Research Article

Improved Single-Key Attacks on 2-GOST

Qiuhua Zheng, Yinhao Hu, Tao Pei, Shengwang Xu, Junzhe Yu, Ting Wu, Yanzhao Shen, Yingpei Zeng, and Tingting Cui

1 Hangzhou Dianzi University, Hangzhou 310018, China
2 Wuhan Maritime Communication Research Institute, Wuhan 430079, China
3 Science and Technology on Communication Security Laboratory, Chengdu 610041, China

Correspondence should be addressed to Tingting Cui; cuitingting@hdu.edu.cn

Received 2 June 2020; Revised 30 August 2020; Accepted 24 September 2020; Published 15 October 2020

Copyright © 2020 Qiuhua Zheng et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

GOST, known as GOST-28147-89, was standardized as the Russian encryption standard in 1989. It is a lightweight-friendly cipher and suitable for the resource-constrained environments. However, due to the simplicity of GOST’s key schedule, it encountered reflection attack and fixed point attack. In order to resist such attacks, the designers of GOST proposed a modification of GOST, namely, 2-GOST. This new version changes the order of subkeys in the key schedule and uses concrete S-boxes in round function. But regarding single-key attacks on full-round 2-GOST, Ashur et al. proposed a reflection attack with data of $2^{32}$ on a weak-key class of size $2^{224}$, as well as the fixed point attack and impossible reflection attack with data of $2^{64}$ for all possible keys. Note that the attacks applicable for all possible keys need the entire plaintext space. In other words, these are codebook attacks. In this paper, we propose single-key attacks on 2-GOST with only about $2^{32}$ data instead of codebook. Firstly, we apply 2-dimensional meet-in-the-middle attack combined with splice-cut technique on full-round 2-GOST. This attack is applicable for all possible keys, and its data complexity reduces from previous $2^{64}$ to $2^{32}$. Besides that, we apply splice-cut meet-in-the-middle attack on 31-round 2-GOST with only data of $2^{32}$. In this attack, we only need 8 bytes of memory, which is negligible.

1. Introduction

GOST block cipher [1] is known as GOST-28147-89 designed during the 1970s by the Soviet Union. It was standardized as the Russian encryption standard in 1989. As a lightweight-friendly block cipher, GOST is suitable for the resource-constrained environments such as RFID tags and sensor nodes.

GOST’s block size is 64 bits and key size is 256 bits. Round function adopts Feistel construction, in which there are a modular addition with subkey, 8 S-boxes and one rotation operation. However, the S-boxes used in GOST are not specified in the standard document. Each industry can use its own secret favored set of S-boxes to enhance the security of GOST. For example, the S-boxes used in the Central Bank of the Russian Federation is known in [2]. Besides that, the key schedule of GOST is extremely simple. 256-bit master key is divided into eight 32-bit words; then the 32-bit subkeys used in different round functions directly extract from these 8-word keys according to a special order.

Due to the simplicity of GOST’s key schedule, two attacks on full-round GOST were published by Isobe in [3] and Dinur et al. in [4] in 2011. In [3], Isobe combined the reflection property and meet-in-the-middle (MITM) attack to propose the single-key attack on full-round GOST. As a result, the key can be recovered with $2^{224}$ computations and $2^{32}$ known plaintexts. In [4], Dinur et al. introduced a new fixed point property as well as a better way to improve the attacks on full-round GOST. Given $2^{32}$ data, the memory complexity can reduce from $2^{64}$ to $2^{56}$ with the same time complexity $2^{224}$. Given $2^{64}$ data, the time complexity can be down to $2^{192}$. Although these attacks are not practical, they indicate the a priori in security of GOST.

In order to resist reflection attack and fixed point attack, the designers of GOST proposed a modification of GOST block cipher, named, 2-GOST [5]. In the new modification, there are two differences from original GOST. Firstly, the authors retained the same principle for key schedule as in GOST but changed the order of subkeys against existed...
attacks. Secondly, two concrete S-boxes were specified in the design document of 2-GOST for convenient cryptanalysis and better implementation.

Unfortunately, full-round 2-GOST still encounters reflection attack and fixed point attack. At FSE’17, Ashur et al. [6] proposed single-key attacks on it. Given 2^{32} data, the key can be recovered with 2^{192} computations by reflection attack. However, this attack only works for 2^{224} out of 2^{256} possible keys, which means this is a weak-key attack. For sake of valid for all possible keys, the authors proposed impossible reflection attack and fixed point attack. Both need 2^{64} known plaintexts. In other words, these are codebook attacks, since they use the entire plaintext space. These results are summarized in Table 1.

In this paper, our motivation is to propose attacks on 2-GOST with about 2^{32} data instead of codebook, further to indicate that the key schedule in modification version 2-GOST is not a good choice yet. Our contributions are summarized as follows:

- 2-dimensional MITM attack on full-round 2-GOST
- 2-dimensional MITM attack was proposed by Zhu and Gong in [7] to attack KATAN. Then, it has been applied on TWINE [8], GOST [4], and so on. This attack can improve the performance of general MITM attack, but attackers must be careful about the time complexity of accessing tables. In this paper, we apply 2-dimensional MITM attack combined with splice-cut technique [9] on full-round 2-GOST exploiting the weakness in key schedule. This attack is applicable for all possible keys with time complexity of 2^{52} full-round encryptions and memory complexity of 2^{28} bytes. Furthermore, the data reduced from previous 2^{64} (codebook) to 2^{32} chosen plaintexts under single-key setting. The result is shown in Table 1.

- Splice-cut MITM attack on 31-round 2-GOST
Based on some observations on key schedule and modular addition in the round function of 2-GOST. We apply MITM attack combined with splice-cut technique on reduced 31-round 2-GOST (0 ∼ 30 rounds). This attack is applicable for all possible keys with data complexity of 2^{32} chosen plaintexts and time complexity of 2^{52.9} full-round encryptions. It is important to stress that we only use 8-byte memory in this attack which is negligible. The result is shown in Table 1.

This paper is organized as follows. In Section 2, we introduce the specifications of GOST and 2-GOST. Then, in Section 3, we briefly describe the general MITM attack, splice-cut MITM attack, and 2-dimensional MITM attack. In Sections 4 and 5, we propose the 2-dimensional MITM attack on full-round 2-GOST and splice-cut MITM attack on 31-round 2-GOST, respectively. Lastly, we summarize this paper in Section 6.

2. Specifications of GOST and 2-GOST

GOST [1] is a bit-wise lightweight block cipher proposed by the Soviet Union. Its block size is 64 bits, key size is 256 bits, and total rounds are 32. GOST adopts Feistel construction as its round function, in which there are a nonlinear layer composed of eight bijective 4-bit S-boxes Sj, i = 0, 1, ..., 7 and a linear layer only containing a left rotation ≪11. Especially, subkeys are mixed with internal state by modular addition ≡ instead of traditional XOR. Please see the round function depicted in Figure 1.

The S-boxes Si: Fj ≡ Fj, i = 0, 1, ..., 7 used in GOST are bijective but not specified in the standard document. Each industry can choose its own secret favored set of S-boxes to enhance the security of GOST. Please refer to an example, the S-boxes used in the Central Bank of the Russian Federation in [2].

GOST’s key schedule is extremely simple. Each subkey uses one word (32 bits) of master key directly. Assume the master key K is divided into eight words K = K^0∥K^1∥K^7, each subkey k_i used in round function F_j, i = 0, 1, ..., 31 adopts one of K^i, K^j, ..., K^7. In detail, the first 24 rounds periodically use K^i, K^j, ..., K^7 as subkeys in ascending order; that is, that k_{i+j} = K^i, where i = 0, 1, ..., 7 and j = 0, 1, 2. The last 8 rounds use K^i, K^j, ..., K^7 as subkeys in descending order; that is, that k_j = K^{31−j}, i = 24, 25, ..., 31. All subkeys are summarized in Table 2.

2-GOST [5] is the modified version of GOST. It was designed by the same designers of GOST for the purpose of fixing weaknesses in key schedule against reflection attack and fixed point attack. The differences between 2-GOST and GOST are selection of S-boxes and the order of subkeys in the key schedule. Unlike uncertain S-boxes used in GOST, 2-GOST adopts two concrete bijective S-boxes. Since we only use the bijective property of S-box in this paper, we omit the specification of S-box here. Besides that, 2-GOST uses another order of subkeys comparing with GOST, which is summarized in Table 3.

3. Meet-in-the-Middle Attack

In this section, we will briefly recall general meet-in-the-middle (MITM) attack [10] combined with splice-cut technique [9] and 2-dimensional MITM attack [7].

3.1. General MITM Attack. The general MITM attack has two phases, one is the MITM phase and the other one is the brute-force testing phase.

Assume an n-bit block cipher E with k-bit secret key K is divided into two subciphers E_1, E_2, while K is divided into three key parts K_1, K_2, and K_3. K_1 is only used in E_1 and K_2 is only used in E_2 and K_3 is the rest of K. The framework of general MITM attack is shown in Figure 2 and the steps of this attack is summarized as follows.

(i) MITM phase: given a plaintext-ciphertext (P, C).
(1) For each possible K_1, guess each possible K_j, then compute v = E_j^-1 (P), and store all possible K_j into a table S indexed by v.
(2) For each possible K_2, compute the v’ = E_2^-1 (C), then access the v’-th entity of table S to extract
### Table 1: Summary of attacks on 2-GOST under single-key setting.

| Attack type     | Rounds | Data   | Time   | Memory (bytes) | No. of keys | Source |
|-----------------|--------|--------|--------|----------------|-------------|--------|
| Reflection      | 32     | $2^{32}$KP | $2^{190.2}$ | $2^{58.6}$ | $2^{224}$ | [6]    |
| Impossible reflection | 32     | $2^{33}$CP | $2^{252.5}$ | $2^{166.6}$ | $2^{226}$ | [6]    |
| Impossible reflection | 32     | $2^{44}$KP | $2^{253.5}$ | $2^{166.6}$ | $2^{226}$ | [6]    |
| Fixed point     | 32     | $2^{44}$KP | $2^{236}$  | $2^{138.2}$  | All        | [6]    |
| 2-D MITM        | 32     | $2^{32}$CP | $2^{252}$  | $2^{228}$     | All        | Section 4 |
| MITM            | 31     | $2^{32}$CP | $2^{322.9}$ | 8             | All        | Section 5 |

(i) The unit of time complexity is one full-round encryption; (ii) 2D MITM: 2-dimensional meet-in-the-middle attack; (iii) CP: chosen plaintext; KP: known plaintext.

---

#### 3.2. Splice-Cut MITM Attack.

In the chosen plaintext and chosen ciphertext settings, the first and the last rounds of the block cipher can be regarded as two successive rounds. Hence, data complexity is $2^{2n}$. All related keys in K1, K2, and K3 need to be involved. Assume an n-bit block cipher $\mathcal{E}$ with k-bit secret key K is divided into three subciphers $\mathcal{E}_1$, $\mathcal{E}_2$, and $\mathcal{E}_3$, while K is divided into three key parts $K_1$, $K_2$, and $K_3$. $K_1$ only is used in $\mathcal{E}_1$ and $\mathcal{E}_3$. $K_2$ is only used in $\mathcal{E}_2$, and $K_3$ is the rest of K. The framework of splice-cut MITM attack is shown in Figure 3, and the steps of the attack are summarized as follows:

(i) MITM phase:

1. Choose an n-bit state value $P'$.
2. Guess each possible $K_1$, then compute the $\nu = \mathcal{E}_3^{-1} \circ \mathcal{E}_2 \circ \mathcal{E}_1^{-1}(P')$, and store all $K_1$ into a table S indexed by $\nu$.
3. For each possible $K_2$, compute the $\nu' = \mathcal{E}_2(P')$, then access the $\nu'$-th entity of table S to extract $K_1$. Current $(K_1, K_2, K_3)$ is a candidate key.

(ii) Brute-force testing phase:

1. Test every candidate key with other plaintext/ciphertext pairs until only the right key is remained. The time and memory complexities are same as those in general MITM attack. However, the data complexity depends on $\mathcal{E}_1$ and $K_1$. Assume that $m$ bits of plaintext $P$ are not affected by $K_1$, then the data complexity is $2^{m}$.

#### 3.3. Dimensional MITM Attack.

This attack was proposed in [7]. It is suitable to attack ciphers whose key size is larger than block size. Assume an n-bit block cipher $\mathcal{E}$ with k-bit secret key K is divided into four subciphers $\mathcal{E}_1$, $\mathcal{E}_2$, $\mathcal{E}_3$, and $\mathcal{E}_4$. Key part $K_i$ is used in subcipher $\mathcal{E}_i$, $i = 1, 2, 3, 4$. The framework of 2-dimensional MITM attack is shown in Figure 4 and the steps of the attack are summarized as follows:

(i) MITM phase:

1. For each possible $K_1$, compute $\nu_1 = \mathcal{E}_1(P)$, and put $K_1$ into table $S_1$ indexed by the value of $\nu_1$;
| Round | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|-------|---|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| Subkey | $K^0$ | $K^1$ | $K^2$ | $K^3$ | $K^4$ | $K^5$ | $K^6$ | $K^7$ | $K^8$ | $K^9$ | $K^{10}$ | $K^{11}$ | $K^{12}$ | $K^{13}$ | $K^{14}$ | $K^{15}$ | $K^{16}$ | $K^{17}$ | $K^{18}$ | $K^{19}$ | $K^{20}$ | $K^{21}$ | $K^{22}$ | $K^{23}$ | $K^{24}$ | $K^{25}$ | $K^{26}$ | $K^{27}$ | $K^{28}$ | $K^{29}$ | $K^{30}$ | $K^{31}$ |
| Round | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|-------|---|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| Subkey | $K_0$ | $K_1$ | $K_2$ | $K_3$ | $K_4$ | $K_5$ | $K_6$ | $K_7$ | $K_0$ | $K_1$ | $K_2$ | $K_3$ | $K_4$ | $K_5$ | $K_6$ | $K_7$ | $K_0$ | $K_1$ | $K_2$ | $K_3$ | $K_4$ | $K_5$ | $K_6$ | $K_7$ | $K_0$ | $K_1$ | $K_2$ | $K_3$ | $K_4$ | $K_5$ |

Table 3: Key schedule of 2-GOST.
In such attack, the data complexity is \( kn/n \), while the memory complexity happens to store tables \( S_i \), \( i = 1, 2, 3 \).

Remark. In the 2-dimensional MITM attack model, the time of accessing tables is omitted in step 3b. However, it is much possible to be the main time complexity in some attacks. For example, Wen et al. indicated in [11] that the actual time complexity of 2-dimensional MITM attack on TWINE proposed in [8] exceeded the brute-force time. So in our attack on 2-GOST, we will take this part time into consideration.

4. 2-Dimensional MITM Attack on Full-Round 2-GOST

In this section, we apply 2-dimensional MITM attack combined with splice-cut technique on full-round 2-GOST.

Before formally introducing the attack, we firstly illustrate how to decide the partial matching (meeting) point.

\[
\begin{array}{c|c|c}
K_3 + K_1 & K_3 + K_2 & K_3 + K_1 \\
\end{array}
\]

Figure 3: Splice-cut MITM attack.

In the 2-dimensional MITM attack model, the time complexity of MITM attack on TWINE exceeds the brute-force time. So in our complexity of 2-dimensional MITM attack on TWINE proposed in [8] exceeded the brute-force time. So in our attack on 2-GOST, we will take this part time into consideration.

\[
\begin{array}{c|c|c}
\epsilon_1 & \epsilon_2 & \epsilon_3 \\
\end{array}
\]

Figure 4: 2-dimensional MITM attack.

For each possible \( K_4 \), compute \( v_2 = \delta_3^{-1}(C) \), and put \( K_4 \) into table \( S_3 \) indexed by the value of \( v_2 \);

(3) For each possible value of a:

(a) For each possible \( K_2 \), compute the \( v'_1 = \delta_2^{-1}(a) \), and check if \( v'_1 \) is in table \( S_1 \). If so, put \( (K_1, K_2) \) into table \( S_3 \).

(b) For each possible \( K_3 \), compute \( v'_2 = \delta_3^-(a) \) and check if \( v_2 \) is in table \( S_2 \).

So, put \( (K_1, K_2) \) into table \( S_3 \) indexed by the value of \( v_2 \).

Remark. (K_3, K_4) is also in table \( S_3 \). If true, current \( (K_1, K_2, K_3, K_4) \) is a candidate key.

(ii) Brute-force testing phase:

1. Test every candidate key with other plaintext/ciphertext pairs until only the right key is found.

For each possible value of a, there are \( 2^{2^{n_1}} \times 2^{2^{n_2}} \) possible keys. In brute-force testing phase, the attacker exhaustively searches the true key by using extra plaintext/ciphertext pairs. Finally, the time complexity \( C_{\text{comp}} \) of the attack in total is

\[
C_{\text{comp}} = 2^{|K_1| + 2^{|K_1|} + 2^{|a|} \times (2^{|K_1|} + 2^{|K_1|}) + \left(2^{|a|+k-|v_1|-|v_2|} + 2^{|a|+k-|v_1|-|v_2|} + 2^{|a|+k-|v_1|-|v_2|} - 2^n + \ldots \right)}.
\]

Along the forward direction, each bit on \( X_{i+1}^L, X_{i+1}^R \) can be deduced from \( X_{i-1}^L, X_{i-1}^R \) and subkey \( K_i \) as follows:

\[
\begin{align*}
X_{i+1}^L[a] &= X_{i+1}^L[a], \\
X_{i+1}^R[a] &= X_{i-1}^L[a] \oplus S(X_{i-1}^L[4j+3 \sim 0] \oplus K'[4j+3 \sim 0])[a-11],
\end{align*}
\]

where \( 4j \leq a \leq 11 \) (mod 32) < 4j+4, \( X[a] \) and \( X[a \sim b] \) denote the a-th bit and the a-th to b-th bits of state \( X \), respectively. From (3), we can find that no bits of \( K_i \) are needed to deduce every bit on \( X_i^L \) and at least 4 bits of \( K_i \) are involved to deduce one bit on \( X_i^R \) from \( X_{i-1}^L, X_{i-1}^R \). Especially, only \( X_{i}^R[a] \), \( 11 \leq j \leq 14 \), \( j = 0 \) can be deduced by 4 bits of \( K_i \), that is, \( K_i'[3 \sim 0] \). In order to guess key bits as few as possible to reduce the attack’s time complexity, \( X_{i}^L, X_{i}^R[14 \sim 11] \) is a good candidate as matching point. Along the backward direction, the way that each bit on \( X_{i}^L, X_{i}^R \) deduced from \( X_{i+1}^L, X_{i+1}^R \) and subkey \( K^{i+1} \) is similar with that along the forward direction because round function of 2-GOST adapts Feistel construction. Therefore, \( X_{i}^L[14 \sim 11], X_{i}^R \) is a good candidate as matching point as well. In a short, we select matching points from \( X_{i}^L, X_{i}^R[14 \sim 11] \) and \( X_{i}^R[14 \sim 11], X_{i}^R \) in our attack.
Next, we start to describe our attack on full-round 2-GOST. Its framework is shown in Figure 5. Firstly, we divide full-round 2-GOST into 5 subciphers: Round 0, Round 1~13, Rounds 13~18, Rounds 19~24, and Rounds 25~31, denoted as $\mathbb{E}_1$, $\mathbb{E}_2$, $\mathbb{E}_3$, $\mathbb{E}_4$, and $\mathbb{E}_5$, respectively. The actual start point $P'$ is the input of round 1 instead of the plaintext and two matching points are $(X^L_1[14 \sim 11], X^R_1)$ and $(X^L_{25}, X^R_{25}[14 \sim 11])$ depicted as in Figures 6 and 7, where $X[a \sim b]$ denotes the a-th to b-th bits of state $X$. In order to meet on $(X^L_1[14 \sim 11], X^R_{13})$, there are 224 bits key $K_1=(K^0_1, K^1_2, K^2_3, K^3_4, K^4_5, K^5_6)$ involved in subciphers $\mathbb{E}_2$ and 164 bits of key $K_2=(K^0_0[3 \sim 0], K^1_1, K^2_2, K^3_3, K^4_4, K^5_5, K^6_6)$ involved in subciphers $\mathbb{E}_3$. Similarly, in order to meet on $(X^L_{25}, X^R_{25}[14 \sim 11])$, there are 164 bits of key $K_3=(K^0_0, K^1_1, K^2_2, K^3_3, K^4_4, K^5_5, K^6_6)$ involved in subciphers $\mathbb{E}_4$, and 224 bits key $K_4=(K^0_0, K^1_1, K^2_2, K^3_3, K^4_4, K^5_5, K^6_6)$ involved in subciphers $\mathbb{E}_5$ and $\mathbb{E}_6$. The attack steps are as follows:

1. Guess each possible $K_1$, compute the value of $(X^L_1, X^R_1)$ by $v_1=\mathbb{E}_1(P')$, and put all possible $(K^0_3, K^4_3)$ into table $S_1$ indexed by $(v_1, K^1_2, K^2_3, K^5_6, K^6_5)$. 

2. Guess each possible $K_3$, compute the value of $(X^L_{25}, X^R_{25})$ by $v_2=\mathbb{E}_3^{-1}(P')$, and put $(K^3_2, K^0_0)$ into table $S_2$ indexed by $(v_2, K^0_1, K^1_4, K^2_3, K^3_4)$. 

3. For each possible value of $a$,
   a. Guess each possible $K_2$, compute the value of $(X^L_1, X^R_1)$ by $v_1=\mathbb{E}_2^{-1}(a)$, then extract the entity with index from table $S_1$ and store $(K^0_0[3 \sim 0], K^0_1)$ to table $S_3$ indexed by $(K^1_2, K^2_3, K^3_4, K^5_6, K^6_5)$. 
   b. Guess each possible $K_4$, compute the value of $(X^L_{25}, X^R_{25})$ by $v_2=\mathbb{E}_4(a)$, then extract the entity from table $S_2$ by index. 
   c. For each compatible $K_1 \cup K_3$, from step 3b, access table $S_1$ by index. If $K_1 \cup K_3$ is compatible with $K_4 \cup K_4$, current $K_1 \cup K_2 \cup K_3 \cup K_4 \cup K_4$ is a candidate key. Test the candidate key with other plaintext/ciphertext pairs.

In Step 1, the time complexity to build table $S_1$ is $2^{224 \times (12/32)} \approx 2^{222.6}$ full-round encryptions. Since there are $2^{196}$ entities in table $S_1$, on average, every entity contains $2^{28}$ 64-bit values $(K^3_3, K^4)$. Therefore, the memory complexity is $2^{224 \times 8} \approx 2^{227} \times 8$ bytes. Similarly, in Step 2, the time complexity is roughly $2^{224 \times (7/32)} \approx 2^{218.1}$ full-round encryptions, and the memory complexity is $2^{227}$ bytes.

Under each possible value of $a$, on average, $2^{264} \times 2^{28} = 2^{192}$ possible $K_1 \cup K_2$ will be stored into table $S_3$ in Step 3a. Since table $S_3$ has $2^{192}$ entities by index, each index contains one value $(K^0_0[3 \sim 0], K^0_1)$ on average. Thus, the time complexity of Step 3a is $2^{192} \times (6/32) \approx 2^{161.6}$ full-round encryptions and $2^{192}$ accesses. Assume one access roughly equals to one-round encryption. Then Step 3a needs $2^{161.6} + 2^{187} \approx 2^{187}$ full-round encryptions under each possible $a$. Next, in Step 3b, the time complexity is $2^{192} \times (6/32) \approx 2^{161.6}$ full-round encryptions. Since there are $2^{192}$ possible $K_1 \cup K_4$ remained after step 3b, the time complexity is $2^{192}$ accesses in Step 3c, that is, $2^{187}$ full-round encryptions. As a result, the time complexity of MITM phase is $2^{222.6} + 2^{221.8} + 2^{244} \times (2^{187} + 2^{161.6} + 2^{187}) \approx 2^{252}$ full-round encryptions. Meanwhile, there are $2^{248}$ candidate key remained in the brute-force testing phase, we need 4 plaintext/ciphertext pairs to filter such candidate keys. The time complexity is $2^{248} + 2^{248} \times 64 \approx 2^{248} + 2^{248} \times 64 \times 2 \approx 2^{248}$. Totally, the time complexity of the whole attack on full-round GOST2 is $2^{252} + 2^{248} \approx 2^{252}$ full-round encryptions. The memory complexity happens to build tables $S_1, S_2,$ and $S_3$, which is about $2^{227} + 2^{227} + 2^{196} \times 5 \approx 2^{228}$ bytes. Because $P=\mathbb{E}_1^{-1}(P')$ and 32-bit $K^0_0$ involved in $\mathbb{E}_1^{-1}$, the data complexity is $2^{35}$ chosen plaintexts.

5. MITM Attack on 31-Round 2-GOST

2-GOST is a modified version of GOST by changing the key schedule to avoid reflection attack and fixed point attack. In this section, we apply the general MITM attack combined with splice-cut technique on 31-round 2-GOST due to the new order of subkeys. By analyzing the key schedule of 2-GOST, we observe the fact that $K^0_1$ has no chance to be used from Round 2 to Round 13. On the other hand, $K^7$ has no chance to be used from Round 19 to Round 30 as well. Furthermore, Round 0 to Round 2 could be computed without $K^7$. Based on those observations, we construct a MITM attack on the reduced 31-round 2-GOST. Figure 8 shows an overview of the attack.

In this attack, we divide 31-round 2-GOST into three subciphers: Rounds 0 ~ 1, Rounds 2 ~ 15, and Rounds 16 ~ 30, denoted by $\mathbb{E}_1$, $\mathbb{E}_2$, and $\mathbb{E}_3$, respectively. The actual start point $P'$ is on $(X^L_2, X^R_2)$, and matching point is on $X^L_{16}[14 \sim 11]$. In order to compute the value of $X^L_{16}[14 \sim 11]$ from $(X^L_2, X^R_2)$ forward, there are 124 bits of key except $K^1_1[31 \sim 28]$ involved, while from ciphertext backward, there are 124 bits of key except $K^1_1[31 \sim 28]$ involved. Let $K_1$ denote the key bits only used in $\mathbb{E}_1$, $K_2$ denote the key bits only used in $\mathbb{E}_2$ and $\mathbb{E}_1$, and $K_3$ denote common key part among three subciphers. Here, $K_1=K^1_1[31 \sim 28]$, $K_2=K^1_1[31 \sim 28]$, and $K_3=(K^0_0, K^1_2[27 \sim 0], K^2_3, K^3_4, K^4_5, K^5_6, K^6_7[27 \sim 0])$. In detail, the attack process is as follows (Figure 9).

5.1. Complexity Evaluation. According to (2), in the MITM phase, the time complexity is about $2^{248} \times 2^4$ 14-round encryptions and $2^{248} \times 2^4$ 15-round encryptions, which is equal to $2^{51.8}$ 31-round encryptions. Meanwhile, in the brute-force testing phase, the time complexity is about $2^{252}$ 31-round encryptions. Totally, the time complexity of the whole attack is $2^{251.8} + 2^{252} \approx 2^{252.9}$ 31-round encryptions. Since $X^L_2[14 \sim 11]$, $X^R_2[31 \sim 11]$, and $X^L_{16}[6 \sim 0]$ are not affected by $K^1_1=K^1_1[31 \sim 28]$ (depicted in Figure 10); these 32 bits of plaintext can be fixed in advance. Therefore, the data complexity in MITM phase is $2^{32}$ chosen plaintexts. Regarding the required memory, it mainly happens to build table S, which needs about 8 bytes ($=2^4 \times 4$ bits).
(i) MITM phase:

For each possible $K_3$,

1. Choose an $n$-bit state value $P'$ on the input of round 2.

2. Guess each possible $K_1$, then compute the value $v \oplus E_{2}(P')$ on $XR_{16}\{14,\ldots,11\}$ and store $K_1$ into a table $S$ indexed by $v$.

3. For each possible $K_2$, compute the $v' \oplus E_{1}(E_{2}(P'))$, then check whether there

Figure 5: Splice-cut 2-dimensional MITM attack on full-round 2-GOST.

Figure 6: 36-bit matching point on the input of round 25 in 2-GOST.

Figure 7: 36-bit matching point on the input of round 13 in 2-GOST.
is any value of $K_1$ in the $v'$-th entity of $S$. If so, current $(K_1, K_2, K_3)$ is a candidate key.

(ii) Brute-force testing phase:

(1) Test every candidate key with other plaintext-ciphertext pairs until only the right key is remained

6. Conclusion

In this paper, we improve the single-key attacks on 2-GOST, a modification of GOST, with data of $2^{32}$ for all possible keys. Firstly, we apply 2-dimensional MITM attack combined with splice-cut technique on full-round 2-GOST. Its time and memory complexities are $2^{252.9}$ encryptions and $2^{213.256}$-bit blocks, respectively. Then, we apply splice-cut MITM attack on reduced 31-round 2-GOST. The time complexity is $2^{252.9}$ encryptions and memory complexity is negligible. Note that these attacks are still not practical to be implemented, but they indicate that the key schedule in the modification version 2-GOST is not a good choice yet.

Data Availability

The data used to support the findings of this study are included within the article.
Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

This work was supported by the NSFC Projects (nos. 61902100 and 61902098), Key Research, Development Program of Zhejiang Province (nos. 2020C01078 and 2019C01012), Foundation of Science and Technology on Communication Security Laboratory (no. 614203190105), and Natural Science Foundation of Zhejiang Province (no. Q20F020063).

References

[1] Russian National Bureau of Standards, Federal Information Processing Standard—Cryptographic Protection—Cryptographic Algorithm, GOST 28147-89, 1989, http://tools.ietf.org/html/rfc5830.
[2] OpenSSL, A Reference Implementation of GOST, http://www.openssl.org/source/.
[3] T. Isobe, “A single-key attack on the full GOST block cipher,” Fast Software Encryption, Springer, vol. 6733, pp. 290–305, Berlin, Germany, 2011.
[4] I. Dinur, O. Dunkelman, and A. Shamir, “Improved attacks on full GOST,” Fast Software Encryption, Springer, vol. 7549, pp. 9–28, Berlin, Germany, 2012.
[5] A. A. Dmukh, D. M. Dygin, and G. B. Marshalko, “A lightweight-friendly modification of GOST block cipher,” IACR Cryptology ePrint Archive, vol. 2015, p. 65, 2015, https://eprint.iacr.org/2015/065.pdf.
[6] T. Ashur, A. Bar-On, and D. Orr, “Cryptanalysis of GOST2,” IACR Transactions on Symmetric Cryptology, vol. 1, pp. 203–214, 2017.
[7] B. Zhu and G. Gong, “Multidimensional meet-in-the-middle attack and its applications to KATAN32/48/64,” Cryptography and Communications, vol. 6, no. 4, pp. 313–333, 2014.
[8] Ö. Boztas, F. Karakoç, and M. Çoban, “Multidimensional meet-in-the-middle attacks on reduced-round TWINE-128,” Lightweight Cryptography for Security and Privacy. LightSec 2013. LNCS, Springer, vol. 8162, pp. 55–67, Berlin, Heidelberg, 2013.
[9] K. Aoki and Y. Sasaki, “Preimage attacks on one-block MD4, 63-step MD5 and more,” SAC 2008. LNCS, Springer, vol. 5381, pp. 103–119, Heidelberg, Germany, 2009.
[10] W. Diffie and M. E. Hellman, “Special feature exhaustive cryptanalysis of the NBS data encryption standard,” Computer, vol. 10, no. 6, pp. 74–84, 1977.
[11] L. Wen, M. Wang, A. Bogdanov, and H. Chen, “Note of multidimensional MITM attack on 25-round TWINE-128,” IACR Cryptology ePrint Archive, vol. 2014, p. 425, 2014, https://eprint.iacr.org/2014/425.pdf.