IET Information Security

Research Article

On the extension and security of key schedule of GOST

ISSN 1751-8709
Received on 11th June 2018
Revised 26th September 2018
Accepted on 12th November 2018
E-First on 29th January 2019
doi: 10.1049/iet-ifs.2018.5291
www.ietdl.org

Yafei Zheng[1,2], *Wenling Wu*[1,2] ✉
[1]*TCA Laboratory, SKLCS, Institute of Software, Chinese Academy of Sciences, Beijing 100190, People's Republic of China*
[2]*University of Chinese Academy of Sciences, Beijing 100049, People's Republic of China*
✉ *E-mail: wwl@tca.iscas.ac.cn*

Abstract: A type of simple key schedule especially suitable for lightweight block ciphers is defined as straightforward key schedule in this study. As a typical example, GOST-type key schedule, which is an extension of the key schedules of Russian Standard GOST and its newly modified version GOST2, is introduced and classified. GOST2 is designed based on the GOST encryption structure with different but the same type of key schedule to overcome the weakness of GOST against self-similarity properties-based attacks. However, it has been shown in Fast Software Encryption 2017, the simple change in the key schedule is insufficient to offer 256-bit security. By constructing an evaluation framework combining self-similarity properties and meet-in-the-middle attack, properties of GOST-type key schedules are evaluated, and candidate key schedules are provided in this work. These candidate key schedules are able to provide much better security for GOST and GOST2 ciphers than their original key schedules, and the pre-existing self-similarity properties-based attacks of full round GOST and GOST2 can be avoided. The designers of GOST and GOST2 should have been more cautious choosing the parameters of key schedules. The evaluation framework proposed can be used for reference in the design of other Feistel ciphers with straightforward key schedules.

1 Introduction

In recent years, with the widespread development and application of source constrained devices, such as sensor networks and smart card, lightweight block ciphers have attracted a lot of interest in cryptography. For better hardware implementation, lightweight block ciphers tend to adopt fairly simple key schedules. For example, key schedules of Present [1], LBlock [2] and Twine [3] adopt iteration functions without good diffusion. Key schedules of HIGHT [4], SIMON and SPECK [5] are simple permutations or linear transformations. What is more, there is no diffusion in the key schedules of LED [6], Printcipher [7], XTEA [8], Piccolo [9], GOST [10] and GOST2 [11] at all. Concretely, divide 80-bit master key into five 16-bit key blocks $k_{i(16)}(0 \le i < 5)$, each subkey of Piccolo-80 is simply the XOR result of a constant with a certain key block. GOST adopts a similar key schedule with Piccolo-80, exclusive of the XOR operation of constants.

A simple key schedule will inevitably lead to trade-off between safety and efficiency, and corresponding block cipher will show weak security against both single key and related key attacks. While in single key attack, the weakening effects are mainly in two aspects: (i) for distinguisher-based attack, simple key schedule helps to reduce involved key bits guessed in the key recovery phase and (ii) for attack based on key space partition, simple key schedule will play a vital role, and even directly determine the security of a block cipher against meet-in-the-middle (MITM) attack. The MITM attack was proposed during the evaluation of data encryption standard (DES) [12], and the core idea is the partition of master key space. Simple key schedules simplify the key partition. As a result, the MITM attack has shown excellent power in the security evaluation of lightweight block ciphers owing to simple key schedules. Improved by some classical techniques, the MITM attack sometimes can obtain full round results of lightweight block ciphers, such as the three-subset-MITM attack of KTANTAN [13].

Reflection property [14] and fixed-point property [15] are two self-similarity properties proposed in 2008 and 2012. The combination of the MITM attack and self-similarity properties has shown its power in the security evaluation of full round lightweight block ciphers PRINCE [16], GOST [14, 15, 17] and GOST2 [18]. The basic idea of this combination is to cover as many rounds as possible in the outer loop with self-similarity properties, and the inner loop will be the basic MITM attack on reduced rounds. The complexity is the product of the two loops. The key schedule plays an important role in self-similarity properties. Long round self-similarity properties barely exist owing to the diffusion of the key schedule. For the key schedules without diffusion, such self-similarity properties are also unusual. Consequently, weak key space is occasionally adopted to strive to evaluation of full round ciphers. For attackers, the core idea is to construct as long round self-similarity properties as possible in as large key space as possible.

Based on the above context, we focus on a type of simple key schedule with the definition ‘straightforward key schedule’. Among which key schedules of GOST and GOST2 are two typical examples. Self-similarity properties and MITM attack are both very sensitive to straightforward key schedule. We propose an evaluation framework based on the combination of the MITM attack and self-similarity properties. Take the evaluation of GOST-type key schedules (extension of the key schedules of GOST and GOST2) for example, the evaluation process is detailed under the premise of equivalent classification. The results lead to the conclusion that a simple change of the key schedule can provide better security for GOST and GOST2 ciphers than their original key schedules, and the pre-existing self-similarity properties-based attacks of full round GOST and GOST2 can be avoided. The parameters of GOST-type key schedule should be carefully chosen. As a bad example, based on the analyses of GOST, the key parameters of GOST2 should have been considered more carefully and comprehensively to avoid similar attacks of GOST. We propose candidate key schedules which are able to offer full key security for GOST-type ciphers, even in the weak key space situation.

The rest of the paper is organised as follows. In Section 2, definitions of straightforward key schedule and GOST-type key schedule are proposed. Furthermore, GOST-type key schedule is classified according to two equivalent properties. Section 3 introduces the evaluation framework. Section 4 details the evaluation process of GOST-type key schedules and provides valuable candidate key schedules. Conclusion is given in Section 5.

2 GOST-type key schedule

In this section, we first define straightforward key schedule, and then present the detailed description and classification of GOST-type key schedule.

2.1 Straightforward key schedule

Definition 1: Let l be the master key size, b be the block size, and s be the subkey size of a block cipher. With appropriate generality, we suppose l can be divided by s in this study. Divide the master key into m key blocks of size s. Straightforward key schedule selects each key block in a certain order as the subkey directly.

Example 1: Key schedule of LED-64/128 divides the 128-bit master key into two key blocks $K = K_1 \parallel K_2$, $m = 2$, and K_1, K_2 are selected iteratively as the step keys.
For a straightforward key schedule, there is no diffusion of the master key at all. In fact, each subkey is equal or independent to other subkeys.

Since Feistel structure shares similar encryption and decryption, we choose Feistel structure as the targeted cipher in this study.

Definition 2: Let l be the master key size, b be the block size of a Feistel structure, and the subkey size is $b/2$. Divide the master key into m key blocks of size $b/2$. Suppose the round number r can be divided by m exactly. Straightforward key schedule of Feistel structure selects each key block in a certain order as the subkey directly in each round Feistel encryption, and the m key blocks are traversed once every m rounds starting from round 0. The m-round encryptions are called a bundle.

Example 2: Key schedules of GOST and GOST2 described by Table 1.

The most commonly used blocksize\keysize parameters of mainstream block ciphers nowadays are: 64\64, 64\128, 64\256, 128\128, 128\192, 128\256. The evaluation framework in this study considers no details of the round function. As a matter of fact, two ciphers with different l and b but same l/b are treated equally in our work. The parameters of the mainstream ciphers can be concluded to 64\64, 64\96, 64\128, and 64\256. According to Definition 2, corresponding m is 2, 3, 4, and 8, respectively. The round number can be divided by 2, 3, 4, and 8, respectively. What needs to be pointed out is the evaluation framework in Section 3 is also suitable for situations dissatisfying the suppositions, but with more complex representation. We leave it to analysis of specific ciphers.

2.2 Definition of GOST-type key schedule

Set $l = 256, b = 64, m = 8, r = 32$. The 256-bit master key is divided into eight 32-bit key blocks $K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7$. According to the definition of straightforward key schedule of Feistel structure in Definition 2, the 32 rounds are composed of four bundles, and eight key blocks $K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7$ are traversed in each bundle.

GOST-type key schedule is a special and typical case of straightforward key schedule. The particularity is embodied in the additional condition that subkey sequence in a bundle adopts a left-rotation of key block sequence $K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7$ or its reversed sequence.

If subkey sequences of the first 24 rounds (the first three bundles) are key block sequence $K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7$ under parameters (0, 0, 0), which means each sequence is rotated to the left by 0 block, and subkey sequence of the last eight rounds (the last bundle) is key block sequence $K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0$ under parameter (0), then the sequence of the 32 key blocks constitutes the subkey sequence of GOST. If subkey sequences of the first three bundles are sequence $K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7$ under parameters (0, 3, 5), which means the subkey sequence of the first three bundles are sequence $K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7$ rotated to the left by 0, 3, 5 key blocks, respectively, and the subkey sequence of the last bundle is sequence $K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0$ under parameter (1), then the sequence of the 32 key blocks constitutes the subkey sequence of GOST2. The specifications of subkey sequences of GOST and GOST2 are presented in Table 1.

Compared to the general definition of straightforward key schedule of Feistel structure, GOST-type key schedule restricts the subkey sequence in a bundle to be rotation of the natural key block sequence or its reversed sequence, instead of sequence under full permutation. This restriction results in a reduced space of key schedules. In the meantime, more ordering key schedules will be retained as candidates. As long as candidate key schedules are able to satisfy the security demand of the designers, the restriction is reasonable and practical. Reduced space of key schedules may be not comprehensive. However, for cipher designers, it is enough to have a certain number of key schedules ensuring security demand.

We give the definition of GOST-type key schedules, which is an extension of the key schedules of GOST and GOST2.

Table 1 GOST and GOST2 key schedules

Round	0	1	2	3	4	5	6	7
GOST	K_0	K_1	K_2	K_3	K_4	K_5	K_6	K_7
GOST2	K_0	K_1	K_2	K_3	K_4	K_5	K_6	K_7

Round	8	9	10	11	12	13	14	15
GOST	K_0	K_1	K_2	K_3	K_4	K_5	K_6	K_7
GOST2	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
GOST	K_0	K_1	K_2	K_3	K_4	K_5	K_6	K_7
GOST2	K_5	K_6	K_7	K_0	K_1	K_2	K_3	K_4

Round	24	25	26	27	28	29	30	31
GOST	K_7	K_6	K_5	K_4	K_3	K_2	K_1	K_0
GOST2	K_6	K_5	K_4	K_3	K_2	K_1	K_0	K_7

Definition 3: A GOST-type key schedule meets the following three specifications:

i. The 256-bit master key is divided into eight key blocks $K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7$.
ii. The 32-round encryption is divided into four bundles, and each eight key blocks are traversed as subkeys in each bundle.
iii. Each set of parameters $(c_1, c_2, c_3, c_4), c_i \in \{0, 1, \dots, 7\}, i = 1, 2, 3, 4$, represents a 32-round subkey sequence, i.e. to say, a key schedule. The subkey sequence of bundle i is $(K_0 \parallel K_1 \parallel \cdots \parallel K_7)_{<<<c_i}$ if bundle i does not adopt a reversal key sequence. Otherwise, the subkey sequence of bundle i is $(K_7 \parallel K_6 \parallel \cdots \parallel K_0)_{<<<c_i}$, and we denote it by c_i^* and call it reversed bundle. $<<< c_i$ stands for left-rotation by c_i key blocks.

Property 1: Two GOST-type key schedules (c_1, c_2, c_3, c_4) and $(c_1^*, c_2^*, c_3^*, c_4^*)$ are equivalent. $((c_i^*)^* = c_i.)$

We define Property 1 as equivalence of reversion. Any two key schedules satisfying (c_1, c_2, c_3, c_4) and $(c_1^*, c_2^*, c_3^*, c_4^*)$ can be exactly the same by simply adding an reversion operation to the initial key block sequence of one of these two key schedules.

As a simple example, GOST-type key schedules $(0, 1, 2, 0^*)$ and $(0^*, 1^*, 2^*, 0)$ are equivalent according to Property 1.

Property 2: Two key schedules (c_1, c_2, c_3, c_4), (c_1', c_2', c_3', c_4') are equivalent when: (i) The positions of reversed bundles are the same. (ii) For the blocks without reversion: $(c_i' - c_i + 8) \bmod 8 = t$, and for the blocks with reversion: $(16 - (c_i' - c_i)) \bmod 8 = t$, where t is a constant number.

We define Property 2 as equivalence of rotation. Two key schedules satisfying Property 2 can be the same by rotating the initial key sequence of one of the two key schedules.

As a simple example, GOST-type key schedules $(0, 1, 2, 0^*)$ and $(2, 3, 4, 6^*)$ are equivalent according to Property 2.

The securities of equivalent key schedules are the same due to the same reflection rounds and fixed-point rounds, and MITM properties. The equivalence classifications of Properties 1 and 2 help to reduce redundant evaluation work.

Based on the number and position of reversed bundles, GOST-type key schedules can be classified into the following five situations:

(1) All four bundles are not reversed bundles.
(2) Only one bundle is reversed bundle. There are two situations to be considered: (c_1, c_2, c_3, c_4^*) (c_1, c_2, c_3^*, c_4). The other two situations (c_1, c_2^*, c_3, c_4) (c_1^*, c_2, c_3, c_4) are omitted owing to the similarity of encryption and decryption of Feistel structure. Note that GOST and GOST2 both adopt situation (c_1, c_2, c_3, c_4^*). The GOST key schedule can be denoted as $(0, 0, 0, 0^*)$, and $(0, 3, 5, 1^*)$ for the key schedule of GOST2.
(3) Two bundles are reversed bundles. There are three situations to be considered: (c_1^*, c_2^*, c_3, c_4) (c_1^*, c_2, c_3^*, c_4) (c_1^*, c_2, c_3, c_4^*). Other three situations are also omitted owing to the similarity of encryption and decryption of Feistel structure or the equivalent property as defined by Property 1 in the following.
(4) The situation with three reversed bundles is equivalent to situation 2, as defined by Property 1.
(5) The situation with four reversed bundles is equivalent to situation 1, as defined by Property 1.

For ease of expression, we denote key schedules belonging to situation 2 as type-1 GOST-type key schedule, and type-2 for the key schedules belonging to situations 1 and 3. We will evaluate the self-similarity properties of the above GOST-type key schedules. As GOST and GOST2 both adopt type-1 key schedule, we will first discuss this most typical situation in details in Section 4.1. For type-2 key schedule, which contains all extended key schedules of GOST and GOST2, we obtained the best candidate key schedules in this study, and this part will be presented in Section 4.2.

3 Evaluation framework

Our research studies exploit self-similarity properties using a general framework, which was proposed by Itai Dinur *et al.* in Fast Software Encryption (FSE) 2012 [15]. The difference is, Itai Dinur *et al.* reduce the general problem of attacking the full 32-round GOST into an attack on 8-round GOST with two known input–output pairs, and our attack reduces the full-round attack into an attack on reduced-round Feistel ciphers, whatever the self-similarity properties lead to. The framework consists of an outer loop, which iterates over the given plaintext–ciphertext pairs, and uses them to bypass as many rounds of encryptions as possible. After that, the inner loop will simply be a MITM attack on a reduced-round cipher, which will give some candidates for the right key. The key candidates can be further verified using some other plaintext–ciphertext pairs.

3.1 Self-similarity property

A brief introduction to self-similarity properties including reflection property and fixed-point property is presented in this section. There are two points to be noted first

i. Both these two properties are feasible to Feistel structured ciphers, and the fixed-point property can be even applied to SP structured and any other structured ciphers. For the sake of generality, we treat Feistel (the structure of GOST-type ciphers) as the underlying structure.
ii. Except these two properties, there may be other self-similarity properties. In this study, we just focus on these two simple but valuable properties.

Reflection property was proposed in 2008 by Orhun Kara *et al.* in the evaluation of several block ciphers [14], among which is the attack of 30-round GOST both in full key space and weak key space of size 2^{224}. In 2012, Itai Dinur *et al.* successfully evaluated full 32-round GOST in full key space by improving the combination of reflection property and MITM attack [15]. In 2017, Tomer Asher *et al.* applied the MITM attack with reflection property to GOST2, which is a subsequent cipher of GOST, and full round attack in weak space of size 2^{224} is obtained [18].

Definition 4: $S = (L, R)$ is an internal state of Feistel structure. If $L = R$, S is called a reflection point and for any subkey sequence $K_i, K_{i+1}, \dots, K_{i+j}$,
$E_{K_i}(E_{K_{i+1}}(\cdots(E_{K_{i+j}}(S))) = E_{K_i}^{-1}(E_{K_{i+1}}^{-1}(\cdots(E_{K_{i+j}}^{-1}(S))))$.

If a subkey sequence $K_i, K_{i+1}, \dots, K_{i+j}$ is followed right by its reversed sequence, then setting S as a reflection point with a probability $2^{-b/2}$ will help to reduce full round attack to $r - 2(j+1)$ round.

The fixed-point property was introduced by Itai Dinur *et al.* in 2012 [15], where it was used to attack full round GOST in full key space. This property was later exploited by Tomer Ashur *et al.* to attack full GOST2 in full key space [18].

Definition 5: If internal state S satisfies $S = E_{K_i}(E_{K_{i+1}}(\cdots(E_{K_{i+j}}(S))))$ for subkey sequence $K_i, K_{i+1}, \dots, K_{i+j}$ used in rounds i to $i+j$, S is called a fixed-point and for two successive subkey sequences $K_i, K_{i+1}, \dots, K_{i+j}$, it holds that

$$S = E_{K_i}(E_{K_{i+1}}(\cdots(E_{K_{i+j}}(S)))) = E_{K_i}(E_{K_{i+1}}(\cdots(E_{K_{i+j}}(E_{K_i}((E_{K_{i+1}}\cdots(E_{K_{i+j}}(S))))))) \tag{1}$$

If there exist two successive subkey sequences $K_i, K_{i+1}, \dots, K_{i+j}$, supposing S is a fixed-point with a probability 2^{-b}, and $2(j+1)$ -round encryptions can be bypassed.

Self-similarity properties are usually adopted to reduce the actual attacked rounds in the security evaluation of full round block ciphers. However, long self-similarity properties hardly exist in a

```
Input: a subkey sequence of $r$ length;
Output: the longest reflection property $ref_{result}$;
1: for $i = 1 \to r-1$ do
2:     for $j = min\{i, r-i\} \to 1$ do
3:         if $checkref(j) == 1$ then
4:             $ref_i = 2 \times j$
5:             break
6:         else
7:             $ref_i = 0$
8:         end if
9:     end for
10: end for
11: $ref_{result} = max\{ref_1, ref_2, \ldots, ref_{r-1}\}$
```

Fig. 1 *Algorithm 1: search the reflection property in full key space*

cipher owing to the key schedule. Straightforward key schedule, together with the introduction of weak key space, will help to construct effective self-similarity properties. Take the rationality into consideration, without special explanation, weak key space in this study stands for weak key space of size 2^{224}, which means there is only one equal assumption between the initial key blocks.

The algorithm searching for the longest reflection property and fixed-point property of a Feistel cipher with certain key schedule is trivial according to definition. The pseudo codes are as follows (Fig. 1 and Fig. 2).

ref_i is the length of the reflection property at position i. checkref(j) is the function checking whether for all $1 \le g \le j$, $\text{sk}_{i-g+1} = \text{sk}_{i+g}$. If yes, the function returns 1, else returns 0. sk_i is the subkey of the ith round (Fig. 1).

fp_i is the length of the fixed-point property at position i. checkfp(j) is the function checking whether for all $1 \le g \le j$, $\text{sk}_{i-g+1} = \text{sk}_{i+(j-g)+1}$. If yes, the function returns 1, else returns 0 (Fig. 2).

While in weak key space, a new variable *flag*, which is initialised to 0, will be added in checkref(j) and checkfp(j). During the checking of equality, once the unequal situation occurs, *flag* will be increased to $\text{flag} + 1$. As long as $\text{flag} = 2$, the current checking for j will be terminated and $\text{ref}_i = 0, \text{fp}_i = 0$.

3.2 MITM attack

The MITM attack can be efficiently applied to block ciphers in which some intermediate encryption variables (bits or combinations of bits) depend only on a subset of key bits from the encryption side and on another subset of key bits from the decryption side. The attacker guesses the relevant key bits from each side independently, and tries only keys in which the values suggested by the intermediate variables match up.

We describe a very basic MITM attack, which may be replaced by improved MITM attacks to increase the attacked rounds or decrease the complexity.

Divide the full key space into three groups A_0, A_1, A_2, where A_1, A_2 contains the subkeys involved in the encryption side and the decryption side, respectively. A_0 stands for the set of common bits in both sides.

i. For each possible key of A_0.
ii. For each possible key of $A_1 \setminus A_0$, encrypt the inputs and obtain the intermediate encryption values after certain rounds. Store the intermediate values in a list, sorted according to the intermediate values, and along with the corresponding value of $A_1 \setminus A_0$.
iii. For each possible key of $A_2 \setminus A_0$, decrypt the corresponding outputs and obtain the same intermediate values after certain rounds. Search the sorted list for matching.
iv. For each match, obtain the corresponding values of $A_1 \setminus A_0$ from the sorted list and together with A_0 to derive the full key. Perform trial encryptions of several plaintexts and return the full key.

```
Input: a subkey sequence of $r$ length;
Output: the longest fixed-point property $fp_{result}$;
1: for $i = 1 \to r-1$ do
2:     for $j = min\{i, r-i\} \to 1$ do
3:         if $checkfp(j) == 1$ then
4:             $fp_i = 2 \times j$
5:             break
6:         else
7:             $fp_i = 0$
8:         end if
9:     end for
10: end for
11: $fp_{result} = max\{fp_1, fp_2, \ldots, fp_{r-1}\}$
```

Fig. 2 *Algorithm 2: search the fixed-point property in full key space*

The guess of A_0 in the outer loop is for the sake of memory complexity. The time complexity can be estimated as $2^{|A_0|} \times \max\{2^{|A_1 \setminus A_0|}, 2^{|A_2 \setminus A_0|}\} + 2^{|A_0| + |A_1 \setminus A_0| + |A_2 \setminus A_0| - v}$, where v is the bit size of the intermediate values. The memory complexity will be $\max\{2^{|A_1 \setminus A_0|}, 2^{|A_2 \setminus A_0|}\}$, and the data complexity will be $\lceil l/b \rceil$.

3.3 Evaluation criterion

The less rounds covered by self-similarity properties in the outer loop, the more rounds need to be covered by the MITM attack in the inner loop, and consequently the security of the cipher will be stronger.

Proposition 1: For Feistel ciphers, like GOST and GOST2, the number of rounds of reflection property not more than two will be meaningless, and the number of rounds of the fixed-point property not more than four will be meaningless under the basic MITM attack.

We define the concept of 'meaningless', because for Feistel ciphers with self-similarity properties satisfying Definition 3, the complexity of self-similarity properties-based MITM attack will show no advantage over the complexity of the basic MITM attack.

Take attacks of Feistel cipher with four-round fixed-point property as an example.

When the outer loop of the whole attack is covered by four-round fixed-point property, the probability of a random plaintext satisfies the fixed-point property is 2^{-b}. Then the complexity of the whole attack is the product of the outer loop complexity 2^b and the inner loop complexity (simple MITM attack, which is denoted as 2^{sb}), which will be 2^{b+sb}. For the basic MITM attack of the cipher, without considering the four special rounds (covered by the fixed-point property) first, the rest rounds of the cipher is divided into two parts, each with key sets K_1 and K_2. The complexity of the simple MITM is 2^{sb}. Then the four subkeys of the four special rounds should be distributed to the two parts. Take the initial structure property into consideration, the worst situation will be two more subkeys (when each two subkeys are independent of K_1 and K_2, respectively) to be guessed in each part, and the complexity will be 2^{b+sb}. Stated thus, the self-similarity property-based attack has no advantage over even the worst situation of the basic MITM without self-similarity property.

Under the above definition of meaningless rounds of self-similarity properties, the meaningless sets will be:

$$\begin{array}{ccc} \{0,0\} & \{0,2\} & \{0,4\} \\ \{2,0\} & \{2,2\} & \{2,4\} \end{array}$$

$\{a, b\}$ means the number of reflection rounds is a and the number of fixed-point rounds is b.

During the evaluation of the self-similarity properties of GOST-type key schedules, we aim at candidates with self-similarity properties satisfying the above meaningless sets. When such a request is impossible, relatively better key schedules with the shortest possible self-similarity properties are provided.

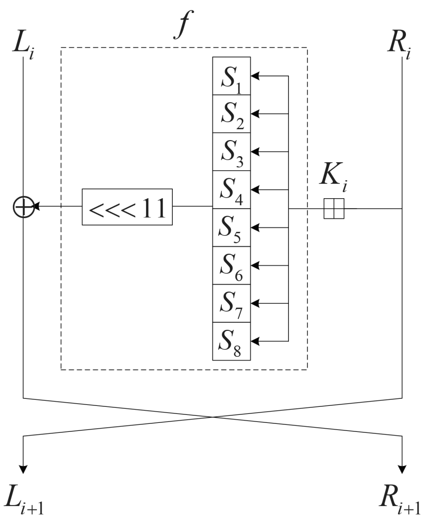

Fig. 3 *One round of GOST*

What needs special explanation is that, key schedules satisfying the above meaningless properties are secure against the basic MITM attacks of GOST-type Feistel ciphers, however, it does not mean key schedules do not satisfy this property will be insecure against the basic MITM attack. As a matter of fact, candidate key schedules later presented in Section 4 with self-similarity properties 'unmeaningless' are also able to provide full key security for GOST-type Feistel ciphers against the basic MITM attack. We use the meaningless definition as a filtering condition just to ensure the 'absolute' security, and to reduce the redundant evaluation work.

For a GOST-type Feistel cipher with a certain key schedule, the rounds of self-similarity properties in full key space will be no more than the rounds of self-similarity properties in weak key space. During the evaluation in Section 4, this property will be used to simplify the evaluation.

4 Evaluation of GOST-type key schedule

GOST is a well-known block cipher proposed as an alternative to the US-developed DES by Soviet Union in the 1970s. The general design of GOST was published in 1994, but some criteria, such as the choice of Sboxes, are secret to the public to date. GOST adopts a 32-round Feistel structure with 64-bit block and 256-bit key. One round encryption consists of adding a 32-bit round key to the right branch and a f function as described in Fig. 3. The Sbox layer consists of eight different 4×4 Sboxes, followed by a rotation by 11 bits to the left. The key schedule of GOST is rather simple, and has been described in Section 2.2.

GOST was accepted as Russian standard after the Soviet Union has been dissolved, and was proposed to be included in ISO-18033. However, the proposition to standardise GOST was rejected. Before 2011, there is no single key attack of full round GOST. The first single key attack of full round GOST, based on the reflection property proposed by Kara *et al.* in 2008 [14], was presented by Takanoti Isobe *et al.* in FSE 2011 [17]. Shortly afterwards, Courtois *et al.* gave a single key attack of full round GOST benefitting from the differential property [19]. In 2012, a new self-similarity property named fixed-point property was proposed by Dinur *et al.*, and attacks on full round GOST were able to be reduced to attacks on eight-round GOST with two known input–output pairs, either use the previous reflection property or the new fixed-point property [15].

In 2015, TC 26 added a new block cipher Kuznyechik [20] to the standard. This cipher was recently suggested to be included in ISO/IEC 18033-3. TC 26 also published GOST2 [11], a modified version of the 64-bit GOST that supposedly resists previous attacks. GOST2 adopts the general structure of GOST, and differs in two aspects: (i) it has a different key schedule, defined to avoid previous single key attacks and (ii) it makes an explicit choice for the Sboxes. However, the proposed fixes of the original key schedule are actually insufficient in resisting previous attacks. In FSE 2017, GOST2 was shown to be vulnerable to the same kind of attack under self-similarity properties as GOST, and it is suspected that a simple change of the key schedule is insufficient to offer 256-bit security for GOST-type ciphers [18].

We apply the evaluation framework in Section 3 to type-1, type-2 GOST-type key schedules defined in Section 2.2. Two equivalent properties are proposed to reduce the searching space. Above all, we draw the conclusion: simple change of GOST-type key schedule can provide full key security for GOST-type ciphers (aim at its pre-existing most popular and effective self-similarity properties-based attacks), based on the premise that the designer being more cautious with the parameters of the key schedule.

In this section, we provide candidate key schedules with meaningless self-similarity properties or relatively better ones. As has been classified in Section 2.2, the evaluation of Type-1 GOST key schedule is first presented in Section 4.1.

4.1 Type-1 GOST key schedule

According to the equivalent properties, Type-1 GOST key schedule includes two situations to be considered: (c_1, c_2, c_3, c_4^*) and (c_1c_2, c_3^*, c_4), among which (c_1, c_2, c_3, c_4^*) is the situation adopted by both GOST and GOST2. Evaluation of situation (c_1, c_2, c_3, c_4^*) is first presented in Section 4.1.1.

4.1.1 (c_1, c_2, c_3, c_4^*)*: Evaluation in full key space:* The search space is constrained as $(c_1, c_2, c_3, c_4^*), c_i \in \{0, 1, \ldots, 7\}, i = 1, 2, 3, 4$. For all 8^4 possible key schedules, we evaluate the numbers of reflection rounds and fixed-point rounds in the full key space firstly, and give candidates based on their securities against the basic MITM attacks.

We search each possible key schedule belonging to situation (c_1, c_2, c_3, c_4^*) for its longest reflection rounds and at the same time fixed-point rounds. Preserve candidates with meaningless self-similarity properties. As the result shows, it can be {2, 2}, {0, 4}, and {2, 4}. For {2, 2} and {0, 4}, there are 56 keys, schedules survive the filtering each, and for {2, 4}, there are 112 keys survive the filtering.

These key schedules satisfying {2, 2} and {0, 4} can be classified into seven candidates, respectively, and key schedules satisfying {2, 4} can be classified into 14 candidates, according to the equivalence defined by Property 2.

(see equation below).

We find no other key schedules satisfying meaningless self-similarity properties.

$$\begin{array}{lllll}
\text{Set 1:} & 1(0,7,6,7^*) & 2(0,7,6,6^*) & 3(0,7,6,5^*) & 4(0,7,6,4^*) \\
 & 5(0,7,6,3^*) & 6(0,7,6,1^*) & 7(0,7,6,0^*). & \\
\text{Set2:} & 8(0,6,4,7^*) & 9(0,6,4,6^*) & 10(0,6,4,5^*) & 11(0,6,4,3^*) \\
 & 12(0,6,4,2^*) & 13(0,6,4,1^*) & 14(0,6,4,0^*). & \\
\text{Set 3:} & 15(0,6,5,7^*) & 16(0,6,5,6^*) & 17(0,6,5,5^*) & 18(0,6,5,4^*) \\
 & 19(0,6,5,2^*) & 20(0,6,5,1^*) & 21(0,6,5,0^*) & \\
 & 22(0,7,5,7^*) & 23(0,7,5,6^*) & 24(0,7,5,5^*) & 25(0,7,5,4^*) \\
 & 26(0,7,5,2^*) & 27(0,7,5,1^*) & 28(0,7,5,0^*). &
\end{array}$$

In the full key space situation, the above 28 candidate key schedules all provide very tough security for GOST-type ciphers against MITM attacks, benefiting from weak self-similarity properties. We can announce that, in the full key space situation, there is no basic MITM attack better than exhaustive search for GOST-type ciphers with key schedule belonging to these 28 candidates.

Further evaluation in weak key space: To go a step further, we evaluate the securities of the above 28 candidate key schedules under weak key space. The results are presented in Table 2. We take key schedule 4 as an example to present its properties.

For a GOST-type cipher with key schedule 4, the subkey sequence of rounds 0–17 is

$$K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_7, K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_6, K_7\,. \quad (2)$$

We assume that the key belongs to a weak key space, where $K_6 = K_7$. If plaintext (S_0) is a fixed-point with respect to rounds 0–8, we get that $S_9 = S_{18} = S_0$. Since the probability of such a fixed-point is 2^{64}, an adversary would obtain such a message on average after encrypting 2^{64} plaintexts (the entire codebook). The 18-round encryptions can be reduced with an outer loop complexity 2^{64} owing to a fixed-point property in weak key space.

The inner loop will be a MITM attack on a 14-round cipher with keys $K_0, K_1, K_2, K_3, K_4, K_5, K_3, K_2, K_1, K_0, K_7, K_6, K_5, K_4$.

For a GOST-type cipher with key schedule 4, the order of keys in rounds 16–29 is $K_6, K_7, K_0, K_1, K_2, K_3, K_4, K_5, K_3, K_2, K_1, K_0, K_7, K_6$. Assume the key belongs to a weak key space where $K_4 = K_5$. Then, if S_{23} is a reflection point ($S^L_{23} = S^R_{23}$), we have $S_{16} = S_{30}$. Since the probability of such a reflection point is 2^{32}, an adversary would obtain a message on average after encrypting 2^{32} plaintexts. The 14-round encryptions can be reduced with an outer loop complexity 2^{32} owing to reflection property in weak key space.

The inner loop will be a MITM attack on an 18-round cipher with keys

$$K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_7, K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_5, K_4\,. \quad (3)$$

For key schedule 7, the weak key space of reflection and fixed-point is the same, both attribute to $K_6 = K_7$, and the two properties can be linked cleverly. This coincidence will lead to a full round distinguishing attack directly with probability 2^{96}. This distinguishing property is impossible owing to the limitation of 64-bit block size. However, we still exclude it from secure candidates.

Taken together, key schedules 8, 10, 11, 13, 14, 15, 16, 18, 22, 23, 25, 26, 28 may provide relatively better security for a GOST-type cipher both in full key space and weak key space. However, we must notify that this evaluation is very limited in several aspects: the selection of self-similarity properties (constrained to reflection and fixed-point properties); the afterwards attack (basic MITM attack); the weak key space (constrained to weak key space of size 2^{224}); the cipher structure considered (Feistel). These candidate key schedules seem to be able to offer relatively better security with no basic MITM attack better than exhaustive search. However, their self-similarity properties do not satisfy the definition of meaningless introduced in Section 3.3 in weak key space.

On the other aspect, the filtering phase of these candidates will shed some light on the design of straightforward key schedules.

Direct exhaustive search in weak key space: In the security analysis of both GOST and GOST2, single key attacks in weak key space have played a very important role. As a natural idea, we try to give candidate key schedules in weak key space directly. The results are 13 key schedules with reflection and fixed-point rounds $\{4, 4\}$:

$$\begin{array}{lllll}(0,6,4,7^*) & (0,6,4,5^*) & (0,6,4,3^*) & (0,6,4,1^*) & (0,6,4,0^*) \\ (0,6,5,7^*) & (0,6,5,6^*) & (0,6,5,4^*) & (0,7,5,7^*) & (0,7,5,6^*) \\ (0,7,5,4^*) & (0,7,5,2^*) & (0,7,5,0^*)\,. & & \end{array}$$

These 13 key schedules are exactly the preserved key schedules of the above further evaluation in weak key space, which are suggested to provide relatively better security for GOST-type ciphers both in the full key space and further in the weak key space.

Numbers of reflection and fixed-point rounds in full key space are surely not more than in weak key space. So, the securities of the above 13 key schedules in full key space must be not weaker than their securities in weak key space. Benefiting from the short self-similarity properties, no basic MITM attacks better than exhaustive search exist.

4.1.2 (c_1, c_2, c_3^*, c_4)*: Evaluation in full key space:* We search for key schedules with the shortest reflection property and fixed-point property under full key space. The search space is constrained as (c_1, c_2, c_3^*, c_4).

Table 2 Evaluations of the 28 key candidates in weak key space

Key candidates	1	2	3	4	5	6	7
reflection rounds	4	4	4	14	4	4	14
fixed-point rounds	18	18	18	18	18	18	18

Key candidates	8	9	10	11	12	13	14
reflection rounds	4	14	4	4	14	4	4
fixed-point rounds	4	4	4	4	4	4	4

Key candidates	15	16	17	18	19	20	21
reflection rounds	4	4	14	4	4	14	4
fixed-point rounds	4	4	4	4	18	18	18

Key candidates	22	23	24	25	26	27	28
reflection rounds	4	4	14	4	4	14	4
fixed-point rounds	4	4	4	4	4	4	4

The results are 392 key schedules with {2, 2}, which can be classified into 49 key schedules because of equivalence of Property 2. We have pointed out the meaningless of reflection rounds not more than two and fixed-point rounds not more than four. The 49 key schedules with {2, 2} thus can be treated as secure ones for GOST-type ciphers against MITM attacks with self-similarity properties.

Further evaluation in weak key space: The 49 key schedules are further evaluated under weak key space, and nine of them satisfy {4, 2}

$$(0,7,6^*,5)\quad (0,7,6^*,6)\quad (0,7,6^*,7)\quad (0,7,5^*,0)\quad (0,7,5^*,6)$$
$$(0,7,5^*,7)\quad (0,7,4^*,0)\quad (0,7,4^*,1)\quad (0,7,4^*,7)$$

No basic MITM attacks for GOST-type ciphers with key schedule belonging to these nine suggestions better than exhaustive search exist.

Direct exhaustive search in weak key space: The same nine key schedules as above are obtained. These nine candidates thus are the relatively better ones under key situation (c_1, c_2, c_3^*, c_4).

Key schedules with two-round reflection property exist, but with fixed-point property at least eight-round at the same time. We choose key schedules with these two properties both meaningless or 'balanced shortest'.

We described the evaluation of type-1 GOST-type key schedule very much in detail in this section. The main purpose is to propose the idea and procedure of the evaluation. Candidate key schedules preserved from this part may not be the best ones in all GOST-type key schedules, but type-1 key schedule is the most familiar and typical form.

4.2 Type-2 GOST key schedule

In this section, type-2 GOST-type key schedule is evaluated. Type-2 key schedule is a natural extension of the original GOST and GOST2 key schedules. Candidate key schedules with the best (shortest) self-similarity properties are preserved from this situation. Owing to exactly the same idea as Section 4.1, candidates in weak key space are directly presented afterwards:

i. (c_1^*, c_2^*, c_3, c_4): There are no key schedules preserved from the filtering with self-similarity properties {2, 2} and {2, 4}.

ii. (c_1^*, c_2, c_3^*, c_4): 23 key schedules with {2, 2} exist. This situation offers the best candidates for GOST-type key schedule

$$(7^*,4,7^*,5)\quad (7^*,4,7^*,6)\quad (7^*,4,1^*,2)\quad (7^*,4,1^*,3)\quad (7^*,4,1^*,4)$$
$$(7^*,4,0^*,3)\quad (7^*,4,0^*,4)\quad (7^*,5,7^*,4)\quad (7^*,5,7^*,6)\quad (7^*,5,6^*,5)$$
$$(7^*,5,6^*,6)\quad (7^*,5,6^*,7)\quad (7^*,5,0^*,3)\quad (7^*,5,0^*,4)\quad (7^*,5,0^*,5)$$
$$(7^*,6,7^*,4)\quad (7^*,6,7^*,5)\quad (7^*,6,6^*,5)\quad (7^*,6,6^*,6)\quad (7^*,6,6^*,7)$$
$$(7^*,6,5^*,0)\quad (7^*,6,5^*,6)\quad (7^*,6,5^*,7).$$

For GOST-type ciphers, two rounds are meaningless both for reflection and fixed-point properties. These 23 key schedules are candidates we finally presented for GOST-type ciphers. GOST-type ciphers can directly adopt one of these 23 suggestions for security against self-similarity properties-based attacks much better than original GOST and GOST2 key schedules. In fact, these key schedules are able to provide full key security for GOST-type ciphers against such attacks. As a conclusion, we make it clear that a simple change of GOST-type key schedule can provide a better security and even full key security against self-similarity properties-based attacks for GOST-type ciphers. For design of other GOST-type ciphers (which can be extended to Feistel block ciphers), a similar evaluation framework can be followed to make the best choice of key schedule.

iii. (c_1^*, c_2, c_3, c_4^*): There are no key schedules with {2, 2} and {2, 4}.

iv. (c_1, c_2, c_3, c_4): There are no key schedules with {2, 2} and {2, 4}.

Owing to weak key, both the numbers of the reflection rounds and the fixed-point rounds would be at least 2. So, meaningless situations considered above are only {2, 2} and {2, 4}.

There are 94 key schedules belonging to (c_1^*, c_2, c_3^*, c_4) satisfy {2, 4}. Since these key schedules are obviously not better than key schedules with {2, 2}, we choose not to list them out here.

5 Conclusion

In this study, we give the definition of straightforward key schedule and GOST-type key schedule is detailed as a typical example. By constructing an evaluation framework combining self-similarity properties and MITM attack, properties of GOST-type key schedules are evaluated, and candidate key schedules are provided in our work. These candidate key schedules are able to provide much better security for GOST and GOST2 ciphers than their original key schedules, and the pre-existing self-similarity properties-based attacks of full round GOST and GOST2 can be avoided. As a matter of fact, self-similarity properties-based MITM attack is the main threat to GOST and GOST2 with their original key schedules up to now. According to this fact, we evaluated the security of such key schedules against MITM attack using the reflection and fixed-point properties, while other attacks, such as the related-key attacks, are not considered.

We come to the conclusion that a simple change of the GOST-type key schedule is sufficient to offer 256-bit security against pre-existing self-similarity property-based attacks for full-round GOST-type ciphers, and designers of GOST and GOST2 should be more careful choosing the key schedule parameters to avoid such a full round attack.

The analysis in this study is very straightforward, and we think this straightforward way can be treated as the advantage of our study. A plain framework is constructed, and candidate key schedules are obtained to improve the present situation of a type of key schedule, which are applied in GOST and GOST2, a Russian Standard and a newly proposed cipher. Only specific attack is concerned in our study, and the generalisations of both the target ciphers and attack methods are exactly the work we are focusing on in the present and future.

6 Acknowledgments

We would like to thank anonymous referees for their helpful comments and suggestions. The research presented in this paper is supported by the National Natural Science Foundation of China (no. 61672509), and the National Cryptography Development Foundation of China (no. MMJJ20170101).

7 References

[1] Bogdanov, A., Knudsen, L.R., Leander, G., *et al.*: 'PRESENT: an ultra-lightweight block cipher'. Proc. Int. Conf. on Cryptographic Hardware and Embedded Systems (CHES), Vienna, Austria, September 2007, pp. 450–466

[2] Wu, W., Zhang, L.: 'LBlock: a lightweight block cipher'. Proc. Int. Conf. on Applied cryptography and network security (ACNS), Nerja, Spain, June 2011, pp. 327–344

[3] Suzaki, T., Minematsu, K., Morioka, S., *et al.*: 'Twine: a lightweight block cipher for multiple platforms'. Proc. Int. Conf. on Selected Areas in Cryptography (SAC), Windsor, ON, Canada, August 2012, pp. 339–354

[4] Hong, D., Sung, J., Hong, S., *et al.*: 'HIGHT: A new block cipher suitable for low-resource device'. Proc. Int. Conf. on Cryptographic Hardware and Embedded Systems (CHES), Yokohama, Japan, October 2006, pp. 46–59

[5] Beaulieu, R., Shors, D., Smith, J., *et al.*: 'The SIMON and SPECK families of lightweight block ciphers'. IACR Cryptology ePrint Archive, June 2013

[6] Guo, J., Peyrin, T., Poschmann, A., *et al.*: 'The LED block cipher'. Proc. Int. Conf. on Cryptographic Hardware and Embedded Systems (CHES), Nara, Japan, September 2011, pp. 326–341

[7] Knudsen, L.R., Leander, G., Poschmann, A., *et al.*: 'Printcipher: a block cipher for IC-printing'. Proc. Int. Conf. on Cryptographic Hardware and Embedded Systems (CHES), Santa Barbara, USA, August 2010, pp. 16–32

[8] Needham, R.M., Wheeler, D.J.: 'Tea extensions'. Computer Laboratory, University of Cambridge, October 1997

[9] Shibutani, K., Isobe, T., Hiwatari, H., *et al.*: 'Piccolo: an ultra-lightweight blockcipher'. Proc. Int. Conf. on Cryptographic Hardware and Embedded Systems (CHES), Nara, Japan, September 2011, pp. 342–357

[10] Russian National Bureau of Standards: 'Federal information processing standard-cryptographic protection-cryptographic algorithm. GOST 28147-89', 1989

[11] Dmukh, A.A., Dygin, D.M., Marshalko, G.B.: 'A lightweight-friendly modification of GOST block cipher', *Technical Committee for Standardization*, 2014, **5**, (2), pp. 47–55

[12] Diffie, W., Hellman, M.E.: 'Special feature exhaustive cryptanalysis of the NBS data encryption standard', *IEEE Computer*, 1977, **10**, (6), pp. 74–84

[13] Bogdanov, A., Rechberger, C.: 'A 3-subset meet-in-the-middle attack: cryptanalysis of the lightweight block cipher KTANTAN'. Proc. Int. Conf. on Selected Areas in Cryptography (SAC), Waterloo, Ontario, Canada, August 2010, pp. 229–240

[14] Kara, O.: 'Reflection cryptanalysis of some ciphers'. Proc. Int. Conf. on INDOCRYPT, Kharagpur, India, December 2008, pp. 294–307

[15] Dinur, I., Dunkelman, O., Shamir, A.: 'Improved attacks on full GOST'. Proc. Int. Conf. on Fast Software Encryption (FSE), Washington, DC, USA, March 2012, pp. 9–28

[16] Soleimany, H., Blondeau, C., Yu, X*., et al.*: 'Reflection cryptanalysis of PRINCE-like ciphers', *J. Cryptol.*, 2015, **28**, (3), pp. 718–744

[17] Isobe, T.: 'A single-key attack on the full GOST block cipher'. Proc. Int. Conf. on Fast Software Encryption (FSE), Lyngby, Denmark, February 2011, pp. 290–305

[18] Ashur, T., Bar-On, A., Dunkelman, O.: 'Cryptanalysis of GOST2', *IACR Trans. Symmetric Cryptol*, 2017, **2017**, (1), pp. 203–214

[19] Courtois, N.T.: 'An improved differential attack on full GOST' in '*The new codebreakers*' (Springer, Berlin, Heidelberg, 2016), pp. 282–303

[20] Federal Agency on Technical Regulation and Metrology: 'National Standard of the Russian Federation. GOST R_34_12_2015', 2015