Cryptanalysis of GOST2

Tomer Ashur[1], Achiya Bar-On[2], and Orr Dunkelman[3]

[1] Dept. Electrical Engineering, ESAT/COSIC, KU Leuven, and iMinds, Belgium
[2] Department of Mathematics, Bar-Ilan University, Israel
[3] Computer Science Department, University of Haifa, Israel

Abstract. GOST 28147 is a 256-bit key 64-bit block cipher developed by the USSR, later adopted by the Russian government as a national standard. In 2010, GOST was suggested to be included in ISO-18033, but was rejected due to weaknesses found in its key schedule.
In 2015, a new version of GOST was suggested by Russia's standardization body (TC 26), with the purpose of mitigating such attacks. In this paper, we show that similar weaknesses exist in the new version as well. More specifically, we present a fixed-point attack on the full cipher with time complexity of 2^{237} encryptions. We also present a reflection attack with time complexity of 2^{192} for a key that is chosen from a class of 2^{224} weak keys. Finally, we discuss an impossible reflection attack and several possible related-key attacks.
Keywords: Block ciphers, cryptanalysis, GOST, GOST2, reflection attack, fixed-point attack, related-key attack, impossible reflection attack

1 Introduction

GOST [13] is a block cipher designed during the 1970's by the soviet union as an alternative to the American DES [16]. Similarly to DES it has a 64-bit Feistel structure, employing 8 S-boxes and is intended for civilian use. Unlike DES, it has a much larger key (256 bits instead of just 56), more rounds (32 compared with DES' 16), and it uses different sets of S-boxes. What is unique about GOST is that the S-boxes are not an integral part of the standard, and in fact, they were kept secret, which allowed the government to give different sets of S-boxes to different users.

After the USSR had been dissolved, GOST was accepted as a Russian standard in [13], and was proposed to be included in ISO-18033. At the time, GOST seemed like a natural candidate to be included in ISO-18033. As was shown in [12], it can be implemented with only 651–1017 GE, depending on the choice of S-boxes. From a security point of view, although there have been several attacks such as [10] in the related-key model, the only attack on the full GOST in the single key model was published in 2008 in [9], and was limited to a weak-key class.

However, as a result of the renewed interest, Isobe presented in [8] an improvement to [9] that eliminates the weak-key assumption resulting in an attack with time complexity of 2^{224}. A year later, as the attack was improved by Dinur, Dunkelman, and Shamir in [5], and as new attacks were presented by Courtois in [2], the idea to standardize GOST was rejected.

In 2015, the Technical Committee for Standardization (TC 26) added a new block cipher with a 128-bit block and a 256-bit key to the standard under the name *Kuznyechik* (Russian for Grasshopper) [7]. This cipher was later published in RFC-7801, and was recently suggested to be included in ISO/IEC 18033-3.

The new 128-bit cipher does not obsolete the old 64-bit cipher, and in fact, authors from TC 26 also published in 2015 a modified version of the 64-bit variant that supposedly resists previous attacks [6]. The modified version differs from the original GOST in two aspects: (i) it has a different key schedule, designed to avoid previous attacks and, (ii) it makes an explicit choice for the S-boxes.

In this paper we show that the modified version is vulnerable to the same kind of attacks as the original one. We adapt attacks from the original GOST to the new version. This results in three types of attacks. First, we present a reflection attack on a weak-key class of size 2^{224}. For the cases where the key is not one of these weak keys, we present an impossible reflection attack with time complexity of $2^{253.56}$. We then present a fixed-point attack on the full cipher applicable to all keys with time complexity of 2^{237}. These attacks, which are the main contribution of this paper, are presented in Table 1. Finally, we briefly discuss related-key differential attacks applicable to GOST2.

Type of attack	Time	Data	Memory (bytes)	No. of keys	Section
Fixed point	2^{237}	2^{64}KP	2^{196}	All	4.2
Reflection	2^{192}	2^{32}KP	$2^{68.58}$	2^{224}	4.1
Impossible reflection	$2^{253.56}$	2^{63}CP	$2^{166.58}$	$2^{256} - 2^{224}$	A
Impossible reflection	$2^{254.56}$	2^{64}KP	$2^{166.58}$	$2^{256} - 2^{224}$	A

^{KP} Known plaintexts
^{CP} Chosen plaintexts

Table 1. Single-key Attacks on the Full GOST2

It is important to stress that although all these attacks require less effort than exhaustive search, their complexities are still impractical to be implemented. However, unlike the GOST block cipher which has withstood cryptanalysis for two decades, the modified GOST is a new cipher and should be considered carefully before added to any standard.

This paper is organized as follows: in Section 2 we present the GOST block cipher, and the modified version which we refer to as GOST2. Then, in Section 3 we discuss previous work relating to GOST. In Section 4 we present our attacks for GOST2 and discuss a previously known attack. We summarize the paper in Section 5.

2 The GOST and GOST2 Block Ciphers

The GOST block cipher has a 64-bit Feistel structure using a 256-bit key. The 64-bit block is treated as two words of 32-bit each which are referred to as the "left word" and the "right word". The state in round i is denoted by $S_i = (L_i, R_i)$ where L_i and R_i are the left and right words entering round i, respectively. In each round, a 32-bit to 32-bit round function $\mathcal{F}$ is applied to the right word and the round's subkey K_i. The output of $\mathcal{F}$ is XORed to the left input word, and the words are swapped. We get that

$$\begin{aligned} R_{i+1} &= \mathcal{F}(R_i, K_i) \oplus L_i \\ L_{i+1} &= R_i \end{aligned}$$

where R_0 is the right half of the plaintext, L_0 is the left half of it, K_i is the i^{th} round subkey, and $\oplus$ is the XOR operation. We say that $S_{i+1} \leftarrow R_{K_i}(S_i)$ (resp., $S_{i-1} \leftarrow R^{-1}_{K_i}(S_i)$) is the 1-round encryption (resp., decryption) of S_i with the appropriate subkey. In the sequel we also use $\boxplus$ to denote addition modulo 2^{32}, $||$ for concatenation of strings, $\lll j$ to denote left cyclic rotation by j bits, and X[i–j] to denote bits i to j of X.

Inside the round function, the input is mixed with the round's 32-bit subkey K_i using modular addition. Then, it is split into 8 chunks of 4 bits entering the eight S-boxes. Finally, the output of the S-boxes is left rotated by 11 bits. This is repeated 32 times, for rounds numbered 0–31, and the output of the last round is used as the ciphertext. A schematic view of 1-round GOST is depicted in Figure 1.

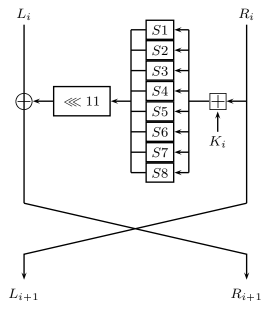

Fig. 1. One round of GOST. The symbols $\boxplus$, $\oplus$, and $\lll_j$ are used for modular addition, bitwise addition, and cyclic left rotation by j bits, respectively.

The differences between the original version of GOST and the modified version are only in the key schedule and the choice of S-boxes. In both versions, the key schedule takes the 256-bit key K and splits it into 8 subkeys of 32 bits denoted K^0 to K^7.

In the original GOST, the first 24 rounds used the subkeys in their cyclic order (i.e., $K_0 = K_8 = K_{16} = K^0$, $K_1 = K_9 = K_{17} = K^1$, etc.). In the final 8 rounds (i.e., rounds 24–31), the subkeys were used in a reverse order such that K^7 was used in round 24, K^6 was used in round 25, etc. In the modified version, the key schedule has changed, but the keys are still used in an ascending cyclic order in rounds 0–7, 8–15, and 16–23, and in a descending cyclic order in rounds 24–31. The order of the subkeys for both versions is presented in Table 2. From here on, we refer to the modified version of GOST presented in [6] as GOST2. Also, whenever we want to stress that K^j is used in round i (i.e., $K^j = K_i$), we write it as K_i^j.

Round	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Subkey (GOST)	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
Subkey (GOST2)	K^0	K^1	K^2	K^3	K^4	K^5	K^6	K^7	K^3	K^4	K^5	K^6	K^7	K^0	K^1	K^2

Round	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Subkey (GOST)	K^0	K^1	K^2	K^3	K^4	K^5	K^6	K^7	K^7	K^6	K^5	K^4	K^3	K^2	K^1	K^0
Subkey (GOST2)	K^5	K^6	K^7	K^0	K^1	K^2	K^3	K^4	K^6	K^5	K^4	K^3	K^2	K^1	K^0	K^7

Table 2. The order of subkeys in GOST and GOST2

Another change made to the design of GOST is the proposal of concrete S-boxes to be used in the standard. The designers suggested to use the permutation Π_1 as the first 4 S-boxes, and the permutation Π_2 as the other 4 S-boxes. Both S-boxes are presented in Table 3. We note that although our complexity analyses use the fact that the S-boxes are bijective in order to derive average case complexities, we do not exploit possible weaknesses in these S-boxes. We refer the interested reader to [6] for the rationale behind the choice of S-boxes.

Input	0_x	1_x	2_x	3_x	4_x	5_x	6_x	7_x	8_x	9_x	A_x	B_x	C_x	D_x	E_x	F_x
Π_1	6_x	A_x	F_x	4_x	3_x	8_x	5_x	0_x	D_x	E_x	7_x	1_x	2_x	B_x	C_x	9_x
Π_2	E_x	0_x	8_x	1_x	7_x	A_x	5_x	6_x	D_x	2_x	4_x	9_x	3_x	F_x	C_x	B_x

Table 3. The proposed S-boxes for GOST2

3 Previous Work

Several works such as [1, 10, 15] were able to attack reduced versions of GOST, or attack the cipher in the related-key model. The first attack in the single key model on the full cipher is due to Kara in [9]. Kara presented a reflection attack on a weak-key class of size 2^{224}, using 2^{32} chosen plaintexts in 2^{192} time.

Kara's attack was improved by Isobe in [8]. Isobe was able to eliminate the weak-key assumption, presenting an attack using 2^{224} time, 2^{64} memory, and 2^{32} known plaintexts. Another improvement was made a year later by Dinur, Dunkelman and Shamir in [5] where a new fixed-point property was presented. They further presented a new attack algorithm that can use either this property or Isobe's reflection property, and several trade-offs for the attack complexities, allowing the adversary to optimize for time, memory, or data complexities.

Other attacks for GOST are either inferior in their complexities, attack less rounds, or work in the related-key model. All single key attacks on the full GOST are presented in Table 4. We stress that none of these attacks is applicable to GOST2, and that to the best of our knowledge, we are the first to attack this new cipher.

Reference	Data	Memory	Time	Type	S-boxes
[8]	2^{32}CP	2^{64}	2^{224}	Reflection	Bijective
[3]	2^{64}KP	2^{64}	2^{248}	Algebraic	Russian Banks [11]
[4]	2^{64}KP	2^{64}	2^{226}	Differential	Russian Banks [11]
[5]	2^{64}KP	2^{36}	2^{192}	fixed point	any
[5]	2^{64}KP	2^{19}	2^{204}	fixed point	any
[5]	2^{32}KP	2^{36}	2^{224}	Reflection	any
[5]	2^{32}KP	2^{19}	2^{236}	Reflection	any

KP Known plaintexts

CP Chosen plaintexts

Table 4. Single-key Attacks on the Full GOST

4 Attacking the Modified Version of GOST

In this section, we show how to adapt previous attacks against GOST to GOST2. This shows that GOST2 fails to deter the exact attacks it was supposed to avoid, thus casting doubt on its design methodology, and most notably on its key schedule.

We start by discussing some properties of Feistel structures. The first of which is the *reflection property* [9]. A reflection point is a state $S = (L, R)$ such that $L = R$. We use the following lemma which is a well-known property of Feistel networks:

Lemma 1. *if S is a reflection point, then for any subkey k, $R_k(S) = R_k^{-1}(S)$.*

Lemma 1 leads to the following corollary:

Corollary 1. *if S is a reflection point, then for any sequence of subkeys K_i, $K_{i+1}, \ldots, K_{i+j}$, it holds that*

$$R_{K_i}(R_{K_{i+1}}(\ldots(R_{K_{i+j}}(S)))) = R^{-1}_{K_i}(R^{-1}_{K_{i+1}}(\ldots(R^{-1}_{K_{i+j}}(S)))).$$

We use Corollary 1 to present in Section 4.1 reflection attacks for a weak-key class of size 2^{224} with time complexity of 2^{161} using 2^{64} known plaintexts and an attack with time complexity of 2^{192} using 2^{32} known plaintexts. In Appendix A we present a complementary attack, i.e., an attack that works when the key is chosen among the other $2^{256} - 2^{224}$ keys.

In Section 4.2 we present a fixed-point attack on the full GOST2 with time complexity 2^{237}. A fixed point is an intermediate state S such that

$$S = R_{K_i}(R_{K_{i+1}}(\ldots(R_{K_{i+j}}(S))))$$

for some sequence of keys $K_i, \ldots, K_{i+j}$ used in rounds i to $i + j$. As we can see from Table 2, rounds 10–15 use the same keys as rounds 16–21. This leads to the following observation used in Section 4.2:

Observation 1 *If S_{10} is a fixed point with respect to rounds 10–15 (i.e., $S_{10} = S_{16}$), then $S_{10} = S_{16} = S_{22}$.*

We conclude by briefly discussing several related-key differential attacks, and their effect on the security of GOST2.

4.1 A Reflection Attack for GOST2

In this attack, we make use of a reflection point in the intermediate state. As can be seen from Table 2, the order of keys in rounds 18–31 is:

$$K^7_{18}, K^0_{19}, K^1_{20}, K^2_{21}, K^3_{22}, K^4_{23}, K^6_{24}, K^5_{25}, K^4_{26}, K^3_{27}, K^2_{28}, K^1_{29}, K^0_{30}, K^7_{31}.$$

We assume that the key belongs to a weak-key class where $K^5_{24} = K^6_{25}$.[4] Then, if the intermediate state before round 25 (i.e., S_{25}) is a reflection point, we get due to Corollary 1 that $C = S_{32} = S_{18}$, and thus the number of effective encryption rounds is 18 rather than 32. The probability of any state in GOST2 to be a reflection point is 2^{-32}, which means that an adversary observing 2^{32} plaintexts should encounter on average one reflection point in S_{25}.

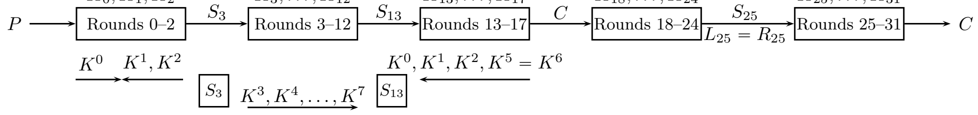

Fig. 2. A schematic description of the reflection attack.

[4]In Appendix A we show how to mount an impossible reflection attack when $K^5_{25} \neq K^6_{25}$.

Algorithm 1 Pseudocode of the attack for GOST2.

```
Input: $2^{32}$ pairs of known plaintexts and ciphertexts - $\{P_i, C_i\}$.

  for $S_3, K_{16}^5 = K_{17}^6$ do {Outer loop}
    for $(P_i, C_i), K_0$ do {Inner loop 1}
      $K_1, K_2 \leftarrow \text{Solve}(P_i, S_3, K_0)$
      $S_{13} \leftarrow R_{K_{13}}^{-1}(R_{K_{14}}^{-1}(R_{K_{15}}^{-1}(R_{K_{16}}^{-1}(R_{K_{17}}^{-1}(C_i = S_{18})))))$
      $T[S_{13}] \leftarrow (P_i, K_0, K_1, K_2)$
    end for
    for $K_3, K_4, K_7$ do {Inner loop 2}
      $S_{13} \leftarrow R_{K_{12}}(R_{K_{11}}(R_{K_{10}}(R_{K_9}(R_{K_8}(R_{K_7}(R_{K_6}(R_{K_5}(R_{K_4}(R_{K_3}(S_3))))))))))$
      $(P_i, K_0, K_1, K_2) \leftarrow T[S_{13}]$
      $\text{TRY}(K^0, K^1, K^2, K^3, K^4, K^5, K^6, K^7)$
    end for
  end for
```

The description of the attack is as follows (cf. Figure 2): The adversary observes 2^{32} pairs of plaintexts and ciphertexts denoted by (P_i, C_i). Then, she uses an outer loop and two inner loops. In the outer loop she iterates over the values of S_3 and $K_{16}^5 = K_{17}^6$. Then, in the first inner loop, she iterates over the known plaintexts and ciphertext pairs (P_i, C_i), and over all possible values for K_0. She computes $S_1 = R_{K_0}(P_i)$ and uses S_3 to retrieve K_1 and K_2 which can be computed through the 2-round Feistel network without additional complexity by solving the equations

$$L_3 \oplus L_1 = F(R_1, K_1)$$

and

$$R_3 \oplus R_1 = F(L_3, K_2)$$

with respect to known R_3 and L_3.[5] She also obtains $S_{13} = R_{K_{13}}^{-1}(R_{K_{14}}^{-1}(R_{K_{15}}^{-1}(R_{K_{16}}^{-1}(R_{K_{17}}^{-1}(C_i = S_{18})))))$ using the newly obtained $K_2 = K_{15}$ and $K_1 = K_{14}$ and the interim $K_0 = K_{13}, K_{16} = K_{17}$. Finally, she stores in $T[S_{13}]$ the tuple P_i, K_0^0, K_1^1, K_2^2.

In the second inner loop, which is run sequentially after the previous one, she iterates over all possible values of K_3^3, K_4^4, and K_7^7 for the encryption of S_3 through rounds 3–12 to obtain S_{13}. Then, she fetches the tuple (P_i, K_0, K_1, K_2) from $T[S_{13}]$ (note that for every S_{13} there is one such tuple on average). At this point, the adversary holds a candidate key $K = (K^0, K^1, K^2, K^3, K^4, K^5, K^6, K^7)$ which can be tested using a trial encryption. A pseudo-code describing the attack is presented in Algorithm 1.

The data complexity of the attack is 2^{32} known plaintexts, suggesting that the reflection property holds for one plaintext on average. The outer loop iterates over 2^{64} possible values for S_3 and over 2^{32} possible values for $K_{16}^5 = K_{17}^6$ making its time complexity 2^{96}. The first inner loop iterates over the 2^{32} known pairs

[5]When the S-box layer is bijective $K_1 = S^{-1}((L_1 \oplus L_3) \ggg 11) \boxminus R_1)$ and $K_2 = S^{-1}((R_1 \oplus R_3) \ggg 11) \boxminus L_3$.

of plaintexts and ciphertexts, and over 2^{32} candidates for K_0, making its time complexity 2^{64}. The second inner loop iterates over 2^{32} candidates for each of the values of K_3, K_4, and K_7, i.e., it runs 2^{96} times. The total time complexity is therefore $2^{96} \cdot (2^{64} + 2^{96}) \approx 2^{192}$. The size of the table T is 2^{64} lines of 192 bits, resulting in memory complexity of $2^{68.58}$ bytes.[6]

4.2 An Attack on the Full GOST2

In this section we show how to mount a fixed-point attack against GOST2 using Observation 1. The probability that S_{10} is a fixed point with respect to rounds 10–15 is 2^{-64} suggesting that an adversary would observe one such message on average after encrypting 2^{64} plaintexts (i.e., the entire codebook). A second observation is that knowing the input and output to a 3-round Feistel network, an adversary can iterate over some of the bits in the first and last rounds and check if they match in the middle round to filter out wrong keys. In this attack, we guess the 12 least significant bits of K_0 and K_2, which are denoted by $K_0[0–11]$ and $K_2[0–11]$, respectively, and match bits 11–22 in R_1. This gives $2^{12} \cdot 2^{12} \cdot 2^{-12} = 2^{12}$ suggestions for 24 bits of the key. By additionally guessing the carry bit in position 11, the adversary can compute key bits 12–19 in K_1 denoted by $K_1[12–19]$.

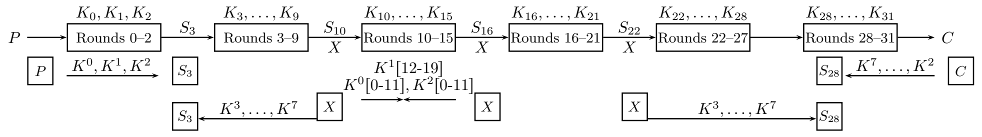

Fig. 3. A schematic description of the fixed-point attack.

The attack procedure is as follows (cf. Figure 3): collect all 2^{64} pairs (P_i, C_i). Then, for each pair (P_i, C_i) of plaintext and ciphertext, the adversary iterates over all possible values of $K^0_{28}, K^1_{29}, K^2_{30}, K^7_{31}$. She decrypts C_i through rounds 31–28 to obtain S_{28} and encrypts P_i through rounds 0–2 to obtain S_3. She uses a table T to store in row $S_3 || S_{28}$ the subkey values $K^0_{28}, K^1_{29}, K^2_{30}$, and K^7_{31} supposedly leading from S_3 to S_{28}.

In the second part she iterates over all possible S_{10} and all possible K_3, K_4, K_5, K_6, K_7 to encrypt S_{10} through rounds 10–12 and obtain S_{13}. From S_{13} and $S_{16} = S_{10}$ she computes 2^{13} candidates for $K_0[0–11], K_1[12–19]$, and $K_2[0–11]$. Then, she decrypts S_{10} through rounds 9–3 to obtain S_3 and encrypts $S_{22} = S_{10}$ through rounds 22–28 to obtain S_{28}. Using the interim S_3 and S_{28} she fetches

[6]The memory complexity can be reduced to $2^{67.58}$ by storing only P_i and K_0 in the first inner loop and recomputing K_1 and K_2 in the second inner one.

Algorithm 2 Pseudocode of the attack for GOST2.

Input: 2^{64} pairs of known plaintexts and ciphertexts.

```
for $(P_i, C_i), K^0_{28}, K^1_{29}, K^2_{30}, K^7_{31}$ do {First part}
    $S_{28} \leftarrow R^{-1}_{K_{28}}(R^{-1}_{K_{29}}(R^{-1}_{K_{30}}(R^{-1}_{K_{31}}(C_i))))$
    $S_3 \leftarrow R_{K_2}(R_{K_1}(R_{K_0}(P_i)))$
    $T[S_3||S_{28}] \leftarrow (K^0_{28}, K^1_{29}, K^2_{30}, K^7_{31})$
end for
for $S_{10} = S_{16} = S_{22}, K^3_3, K^4_4, K^5_5, K^6_6, K^7_7$ do {Second part - outer loop}
    $S_{13} \leftarrow R_{K_{12}}(R_{K_{11}}(R_{K_{10}}(S_{10})))$
    for $K_0[0–11], K_2[0–11], K_1[10]$ do
        $(K_0[0–11], K_1[12–19], K_2[0–11]) \leftarrow \mathrm{SOLVE}(S_{16}, S_{13}, K_0[0–11], K_2[0–11], \mathrm{Carry})$
    end for
    $S_3 \leftarrow R^{-1}_{K_3}(R^{-1}_{K_4}(R^{-1}_{K_5}(R^{-1}_{K_6}(R^{-1}_{K_7}(R^{-1}_{K_8}(R^{-1}_{K_9}(S_{10})))))))$
    $S_{28} \leftarrow R_{K_{27}}(R_{K_{26}}(R_{K_{25}}(R_{K_{24}}(R_{K_{23}}(R_{K_{22}}(S_{22}))))))$
    $(K^0, K^1, K^2, K^7) \leftarrow T[S_3||S_{28}]$
    $\mathrm{Filter}(K^0, K^1, K^2, K^7)$
    $\mathrm{TRY}(K^0, K^1, K^2, K^3, K^4, K^5, K^6, K^7)$
end for
```

2^{64} possible suggestions for K^0_0, K^1_1, K^2_2, and K^7_7 on average from T. She uses $K_0[0–11]$, $K_1[12–19]$, $K_2[0–11]$, and K_7 to further discard wrong values and tries the remaining keys.

The time required for building the table is 2^{64+128}. The time for the second part is $2^{64+5\cdot 32}$ in the outer loop, and 2^{13} in the inner loop. The probability that a row in the table matches the 2^{128} bits of S_3 and S_{28} that were obtained in the second part is $(2^{-64})^2 = 2^{-128}$. These keys are then filtered by K_7 and the known bits from K_0, K_1, and K_2 with probability $(2^{-12})^2 \cdot 2^{-8} = 2^{-32}$. The number of keys suggested by the attack is therefore $2^{64+128} \cdot 2^{64+5\cdot 32+13} \cdot (2^{-64})^2 \cdot 2^{-32} \cdot (2^{-12})^2 \cdot 2^{-8} = 2^{7\cdot 32+13} = 2^{237}$ possible keys, making the time complexity of the attack roughly 2^{237} full GOST2 encryptions, using data complexity of 2^{64} known plaintexts and ciphertexts, and memory complexity of $2^{64+128+4} = 2^{196}$ bytes. A pseudocode of the attack procedure is presented in Algorithm 2.

4.3 Related-key Differential Properties in GOST2 and their Effect on the Security of GOST2

An interesting omission by the authors of [6] is mitigation against related-key differential attacks. Indeed, in their introduction they discuss the works in [5,8], but not any of the works in [4, 10, 14]. It seems that some attacks were not addressed in the design of GOST2, and that the cipher is still vulnerable to these attacks. In this subsection we mention three possible related-key differential attack.

A 32-round Related-key Differential Distinguisher with Probability 1 (A Complemention Property) Assume two related keys differing in the most significant bit of each of the subkeys. A pair of plaintexts also differing in the most significant bits of the two words will result in a pair of ciphertexts differing in the same bits with probability 1. This characteristic can be used to distinguish the full GOST2 from a random permutation in the related-key model, with probability 1 using only 2 chosen plaintexts. It can also be used to speed up exhaustive search on the full GOST2 by a factor of 2. Moreover, using reduced-round variants of this observation can speed up all of our attacks by a factor of 2.

A 16-round Related-key Differential Distinguisher with Probability 1 Assume that a pair of plaintexts (P, P') whose intermediate states (S_8, S'_8) differ only in bit 31 (i.e., the most significant bit of the right half). Then, a pair of related keys having a difference in the most significant bit of the subkeys indexed by an odd number (i.e., K_1, K_3, K_5, and K_7), leads to the same difference before round 24 with probability 1. We can extend this differential characteristic 4 rounds backwards and 3 rounds forward with probability $(\frac{3}{4})^4$. This gives a truncated input difference with 8 known bits in S_4 leading to a truncated output difference with 28 known bits in S_{27}. By guessing K_0, K_1, K_2, K_3 and K_7 we can bridge the distance between the plaintext and S_4, and between the ciphertext and S_{27}. The attack can recover the full key with 2^{36} chosen plaintexts, using 2^{38} bytes of memory, in 2^{226} time, and 2 related keys.

A 30-round Related-key Differential Distinguisher with Probability $\mathbf{2^{-30}}$ As was shown by Ko et al. in [10], a difference in the second most significant bit can be canceled by a related-key having a difference in the same position with probability $\frac{1}{2}$. Ko et al. also show how to concatenate this transition to obtain a related-key differential characteristic for 30 rounds with probability 2^{30}. We observe that the same property holds also for GOST2, and that the key recovery attack of [10] can be adapted in a straightforward way.

5 Summary

In this paper we discussed the security of GOST2, a variant of GOST aimed to mitigate previous attacks against the cipher. We showed that the proposed fixes are insufficient in resisting these attacks, which rely on the key schedule, and that they can be adapted to the new version.

We presented a reflection attack for a weak-key class of 2^{224} keys using 2^{192} full GOST2 encryptions, 2^{32} known plaintexts, and $2^{68.58}$ bytes of memory. We also presented a fixed-point attack on the full cipher applicable to all keys using 2^{237} full GOST2 encryptions, 2^{64} known plaintexts, and 2^{196} bytes of memory. Finally, we discussed possible related-key differential attacks and showed that related-key attack on GOST are applicable to GOST2.

To conclude, it seems that the change in the key schedule was insufficient to offer 256-bit security by the new cipher. Hence, we recommend standardization bodies to avoid adopting GOST2. We also encourage the designers to consider a better key schedule, possibly with rotations, XORs, or round dependent constant additions.

Acknowledgments. The first author was supported in part by the Research Fund KU Leuven, OT/13/071. The second author was supported in part by the Check Point Institute for Information Security. The third author was supported in part by the Israeli Science Foundation through grant No. 827/12 and by the Commission of the European Communities through the Horizon 2020 program under project number 645622 PQCRYPTO.

References

1. Biham, E., Dunkelman, O., Keller, N.: Improved Slide Attacks. In: Biryukov, A. (ed.) Fast Software Encryption, 14th International Workshop, FSE 2007, Luxembourg, Luxembourg, March 26-28, 2007, Revised Selected Papers. Lecture Notes in Computer Science, vol. 4593, pp. 153–166. Springer (2007)
2. Courtois, N.T.: Security Evaluation of GOST 28147-89 In View Of International Standardisation. Cryptology ePrint Archive, Report 2011/211 (2011), http://eprint.iacr.org/
3. Courtois, N.T.: Security Evaluation of GOST 28147-89 in View of International Standardisation. Cryptologia 36(1), 2–13 (2012)
4. Courtois, N.T., Misztal, M.: Differential Cryptanalysis of GOST. IACR Cryptology ePrint Archive 2011, 312 (2011), http://eprint.iacr.org/2011/312
5. Dinur, I., Dunkelman, O., Shamir, A.: Improved Attacks on Full GOST. In: Canteaut, A. (ed.) Fast Software Encryption - 19th International Workshop, FSE 2012, Washington, DC, USA, March 19-21, 2012. Revised Selected Papers. Lecture Notes in Computer Science, vol. 7549, pp. 9–28. Springer (2012)
6. Dmukh, A., Dygin, D., Marshalko, G.: A lightweight-friendly modifcation of GOST block cipher. IACR Cryptology ePrint Archive 2015, 65 (2015), http://eprint.iacr.org/2015/065
7. Federal Agency on Technical Regulation and Metrology: National Standard of the Russian Federation. GOST R 34.12–2015 (2015), http://tc26.ru/en/standard/gost/GOST_R_34_12_2015_ENG.pdf
8. Isobe, T.: A Single-Key Attack on the Full GOST Block Cipher. In: Joux, A. (ed.) Fast Software Encryption - 18th International Workshop, FSE 2011, Lyngby, Denmark, February 13-16, 2011, Revised Selected Papers. Lecture Notes in Computer Science, vol. 6733, pp. 290–305. Springer (2011)
9. Kara, O.: Reflection Cryptanalysis of Some Ciphers. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) Progress in Cryptology - INDOCRYPT 2008, 9th International Conference on Cryptology in India, Kharagpur, India, December 14-17, 2008. Proceedings. Lecture Notes in Computer Science, vol. 5365, pp. 294–307. Springer (2008)
10. Ko, Y., Hong, S., Lee, W., Lee, S., Kang, J.: Related Key Differential Attacks on 27 Rounds of XTEA and Full-Round GOST. In: Roy, B.K., Meier, W. (eds.) Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004, Revised Papers. Lecture Notes in Computer Science, vol. 3017, pp. 299–316. Springer (2004)

11. OpenSSL: A Reference Implementation of GOST, http://www.openssl.org/source/
12. Poschmann, A., Ling, S., Wang, H.: 256 Bit Standardized Crypto for 650 GE - GOST Revisited. In: Mangard, S., Standaert, F. (eds.) Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6225, pp. 219–233. Springer (2010)
13. Russian National Bureau of Standards: Federal Information Processing Standard-Cryptographic Protection - Cryptographic Algorithm. GOST 28147-89 (1989)
14. Seki, H., Kaneko, T.: Differential Cryptanalysis of Reduced Rounds of GOST. In: Stinson and Tavares [15], pp. 315–323
15. Stinson, D.R., Tavares, S.E. (eds.): Selected Areas in Cryptography, 7th Annual International Workshop, SAC 2000, Waterloo, Ontario, Canada, August 14-15, 2000, Proceedings, Lecture Notes in Computer Science, vol. 2012. Springer (2001)
16. US National Institute of Standards and Technology (NIST): Federal Information Processing Standards Publication 46-3 (1999)

A An Impossible Reflection Attack for the Full GOST2

In this section we present an impossible reflection attack on the full GOST2. This attack is a complementary attack to the one presented in Section 4.1 and uses the fact that $K^5 \neq K^6$. When this happens, the event $S_{18} = C$ is impossible, as it implies that $K^6_{24} = K^5_{25}$ (for the bijective S-boxes π_1 and π_2 suggested in [6]).

The attack uses an outer loop and two sequential inner loops. In the outer loop the adversary iterates over all K^5 and $K^6 \neq K^5$. In the first inner loop, she iterates over all possible S_3, and over all pairs (P_i, C_i) of plaintexts and ciphertexts, to find the values K_0, K_1, and K_2 leading from the P_i to S_3. Then, she assumes towards contradiction that $S_{18} = C_i$ and decrypts $S_{18} = C_i$ back to S_{13}. She stores in a temporary table $T1$ the values K^0_0, K^1_1, and K^2_2 indexed by $S_3||S_{18}$.

In the second inner loop the adversary iterates over S_3, K^3_3, K^4_4, and K^7_7, and tries to encrypt S_3 to S_{13}. If $S_3||S_{13}$ is in the table, the keys associated with their entry along with the values of the iterators (i.e., the current K_3, K_4, K_5, K_6, and K_7) are discarded as impossible keys. Finally, the adversary tries all remaining keys to find the right one.

The probability that a key is impossible is e^{-1} and thus, the number of impossible keys is $e^{-1} \cdot 2^{256} \approx 2^{254.56}$. We can reduce the number of possible keys by another factor of 2 by using GOST2's complementing property, leading to a time complexity of $2^{-1} \cdot e^{-1} \cdot 2^{256} \approx 2^{253.56}$. The data complexity is 2^{64} known plaintexts (2^{63} chosen plaintexts when using the complement property), and the memory complexity is $\approx 2^{256}$ (for storing the set of impossible keys). A full description of the attack can be found in Algorithm 3.

Algorithm 3 Pseudocode of the improved impossible reflection attack for GOST2.

```
Input: $2^{64}$ pairs of known plaintexts and ciphertexts.
  {Outer loop}
  for $K_5, K_6$ do
    for $S_3, (P_i, C_i)$ do {Inner loop 1}
      for $K_2$ do
        $S_2 \leftarrow R_{K_2}^{-1}(S_3)$
        $T[L_2] \leftarrow K_2$
      end for
      for $K_0$ do
        $S_1 \leftarrow R_{K_0}(P_i)$
        $K_2 \leftarrow T[R_1]$
        $K_1 \leftarrow \mathrm{SOLVE}(P_i, S_3, K_0, K_2)$
        $S_{18} \leftarrow C_i$
        $S_{13} \leftarrow R_{K_0}^{-1}(R_{K_1}^{-1}(R_{K_2}^{-1}(R_{K_5}^{-1}(R_{K_6}^{-1}(S_{18})))))$
        $T1[S_3||S_{13}] \leftarrow (K_0, K_1, K_2)$
      end for
    end for
    for $S_3, K_3, K_4, K_7$ do {Inner loop 2}
      $S_{13} \leftarrow R_{K_3}(R_{K_4}(R_{K_5}(R_{K_6}(R_{K_7}(R_{K_3}(R_{K_4}(R_{K_5}(R_{K_6}(R_{K_7}(S_3))))))))))$
      $(K_0, K_1, K_2) \leftarrow T1[S_3||S_{13}]$
      Discard$(K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7)$
    end for
  end for
  for all undiscarded keys do
    TRY$(K^0, K^1, K^2, K^3, K^4, K^5, K^6, K^7)$
  end for
```