 \leq i \leq 7$ hash tables.

Integration Phase. The general procedure of this phase is also the same as Attack 2. However, according to the different primary filter obtained here, the nested loops should be considered in a manner that the primary filter could be utilized as soon as possible. As the primary filter only contains S_0 and S'_{30} bits, it translates to compute more bits of S_0 and S'_{30} by using known bits of K^0 when it is possible. Again, the procedure is started with hash table $\mathcal{Q}_0$ with $a^0_{2,13,15,28} = (0, 0, 0, 0)$ (2^{10} rows), and continued with $\mathcal{Q}_1$ and $\mathcal{Q}_2$ (each with 2^4 rows) with the same manner as Attack 1. However, in the third loop, by guessing two more bits of $MV_{0,30}[3]$, all the variables needed for computing more 8 bits of S_0, S'_{30} are also

known. Hence, they are considered to compute these 8 bits according to line 21 of Algorithm 3. So, the third loop contains 2^6 iterations for the total procedure of the attack instead of 2^4. But it should be noted that the two guessed bits $MV_{0,30}[3]$ are also in the outputs of $\mathcal{Q}_3$. Therefore, the number of iterations for the remaining loops will not change (and they are equal to 2^4). From now on, in each loop, computing more 8 bits of (S_0, S'_{30}) by guessing two more bits and going throughout two S-boxes is possible. So, the integration phase continues in the same way for hash tables $\mathcal{Q}_3$ and $\mathcal{Q}_4$. After the fifth loop, all the input indexes of primary filter $\mathcal{U}$ are attained and so, all the bits of (S_0, S'_{30}) can be achieved by the primary filter. It is worthy to be noted that by passing through each loop after the third loop, 16 bits of indexes of hash table $\mathcal{U}$ are obtained. So, the memory access for using the primary filter (line 26 of Algorithm 2) can be reduced to access a hash table with 2^{16} rows. Now, it is the time to use the other hash tables of $\mathcal{Q}_i, 5 \leq i \leq 7$ to filter out the wrong values in a straightforward manner. All the procedures are reflected in Algorithm 3, from lines 18 to 40.

According to these lines and the above-mentioned illustrations, there are five nested loops followed by four nested filters. These five loops have $2^{10}, 2^4, 2^6, 2^4$, and 2^4 iterations, respectively, while the four nested filters have $2^{-2}, 2^{-4}, 2^{-4}$, and 2^{-10} filtrations, respectively. Hence, the normalized time complexity of the integration phase by considering dominant parts can be obtained according to Table 4.

Again, the memory complexity of this step is negligible and can be ignored. At the end of this step, a candidate key is checked and after performing all procedures, the master key should be extracted.

Total Attack Complexities. For computing the complexities of the Attack 2, we should sum all the dominated complexities achieved by all four steps. Hence the total normalized complexities are:

$$\begin{aligned} C_{time} &\simeq 2^{224} \times 2^{21.1} = 2^{245.1} \text{ encryptions} \\ C_{MA} &\simeq 2^{224} \times 2^{32.4} = 2^{256.4} MA \\ C_{memory} &\simeq 2^{67} \text{ bytes} \end{aligned} \quad (3)$$

Again, one can change memory access to encryption complexity. With this modification, Attack 2 complexities are $2^{245.1} + 2^{256.4} \times 2^{-10} \simeq 2^{246.9}$ encryptions and 2^{67} bytes. It means that Attack 2 results in more increasing of 5.7-bit in time complexity while reducing more 15-bit in memory complexity in comparison to Attack 1. It should be noted that the data complexity of this attack is also 2^{64}.

Table 4. Dominant time complexities for integration phase of the Attack 2

Line	Memory access	Computations
Line 21	-	$2^{20} \times 2$
	Can be ignored	S-boxes
After line 21	$2^{20} \times (log_2(2^{16}))$	-
	$\star$ Third partial MA	-
Line 22	$2^{20}(log_2(2^{12}) + 2^2)$	-
	MA of $\mathcal{Q}_3$ and choosing 2^2 rows from that	-
Line 23	-	$2^{24} \times 2$
	Can be ignored	S-boxes
After line 23	$2^{24} \times (log_2(2^{16}))$	-
	$\star$ Fourth partial MA	-
Line 24	$2^{24}(log_2(2^{12}) + 2^2)$	-
	MA of $\mathcal{Q}_4$ and choosing 2^2 rows from that	-
Line 25	-	$2^{28} \times 2$
	Can be ignored	S-boxes
Line 26	$2^{28} \times (log_2(2^{16}))$	-
	Fifth (last) partial MA	-
Line 27	-	-
	Doesn't need any MA	-
Line 28	$2^{26} \times (log_2(2^{14}))$	-
	MA of $\mathcal{Q}_5$	-
Line 29	$2^{22} \times (log_2(2^{14}))$	-
	MA of $\mathcal{Q}_6$	-
Line 30	$2^{18} \times (log_2(2^{14}))$	-
	MA of $\mathcal{Q}_7$	-
Line 31	-	$2^8 \times 256$
	-	S-boxes
Total	$MA \approx 2^{20}(log_2(2^{16}) + log_2(2^{12}) + 2^2)$	$\frac{2^{21}}{256} + \frac{2^{25}}{256}$
	$+2^{24}(log_2(2^{16}) + log_2(2^{12}) + 2^2)$	
	$+2^{28}log_2(2^{16}) + 2^{26}log_2(2^{14})$	$\frac{2^{29}}{256} + \frac{2^{16}}{256}$
	$+2^{22}log_2(2^{14}) + 2^{18}log_2(2^{14})$	
	$\approx 2^{25}+2^{29}+2^{32}+2^{29.8}+2^{25.8}+2^{21.8} \approx 2^{32.4}$	$\approx 2^{21.1}$ enc.

$\star$ First to fourth partial memory access of primary filter $\mathcal{U}$ are not shown in the Algorithm 3

5 Conclusion

Efficient division of 3-round GOST2 encryption into 8 smaller parallel filters and paths along with three identical 3-round encryptions in rounds 2 to 0, 15 to 13, and 28 to 39 made an attacker stronger to apply a fixed point attack on GOST2 with lower memory complexity. For this purpose, the attacker should first compute a primary filter as a precomputation phase. Then she should check inside of the two successive fixed points to extract parallel filters. Next, the rest of the cipher should be considered for obtaining parallel paths. Finally, with the integration of all previous steps, the attacker could filter out the wrong keys. In this paper, two new fixed point attacks are provided with this approach. The results show a significant reduction in memory complexity while preserving the time complexity near the previous results. Though the new attacks do not threaten GOST2 block cipher, they opened a new area for enhancing memory complexity of the fixed point attack.

Acknowledgement

This work was partially supported by the Iranian National Science Foundation (INSF) under contract no. 96.53979 and INSF cryptography chair and by the Office of Vice-President for the Science and Technology, I. R. Iran.

References

[1] Matsui, M.: Linear Cryptoanalysis Method for DES Cipher. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386397. Springer (1994)

[2] Bogdanov, A., Rijmen, V.: Zero-correlation linear cryptanalysis of block ciphers. Accepted to Designs, Codes and Cryptography, 2012

[3] E. Biham, A. Shamir, Differential Cryptanalysis of DESlike Cryptosystems, Journal of Cryptology, Vols.4, no.1, pp. 3-72 (1991)

[4] E. Biham, A. Biryukov, A. Shamir, Cryptanalysis of Skipjack Reduced to 31 Rounds using Impossible Differentials," in Advances in Cryptology: EUROCRYPT'99 LNCS 1592, pp. 12-23, Springer Verlag (1999)

[5] W. Diffie and M. Hellman. Exhaustive cryptanalysis of the NBS data encryption standard. Computer, 10(6):7484, (1977)

[6] Canteaut, A., Naya-Plasencia, M., and Vayssire, B. Sieve-in-the-middle: Improved MITM attacks. In CRYPTO (2013), R. Canetti and J. Garay, Eds., vol. 8042 of Lecture Notes in Computer Science, Springer, pp. 222240.

[7] Bogdanov, A., Khovratovich, D., Rechberger, C.: Biclique Cryptanalysis of the Full AES. ASIACRYPT 2011, LNCS, vol. 7073, pp. 344-371. Springer, Heidelberg (2011)

[8] Ahmadi, Siavash, *et al.* Low-data complexity biclique cryptanalysis of block ciphers with application to piccolo and hight. IEEE Transactions on Information Forensics and Security 9.10 (2014): 1641-1652 (2014)

[9] Dunkelman, O., Keller, N., and Shamir, A. Improved single-key attacks on 8-round AES-192 and AES-256. In ASIACRYPT (2010), M. Abe, Ed., vol. 6477 of Lecture Notes in Computer Science, Springer, pp. 158176.

[10] Derbez, P., Fouque, P.-A., and Jean, J. Improved key recovery attacks on reduced-round AES in the single-key setting. In EUROCRYPT (2013), T. Johansson and P. Nguyen, Eds., vol. 7881 of Lecture Notes in Computer Science, Springer, pp. 371387.

[11] Russian National Bureau of Standards. Federal Information Processing Standard-Cryptographic Protection - Cryptographic Algorithm. GOST 28147-89, 1989

[12] Dolmatov, Vasily. "GOST R 34.12-2015: Block Cipher Kuznyechik. Transformation 50 (2016): 10.

[13] Youngdai Ko, Seokhie Hong, Wonil Lee, Sangjin Lee, and Ju-Sung Kang. Related Key Differential Attacks on 27 Rounds of XTEA and Full-Round GOST. In Bimal K. Roy and Willi Meier, editors, Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004, Revised Papers, volume 3017 of Lecture Notes in Computer Science, pages 299316. Springer, 2004.

[14] Takanori Isobe: A Single-Key Attack on the Full GOST Block Cipher, In FSE 2011, pp. 290-305, Springer LNCS 6733, 2011

[15] Nicolas Courtois: On Multiple Symmetric Fixed Points in GOST, In Cryptologia, Volume 39, Issue 4, 2015, pp. 322-334.

[16] Haruki Seki and Toshinobu Kaneko: Differential Cryptanalysis of Reduced Rounds of GOST. In SAC 2000, LNCS 2012, pp. 315-323, Springer, 2000

[17] Nicolas Courtois, Micha Misztal: First Differential Attack On Full 32-Round GOST, in ICICS'11, pp. 216-227, Springer LNCS 7043, 2011.

[18] Nicolas Courtois, Micha Misztal: Aggregated Differentials and Cryptanalysis of PP-1 and GOST, In CECC 2011, 11th Central European Conference on Cryptology. In Periodica Mathematica Hungarica Vol. 65 (2), 2012, pp. 1126, Springer.

[19] Nicolas T. Courtois, Theodosis Mourouzis, Micha Misztal, Jean-Jacques Quisquater, Guangyan Song: Can GOST Be Made Secure Against Differential Cryptanalysis? In Cryptolo-

gia, vol. 39, Iss. 2, 2015, pp. 145-156.
[20] Nicolas Courtois, Theodosis Mourouzis, Advanced Truncated Differential Attacks Against GOST Block Cipher and Its Variants, In Computation, Cryptography, and Network Security, Springer, pp. 351-380, 2015.
[21] Nicolas Courtois: Security Evaluation of GOST 28147-89 In View Of International Standardisation, in Cryptologia, Volume 36, Issue 1, pp. 2-13, 2012.
[22] Nicolas T. Courtois: Cryptanalysis of GOST in the Multiple Key Scenario, In postproceedings of CECC 2013, Tatra Mountains Mathematical Publications. Vol. 57, no. 4 (2013), p. 45-63.
[23] Nicolas Courtois: On Multiple Symmetric Fixed Points in GOST, In Cryptologia, Volume 39, Issue 4, 2015, pp. 322-334
[24] Takanori Isobe: A Single-Key Attack on the Full GOST Block Cipher, In FSE 2011, pp. 290-305, Springer LNCS 6733, 2011
[25] Itai Dinur, Orr Dunkelman and Adi Shamir: Improved Attacks on Full GOST, FSE 2012, LNCS 7549, pp. 9-28, 2012
[26] Nicolas Courtois: Algebraic Complexity Reduction and Cryptanalysis of GOST, Monograph study on GOST cipher, 2010-2014, 224 pages
[27] Alekseychuk, A. N., and L. V. Kovalchuk. Towards a Theory of Security Evaluation for GOST-like Ciphers against Differential and Linear Cryptanalysis. IACR Cryptology ePrint Archive 2011 (2011): 489.
[28] Ashur, Tomer, Achiya Bar-On, and Orr Dunkelman. Cryptanalysis of GOST2. IACR Transactions on Symmetric Cryptology 2017.1 (2017): 203-214.
[29] Ahmadi, Siavash, and Mohammad Reza Aref. Improved Fixed Point Attack on Gost2. 2017 14th International ISC (Iranian Society of Cryptology) Conference on Information Security and Cryptology (ISCISC). IEEE, 2017.
[30] Comer, Douglas. Ubiquitous B-tree. ACM Computing Surveys (CSUR) 11.2 (1979): 121-137.

Siavash Ahmadi received the B.S. and M.S. degrees in electrical engineering in 2012 and 2014, respectively, from Sharif University of Technology, Tehran, Iran. He is currently a Ph.D. candidate in electrical engineering (communication systems and security) at Sharif University of Technology. His special fields of interest include cryptology and wireless security, with emphasis on cryptanalysis.

Mohammad Reza Aref received the B.S. degree in 1975 from the University of Tehran, Iran, and the M.Sc. and Ph.D. degrees in 1976 and 1980, respectively, from Stanford University, Stanford, CA, USA, all in electrical engineering. He returned to Iran in 1980 and was actively engaged in academic affairs. He was a faculty member of Isfahan University of Technology from 1982 to 1995. He has been a professor of electrical engineering at Sharif University of Technology, Tehran, since 1995, and has published more than 290 technical papers in communication and information theory and cryptography in international journals and conferences proceedings. His current research interests include areas of communication theory, information theory, and cryptography.

Appendix

The fixed point attack algorithm proposed by Ashur *et al.* is provided here. Also, the exact algorithms of our new fixed point attacks on GOST2 are presented.

Algorithm 1 The previous fixed point attack

for all $K^7, K^0[0-11], K^2[0-11], K^1[11]$ **do**
  for all $(P,C), K^0[12-31], K^1[0-10,12-31], K^2[12-31]$ **do**
    $S_{28} \leftarrow$ partial decryption from C with (K^2, K^1, K^0, K^7)
    $S_3 \leftarrow$ partial encryption from P with (K^0, K^1, K^2)
    Save $(K^0[12-31], K^1[0-10,12-31], K^2[12-31])$ in a hash table $T[S_3||S_{28}]$
  end for
  for $S_{10} = S_{16} = S_{22}, K^5, K^6$ **do**
    $S_{13} \leftarrow$ partial encryption from S_{10} with (K^5, K^6, K^7)
    if FILTER$(S_{16}, S_{13}, K^0[0-11], K^2[0-11])$ ==TRUE **then**
      for K^3, K^4 **do**
        $(K^0[0-11], K^1[12-19], K^2[0-11]) \leftarrow$ SOLVE$(S_{16}, S_{13}, K^0[0-11], K^2[0-11], K^1[11])$
        $S_3 \leftarrow$ partial decryption from S_{10} with K^{3-7}
        $S_{28} \leftarrow$ partial encryption from S_{22} with K^{3-6}
        for each $(K^0[12-31], K^1[0-10,12-31], K^2[12-31])$ in $T[S_3||S_{28}]$ **do**
          $FILTER(K^1[12-19])$
        end for
        TRY $K = K^0||K^1||...||K^7$
      end for
    end if
  end for
end for

Algorithm 2 Details of Attack 1 on Gost2

for all K^7 **do** ▷ Precomputation phase:
  for all $(P, C), K^0[0-15]$ **do**
    $S'^R_{30}, S'^R_{29}[11-26] \leftarrow$ partial decryption with $C, K^7, K^0[0-15]$
    $S^R_1[11-26] \leftarrow$ partial encryption with $P, K^0[0-15]$
    Save $(P^R[7-22], S'^R_{30}[7-22])$ in a new hash table $\mathcal{U}[S^R_1[11-26], P^R[23-6], S'^R_{29}[11-26], S'^R_{30}[23-6], K^0[0-15]]$
  end for
  for all $S_{10} = S_{16} = S_{22}, K^{5,6}$ **do** ▷ Parallel filters:
    $S_{13} \leftarrow$ partial encryption from S_{10} with K^{5-7}
    for all $MV_{14}[11-15]^{+4i}, a^i_{13,15}, a^{i+3}_{14}$ with $i = 0$ to 7 **do**
      Calc. $\mathcal{X} = (K^{0,2}[0-3]^{+4i}, K^1[12-15]^{+4i}, a^{i+1}_{13,15}, a^{i+4}_{14})$ by $S_{13,16}$
      Save $\mathcal{X}, MV_{14}[15]^{+4i}$ in a new hash table $\mathcal{H}_i[a^i_{13,15}, a^{i+3}_{14}, MV_{14}[11]^{+4i}]$
    end for
    for all $K^{3,4}$ **do** ▷ Parallel paths:
      $(S'_2, S_{28}) \leftarrow$ partial decryption/encryption from $S_{10,22}$ with K^{3-7}
      for all $a^{i+3}_{1,29}, a^i_{2,28}, MV_{1,29}[15]^{+4i}$ & all rows in $\mathcal{H}_i$, with $i = 0$ to 7 **do**
        Calc. $MV_{1,29}[11-14]^{+4i}, MV_{0,30}[23-26]^{+4i}, a^{i+4}_{1,29}, a^{i+1}_{2,28}$ by S'_2, S_{28}
        Consider equivalent row in $\mathcal{H}_i$ and save $MV_{0,30}[23-26]^{+4i}, MV_{1,29}[12-15]^{+4i}, MV_{14}[15]^{4i}, K^{0,2}[0-3]^{+4i}, K^1[12-15]^{+4i}, a^{i+1}_{2,13,15,28}, a^{i+4}_{1,14,29}$ in a new hash table $\mathcal{Q}_i[a^i_{2,13,15,28}, a^{i+3}_{1,14,29}, MV_{1,14,29}[11]^{+4i}]$
      end for ▷ Integration phase:
      for all rows in $\mathcal{Q}_0$ with $a^0_{2,13,15,28} = (0,0,0,0)$ **do**
        for all rows in $\mathcal{Q}_1[a^1_{2,13,15,28}, a^4_{1,14,29}, MV_{1,14,29}[15]]$ **do**
          for all rows in $\mathcal{Q}_2[a^2_{2,13,15,28}, a^5_{1,14,29}, MV_{1,14,29}[19]]$ **do**
            for all rows in $\mathcal{Q}_3[a^3_{2,13,15,28}, a^6_{1,14,29}, MV_{1,14,29}[23]]$ **do**
              Get $(P^R[7-22], S'^R_{30}[7-22])$ from $\mathcal{U}[S^R_1[11-26], P^R[23-6], S'^R_{29}[11-26], S'^R_{30}[23-6], K^0[0-15]]$
              if there is a row in $\mathcal{Q}_4[a^4_{2,13,15,28}, a^7_{1,14,29}, MV_{1,14,29}[27]]$ with known $MV_{0,30}[7-10]$ **then**
                if there is a row in $\mathcal{Q}_5[a^5_{2,13,15,28}, a^0_{1,14,29}, MV_{1,14,29}[31]]$ with known $MV_{0,30}[11-14]$ **then**
                  if there is a row in $\mathcal{Q}_6[a^6_{2,13,15,28}, a^1_{1,14,29}, MV_{1,14,29}[3]]$ with known $MV_{0,30}[15-18]$ **then**
                    if there is a row in $\mathcal{Q}_7[a^7_{2,13,15,28}, a^2_{1,14,29}, MV_{1,14,29}[7]]$ with known $(MV_{0,30}[19-22], MV_{1,14,29}[11], a^3_{1,14,29})$ **then**
                      Try $K = K^0||K^1||...||K^7$
                    end if
                  end if
                end if
              end if
            end for
          end for
        end for
      end for
    end for
  end for
end for

Algorithm 3 Details of Attack 2 on Gost2

for all K^7 **do** ▷ Precomputation phase:
  for all (P, C) **do**
    $S'_{30} \leftarrow$ partial decryption with C, K^7
    Save $(P^L[23-10], P^R[11-22], S'^L_{30}[23-10], S'^R_{30}[11-22])$ in a new hash table $\mathcal{U}[P^L[11-22], P^R[23-10], S'^L_{30}[11-22], S'^R_{30}[23-10]]$
  end for
  for all $S_{10} = S_{16} = S_{22}, K^{5,6}$ **do** ▷ Parallel filters:
    $S_{13} \leftarrow$ partial encryption from S_{10} with K^{5-7}
    for all $MV_{14}[11-15]^{+4i}, a^i_{13,15}, a^{i+3}_{14}$ with $i = 0$ to 7 **do**
      Calc. $\mathcal{X} = (K^{0,2}[0-3]^{+4i}, K^1[12-15]^{+4i}, a^{i+1}_{13,15}, a^{i+4}_{14})$ by $S_{13,16}$
      Save $\mathcal{X}, MV_{14}[15]^{+4i}$ in a new hash table $\mathcal{H}_i[a^i_{13,15}, a^{i+3}_{14}, MV_{14}[11]^{+4i}]$
    end for
    for all $K^{3,4}$ **do** ▷ Parallel paths:
      $(S'_2, S_{28}) \leftarrow$ partial decryption/encryption from $S_{10,22}$ with K^{3-7}
      for all $a^{i+3}_{1,29}, a^i_{2,28}, MV_{1,29}[15]^{+4i}$ and all rows in $\mathcal{H}_i$, with $i = 0$ to 7 **do**
        Calc. $MV_{1,29}[11-14]^{+4i}, MV_{0,30}[23-26]^{+4i}, a^{i+4}_{1,29}, a^{i+1}_{2,28}$ by S'_2, S_{28}
        Consider equivalent row in $\mathcal{H}_i$ and save $MV_{0,30}[23-26]^{+4i}, MV_{1,29}[12-15]^{+4i}, MV_{14}[15]^{+4i}, K^{0,2}[0-3]^{+4i}, K^1[12-15]^{+4i}, a^{i+1}_{2,13,15,28}, a^{i+4}_{1,14,29}$ in a new hash table $\mathcal{Q}_i[a^i_{2,13,15,28}, a^{i+3}_{1,14,29}, MV_{1,14,29}[11]^{+4i}]$
      end for ▷ Integration phase:
      for all rows in $\mathcal{Q}_0$ with $a^0_{2,13,15,28} = (0,0,0,0)$ **do**
        for all rows in $\mathcal{Q}_1[a^1_{2,13,15,28}, a^4_{1,14,29}, MV_{1,14,29}[15]$ **do**
          for all rows in $\mathcal{Q}_2[a^2_{2,13,15,28}, a^5_{1,14,29}, MV_{1,14,29}[19]]$ & all $MV_{0,30}[3]$ **do**
            $S^L_0[11-14], S'^L_{30}[11-14], a^1_{0,30} \leftarrow$ partial decryption/encryption with $K^0[0-3], MV_{0,30}[0-3], MV_{1,29}[11-14]$
            for all rows in $\mathcal{Q}_3[a^3_{2,13,15,28}, a^6_{1,14,29}, MV_{1,14,29}[23]]$ with known $MV_{0,30}[3]$ & all $MV_{0,30}[7]$ **do**
              $S^L_0[15-18], S'^L_{30}[15-18], a^2_{0,30} \leftarrow$ partial decryption/encryption with $K^0[4-7], MV_{0,30}[4-7], MV_{1,29}[15-18], a^1_{0,30}$
              for all rows in $\mathcal{Q}_4[a^4_{2,13,15,28}, a^7_{1,14,29}, MV_{1,14,29}[27]]$ with known $MV_{0,30}[7]$ & all $MV_{0,30}[11]$ **do**
                $S^L_0[19-22], S'^L_{30}[19-22] \leftarrow$ partial decryption/encryption with $K^0[8-11], MV_{0,30}[8-11], MV_{1,29}[19-22], a^2_{0,30}$
                Get (P, S'_{30}) from $\mathcal{U}[P^L[11-22], P^R[23-10], S'^L_{30}[11-22], S'^R_{30}[23-10]]$
                if $(P^R[11], S'^R_{30}[11]) == MV_{0,30}[11]$ **then**
                  if there is a row in $\mathcal{Q}_5[a^5_{2,13,15,28}, a^0_{1,14,29}, MV_{1,14,29}[31]]$ with known $MV_{0,30}[11-14]$ **then**
                    if there is a row in $\mathcal{Q}_6[a^6_{2,13,15,28}, a^1_{1,14,29}, MV_{1,14,29}[3]]$ with known $MV_{0,30}[15-18]$ **then**
                      if there is a row in $\mathcal{Q}_7[a^7_{2,13,15,28}, a^2_{1,14,29}, MV_{1,14,29}[7]]$ with known $(MV_{0,30}[19-22], MV_{1,14,29}[11], a^3_{1,14,29})$ **then**
                        Try $K = K^0||K^1||...||K^7$
                      end if
                    end if
                  end if
                end if
              end for
            end for
          end for
        end for
      end for
    end for
  end for
end for

The ISC Int'l Journal of Information Security
ISeCure

July 2019, Volume 11, Number 2 (p. 188)
http://www.isecure-journal.org

Persian Abstract

حملات نقطه ثابت جدید به رمز قالبی GOST2 با بهبودهایی در پیچیدگی حافظه

سیاوش احمدی[1]، محمدرضا عارف[1]

[1]دانشگاه صنعتی شریف، تهران، ایران

رمز قالبی GOST در دهه ۱۹۷۰ طراحی و در سال ۱۹۸۹ به عنوان استاندارد GOST 28147-89 در اتحادیه جماهیر شوروی منتشر شد. به منظور بهبود امنیت رمز قالبی GOST پس از پیشنهاد حملات مختلف روی آن، طراحان نسخه اصلاح‌شده‌ای از GOST، با نام GOST2، را در سال ۲۰۱۵ منتشر کردند که دارای یک فرانمای کلید جدید و همچنین انتخاب صریح برای S-Boxهای آن بود. در این مقاله، با استفاده از سه بخش دقیقا یکسان از GOST2 و ایده نقطه ثابت، حملات نقطه ثابت بهبودیافته‌تری برای حذف کلیدهای اشتباه ارائه شده‌است. به عبارت دیگر، تمرکز حملات جدید روی کاهش پیچیدگی حافظه و در عین حال عدم تغییر پیچیدگی‌های دیگر است. نتایج کسب‌شده، کاهش قابل توجهی در پیچیدگی حافظه حملات را نشان می‌دهد، حال آن که پیچیدگی زمانی به میزان کمی در مقایسه با حملات نقطه ثابت قبلی افزایش پیدا کرده است. بر اساس دانش فعلی ما، کمترین پیچیدگی حافظه برای یک حمله روی نسخه دور-کامل رمز قالبی GOST2 در این‌جا ارائه شده است.

واژه‌های کلیدی: تحلیل رمز، حمله نقطه ثابت، رمز قالبی GOST2، ملاقات در میانه.