Cryptanalysis of GOST R Hash Function

Zongyue Wang[a], Hongbo Yu[b,*], Xiaoyun Wang[b]

[a]*Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Jinan 250100, China*
[b]*Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China*

Abstract

GOST R is the hash function standard of Russia. This paper presents some cryptanalytic results on GOST R. Using the rebound attack technique, we achieve collision attacks on the reduced round compression function. Result on up to 9.5 rounds is proposed, the time complexity is 2^{176} and the memory requirement is 2^{128} bytes. Based on the 9.5-round collision result, a limited birthday distinguisher is presented. Moreover, a method to construct k collisions on 512-bit version of GOST R is given which show the weakness of the structure used in GOST R. To the best of our knowledge, these are the first results on GOST R.

Keywords: Hash Function, GOST R, Rebound Attack, Multi-Collision

1. Introduction

Hash functions are taking important roles in cryptography and have been used in many applications, e.g., digital signatures, authentications and message integrity. Since the break of MD5 and SHA-1 [1, 2], cryptographers have been searching for secure and efficient hash functions. Developed from GOST, GOST R is the hash function standard of Russia [3]. Similar as the structure of Whirlpool [4], it also uses an AES-like block cipher in its compression function.

Rebound attack is a freedom degrees utilized technique which can be applied to find collisions in both permutation based and block cipher based hash constructions. This technique is first proposed by Mendel et al. to achieve collision attacks on reduced Whirlpool and Grøstl [5]. It aims to find a pair of values that follows a pre-determined truncated differential efficiently. The searching procedure is divided into two phase: the inbound phase and the outbound phase. In inbound phase the attacker makes full use of the available degrees of freedom and generates sufficiently many paired values that satisfy the truncated differential path of the inbound phase as starting points. The subsequent outbound phase tests these starting points in order to find paired values that satisfy the truncated differential path of the outbound phase.

Giving better results on Whirlpool, Lamberger et al. improved this technique in [6]. Available degrees of freedom of the key schedule are used to extended the inbound phase of the rebound attack by up to two rounds. The best result of [6] is near-collision attack on 9.5 rounds of the compression function with a complexity of 2^{176}. And this result is further turned into the first distinguishing attack for the full 10 round compression function of Whirlpool. At the same time, Gilbert et al. bring in Super-Sbox technique to rebound attack in [7] where two rounds of AES-like permutations were viewed as a layer of Super-Sbox. Besides, the rebound technique can also be applied to analysis AES and AES-like block ciphers [8, 9] as well as ARX ciphers [10]. Recently, using techniques adapted from the rebound attack, Duc et al. constructed differential characteristics on Keccak in [11].

In contrary to finding collisions for hash functions, Joux proposed a method to construct multicollisions in [12]. He argued that for iterated hash functions, to find a multicollisions is not even harder than finding

☆supported by 973 Project (Grant 2013CB834205), NSF of China (No.61133013) and Tsinghua Initiative Scientific Research Program (No.20111080970)
*corresponding author

ordinary collisions. This method can be applied to generally analyze the security of the hash function structure.

1.0.1. Our Contributions

As the similarity between GOST R and Whirlpool, the rebound techniques used in [6] to analyze Whirlpool can also applied to GOST R. However, GOST R replace of the ShiftRows operation in AES-based designs with the matrix transposition. We show that this difference brings more weakness.

In this paper, we present the first analysis on GOST R. More precisely, by applying the rebound attack techniques similar as in [6], we give collision attacks on 4.5, 5.5, 7.5 and 9.5 rounds GOST R compression function respectively. Our collision attacks on GOST R compression function are summarized in Table1. Then we show that the result of 9.5 rounds can be further converted to a 10-round distinguisher. In addition, we give a method to construct multicollisions on full 512-bit version of GOST R. This result shows that the structure used in GOST R is not an ideal one.

Table 1: Summary of results for GOST R compression function. The complexity in brackets refer to modified attacks using a precomputed table taking 2^{128} time/memory to set up

rounds	complexity time/memory	type	source
4.5	$2^{64}/2^{16}$	collision	Sect.3.3
5.5	$2^{64}/2^{64}$	collision	Sect.3.4
7.5	$2^{128}/2^{16}$	collision	Sect.3.5
9.5	$2^{240}/2^{16}(2^{176}/2^{128})$	collision	Sect.3.6

1.0.2. Outline of the Paper

The paper is organized as follows: In section 2, we briefly describe the GOST R hash function. Then we illustrate rebound attack in detail in section 3; a limited birthday distinguisher is given in section 4. In section 5, we present the method of constructing multicollisions. Finally, in section 6, we conclude this paper.

2. The GOST R hash function

GOST R is the Russian hash function standard [3]. It accepts message block size of 512 bits and produces a hash value of 512 or 256 bits. A l-bits message is first padded into a multiple of 512 bits. The bit '1' is appended to the end of the message, followed by $512 - 1 - (l \bmod 512)$ zero bits. Let $M = M_t \parallel M_{t-1} \parallel \cdots \parallel M_1$ be a t-block message (after padding) which is represented in big endian form. As illustrated in Fig.1, the computation of $H(M)$ can be described as follow:

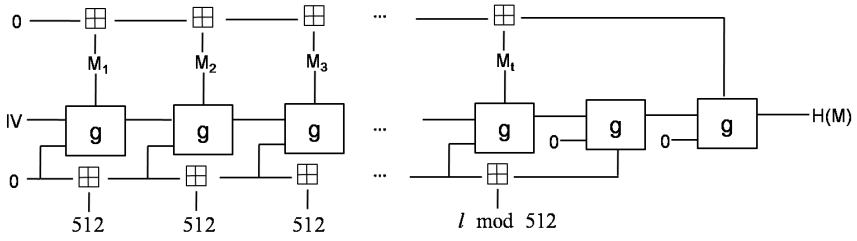

Figure 1: The GOST R hash function

$$h_0 = IV, \quad N = 0, \quad \Sigma = 0 \tag{1}$$

$$h_j = g_N(h_{j-1}, M_j), \quad N = N \boxplus 512, \quad \Sigma = \Sigma \boxplus M_j \qquad for \quad 0 < j < t \tag{2}$$

$$h_t = g_N(h_{t-1}, M_t), \quad N = N \boxplus (l \bmod 512), \quad \Sigma = \Sigma \boxplus M_t \tag{3}$$

$$h_{t+1} = g_0(h_t, N) \tag{4}$$

$$H(M) = g_0(h_{t+1}, \Sigma) \tag{5}$$

where IV is a predefined initial value and '$\boxplus$' means addition operation in the ring $Z_{2^{512}}$. $g_N(h, m)$ is the compress function of GOST R which contains a 512 bits block cipher and is calculated as

$$g_N(h, m) = E(L \circ P \circ S(h \oplus N), m) \oplus h \oplus m \tag{6}$$

The block cipher E used in GOST R is a variant of AES which update an 8×8 state[1] of 64 bytes and round key in 12 rounds. In one round, the state is updated by the round transformation r_i as follow:

$$r_i \equiv X[k_{i+1}] \circ L \circ P \circ S$$

The round transformation is detailed as:

- the non-linear layer S applied an S-Box to each byte of the state independently.
- the byte permutation P transpose the state matrix.
- the linear transformation L is specified by the right multiplication with the 64×64 matrix A over the field $GF(2)$ for each row of the state independently.
- the key addition $X[k_{i+1}]$ XOR the round key k_{i+1} to the state

The round key k_i is update as

$$k_i = L \circ P \circ S(k_{i-1} \oplus C_{i-1}) \qquad for \quad 1 < i \leq 13$$

where C_i are constants in GOST R and k_1 is initialized by $k_1 = L \circ P \circ S(h_{j-1} \oplus N)$.

After the last round of the state update transformation, the output value of the block cipher E, the previous chaining value h_{j-1} and the message M_j are xored together as the output of the compress function.

We denote the resulting state of round transformation r_i by R_{i+1}, and the intermediate states after S, P and L by R_i^S, R_i^P and R_i^L, respectively. The initial state $R_1 = M_1 \oplus k_1$.

3. Rebound Attack on GOST R Compression Function

The rebound attack is a hash function analyzing technique which was first proposed by Mendel et al. in [5] to attack round-reduced Grøstl and Whirlpool. The main idea of this technique is to construct a differential trail using the available degrees of freedom to fulfill the low probability part. Usually, it consists of an inbound phase with a match-in-the-middle part as well as a subsequent probabilistic outbound phase.

With the help of rebound technique, we give collisions on 4.5 and 5.5 rounds of the GOST R compression function. Further more, by using the available degrees of freedom from the key schedule as in [6], we improve the results to collisions on 7.5 and 9.5 rounds.

[1] The state is also 64×64 over the field $GF(2)$

3.1. The Original Rebound Attack

In rebound attack, the block cipher or permutation of a hash function used in the compression function is divided into three sub-parts. Let E be a block cipher, then $E = E_{fw} \circ E_{in} \circ E_{bw}$. The rebound attack works in two phases:

- **Inbound phase:** this phase start with several chosen input/output differences of E_{in} which are propagated through linear layer forward and backward. Then one generate all possible actual value pairs that satisfy the difference by matching the differences though a single Sbox layer. These actual value pairs are the starting points for the outbound phase.
- **Outbound phase:** Compute the matches of inbound phase both forward and backward direction though E_{fw} and E_{bw} to obtain the desire collisions or near-collisions. Usually, E_{fw} and E_{bw} have a low probability so that one has to repeat the inbound phase to obtain more starting points.

3.2. Preliminaries for the Rebound Attack

Before describing the rebound attack on GOST R, we briefly give some properties of its linear transformation L and S-box in the compress function.

- *Difference Propagation in L*: Since L is a linear transformation, a certain input difference of L leads to a certain output difference. As L acts on each row of the state independently, the output difference of a determined row is only affected by the input difference of that row. For only one active byte in a row, it always leads to eight active bytes propagating forward or backward through L, regardless the position of the active byte.
- *Differential Properties of S*: Given $a, b \in \{0,1\}^8$, the number of solutions to the equation

$$S(x) \oplus S(x \oplus a) = b \tag{7}$$

  can only be 0,2,4,6,8 and 256, which occur with frequency 38235, 22454, 4377, 444, 25 and 1, respectively. On average, for a random given differential (a, b), one expects to find one value as a solution. And a 256×256 input/output difference table can help us to find solutions with negligible computation.

3.3. Collision Attack on 4.5-Round GOST R Compression Function

In this section, we will describe the application of rebound attack on the GOST R compression function reduced to 4.5 rounds. The core of the attack is a 4-round differential trail, which has the following sequence of active S-boxes:

$$8 \xrightarrow{r_1} 8 \xrightarrow{r_2} 64 \xrightarrow{r_3} 8 \xrightarrow{r_4} 8$$

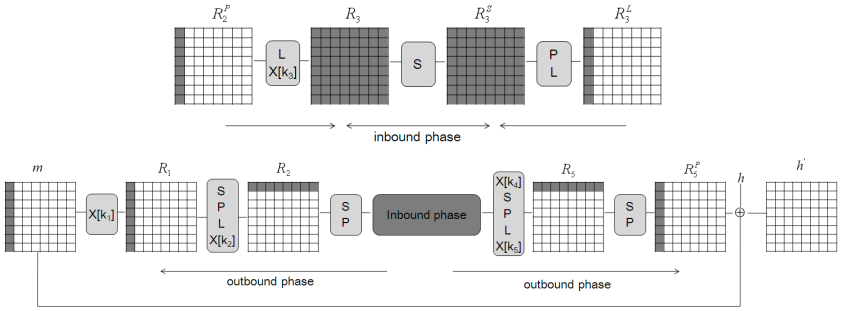

Figure 2: A schematic view of the attack on 4.5 rounds of GOST R compression function. Black state bytes are active.

In the rebound attack, we first split the block cipher E into three sub-ciphers $E = E_{fw} \circ E_{in} \circ E_{bw}$. As shown in Fig.2, the most expensive part of the differential trail is covered in the inbound phase. The available degrees of freedom are used to guarantee the differential trail in E_{in} holds.

3.3.1. Inbound Phase.

In the first step of the inbound phase, we start with 8-byte active differences at both R_2^P and R_3^L, and propagate forward and backward to R_3 and R_3^S respectively (see Fig.2). As the difference propagation property of L described in Sect.3.2, we get a fully active state at both R_3 and R_3^S.

In the second step of the inbound phase, we perform match-in-the-middle on the S layer of r_3 in order to find a matching input/output difference. As indicated in Sect.3.2, we expect to find a solution on average for a given differential trail. Note that there are in total 2^{128} different differential trails, we can get at most 2^{128} actual values for R_3 and R_3^S. As k_3 can be any value, the maximum number of the starting points is $2^{128+512} = 2^{640}$.

3.3.2. Outbound Phase.

In the outbound phase, we use the starting points produced in inbound phase and compute these values backward and forward. As shown in Fig.2, the differences at R_2^P and R_3^L lead differences only in the first column of R_1 and R_5^P respectively.

One can easily construct a collision on the compression function of GOST R reduced to 4.5 rounds using the values generate in above step. As $h' = m \oplus E(k, m) \oplus h$, for the same h and k, the same differences for m and $E(k, m)$ always lead to a collision. For a pair of values generated in the outbound phase, the differences is equal for m and $E(k, m)$ with probability of 2^{-64}. So we have to generate about 2^{64} starting points to construct a collision. The time complexity is about 2^{64}. Since we only need to store a 256×256 input/output differences table for S-box, the memory requirement is only 2^{16} bytes.

3.4. Collision Attack on 5.5-Round GOST R Compression Function

Using the Super-Sbox technique [7], we can improve the 4.5-round result by adding one round to the inbound phase. This turns out to an attack on 5.5-round GOST R compression function. The outbound phase of the attack is same as that of the 4.5-round attack and the new sequence of active S-boxes is:

$$8 \xrightarrow{r_1} 8 \xrightarrow{r_2} 64 \xrightarrow{r_3} 64 \xrightarrow{r_4} 8 \xrightarrow{r_5} 8$$

As shown in Fig.3, the values in every row of R_4^S is only affected by the corresponding column of R_3. In

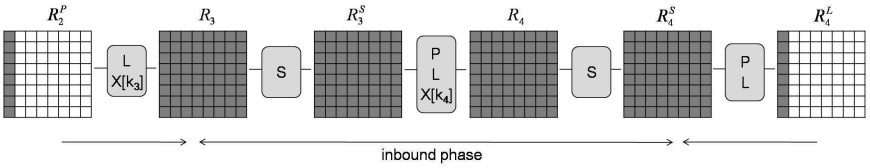

Figure 3: Inbound phase of the attack on 5.5 rounds GOST R compression function.

other words, given a pair of values of a column in R_3, as k_4 is known to us, we could calculate the values of the corresponding row in R_4^S. So we can take the permutation of every column of R_3 to the corresponding row of R_4^S as a Super-Sbox. For every Super-Sbox, randomly given a input and output difference, we suppose to find an actual value on average.

In the following, we give the collision attack for 5.5 rounds in detail:

3.4.1. Inbound Phase.

The inbound phase of rebound attack for 5.5 rounds GOST R is described as follows:

1. Start with a 8-active-bytes difference at the first column R_2^P and propagate forward to R_3.
2. For every independent Super-Sbox, knowing its input difference, go though all the 2^{64} pairs of input value and compute the Super-Sbox forward. This provides 2^{64} output differences values. For every difference reached, store the appropriate pairs of input that led to it. This phase requires about 2^{64} operations and memory.

3. For every 8-byte differences of R_4^L, propagate backward to R_4^S. Check whether for all Super-Sbox, this difference is presented in the storage above.

We can choose different differences at R_2^P to obtain more actual values that satisfy the differences. For a input difference of R_2^P, we suppose to find about 2^{64} starting points.

3.4.2. Outbound Phase.

The outbound phase is the same as the 4.5-round attack where we need 2^{64} starting points. So the time complexity and memory requirement of finding a 5.5 rounds collision are both 2^{64}.

3.5. Collision Attack on 7.5-Round GOST R Compression Function

We can improve the 4.5-round results by adding 3 rounds to the inbound phase by using the degrees of freedom from the key schedule as in [6]. The basic idea is dividing the inbound phase into two subphases. These two subphases can be connected later by making full use of the degrees of freedom in key schedule. As a result, we obtain a collision attack on the compression function of GOST R reduced to 7.5 rounds.

For the improved inbound phase, we use the following sequence of active bytes:

$$8 \xrightarrow{r_1} 64 \xrightarrow{r_2} 8 \xrightarrow{r_3} 8 \xrightarrow{r_4} 64 \xrightarrow{r_5} 8$$

The inbound phase is again split into two subphases in order to find inputs following the differential trail (Fig.4). In the first subphase, we apply match-in-the-middle for rounds 1-2 and 4-5. And in the second subphase, we connect the active values of state between r_2 and r_4 using the degrees of freedom by changing the value of round key.

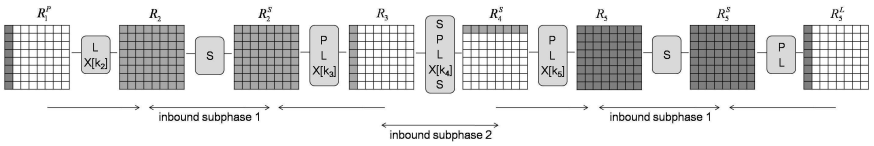

Figure 4: The inbound phase of collision attack on 7.5 Rounds GOST R compression function

3.5.1. Inbound Subphase 1.

In this subphase, we do match-in-the-middle for both rounds 1-2 and 4-5, which can be summarized as follows:

1. Rounds 1-2:
   (a) Start with a 8-active-byte difference at the first column of R_3 and propagate it backward to R_2^S.
   (b) For every differences with 8 active bytes at the first column of R_1^P, propagate them forward to R_2. As there are in total 2^{64} different differences for R_1^P, we could get 2^{64} differences for R_2. After match-in-the-middle, these differences lead to about 2^{64} actual values. However, we could do the match for each row independently and get 2^8 actual values for a row. So the time complexity of generating 2^{64} actual values for which the differential trail holds is only 2^8 round transformations.
2. Rounds 4-5: Do the same as in rounds 1-2.

Now, we get 2^{64} candidates for R_2^S as well as R_5 after the first subphase of the attack with a time complexity of about 2^9 round transformations.

3.5.2. Inbound subphase 2.

In the second subphase, we have to connect the differences of R_2^S and the differences of R_5 as well as the actual values by using the degree of freedom in key schedule. This means we need to solve the following equation:

$$X[k_5]LPSX[k_4]LPSX[k_3]LP(R_2^S) = S_5 \tag{8}$$

with

$$k_4 = LPS(k_3 \oplus C_3) \tag{9}$$

$$k_5 = LPS(k_4 \oplus C_4) \tag{10}$$

For the 2^{64} candidates for R_2^S and 2^{64} candidates for R_5, with the 512 degrees of freedom of k_3, k_4 and k_5, we expect to find 2^{64} solutions. Since $LP(R_2^S) = R_2^L$ and $(X[k_5])^{-1} = X[k_5]$, we can rewrite (8) as follows:

$$LPSX[k_4]LPSX[k_3](R_2^L) = X[k_5]R_5 \tag{11}$$

As we can always change the order of P and S and

$$P^{-1}L^{-1}X[k_5]R_5 = X[P^{-1}L^{-1}(k_5)]P^{-1}L^{-1}(R_5) \tag{12}$$

$$X[P^{-1}L^{-1}(k_5)] = X[S(k_4 \oplus C_4)] \tag{13}$$

we get the following equation:

$$SX[k_4]LSPX[k_3](R_2^L) = X[S(k_4 \oplus C_4)]P^{-1}L^{-1}(R_5) \tag{14}$$

Take $R_2^* = P(R_2^L)$, $k_3^* = P(k_3)$, $k_4^* = S(k_4 \oplus C_4)$ and $T = P^{-1}L^{-1}(S_5)$, the equation above can be rewritten as follows:

$$SX[k_4]LSX[k_3^*](R_2^*) = X[k_4^*](T) \tag{15}$$

Solving the above equation is equivalent to connect the differences and values of R_2^S and R_5. We describe the method used in solving the equation in the following.

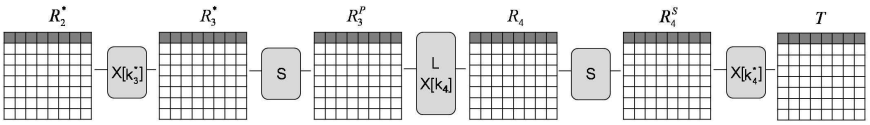

Figure 5: Inbound subphase 2 of collision attack on 7.5 Rounds GOST R compression function

Since the difference of R_2^L and R_4^S is fixed due to inbound subphase 1, the difference of R_2^* and T is also fixed. And the 2^{64} values for state R_2^S and R_5 generated in subphase 1 directly lead to 2^{64} values for R_2^* and T respectively. Now we can solve (15) for every row independently as illustrated in Fig.5, which can be summarized as follows:

1. Compute the 8-byte-difference and the 2^{64} values of R_2^* from R_2^S and compute the 8-byte-difference and corresponding 2^{64} values for T from R_5. We only need to compute and store a row of the values for R_2^* as well as T because we can do the solving row-by-row. The time and memory complexities of this step are both 2^9 instead of the 2^{65}.

2. For all 2^{64} values of the first row of R_3^*, repeat the following step:

   (a) As the difference of R_3^* is known, for the chosen value for the first row of R_3^*, forward compute the values for the first row of R_3^L and the difference for the first row of R_4.

   (b) After doing match-in-the-middle for the difference of the first row of R_4 and R_4^S, we expect to find a solution of the first row of R_4. Since the first row of values of R_3^L is known to us, we can further compute the value of the first row of k_4.

(c) Compute the first row of k_3^*, R_2^*, R_4^S and T. Since for the first row of both R_2^* and R_4^S we have about 2^{64} values, we expect to find a match on both sides. In other words, we have now connected both the values and differences of the first row.

3. In this step, we connect the values of rows 2-8 for the matched R_2^* and T in the above step. For every row independently, exhaustively search over all 2^{64} key values in k_3^* of corresponding row. Note that we want to connect 64 bit values and test 2^{64} key vales, we expect to always find a solution.

After all the steps, we have obtained 2^{64} matches connecting R_2^* and T. In other words, we get 2^{64} starting points for the outbound phase. The time complexity is about 2^{128} round transformation while the memory requirement is about 2^{16} bytes. On average, we expect to find a starting point with time complexity of 2^{64}. Step 3 can be omitted by implying a lookup table of size 2^{128}. With the help of the lookup table, we expect to find a starting point with average complexity of 1. However, the time complexity of building the lookup table is 2^{128} and memory requirement is 2^{128} bytes. Note that there are 2^{64} differences for R_3 and 2^{64} differences for R_4^S. For a fixed pair of differences of R_3 and R_4^S, we expect to find 2^{64} starting points. Thus in total, we could generate at most 2^{192} starting points for outbound phase.

3.5.3. Outbound Phase.

The outbound phase of the 7.5-round attack is the same as that of 4.5 rounds. The collision attack on 7.5 rounds use the following sequence of active bytes:

$$8 \xrightarrow{r_1} 8 \xrightarrow{r_2} 64 \xrightarrow{r_3} 8 \xrightarrow{r_4} 8 \xrightarrow{r_5} 64 \xrightarrow{r_6} 8 \xrightarrow{r_7} 8$$

The time complexity of finding a collision for 7.5 rounds compression function of GOST R is about $2^{64+64} = 2^{128}$ and the memory requirement is 2^{16} bytes.

3.6. Collision Attack on 9.5-Round GOST R Compression Function

Although we can get at most 2^{192} starting points, the outbound phase of 7.5 rounds attack requires only 2^{64} ones. We can extend the outbound phase by adding one round at the beginning and one round at the end to construct a collision attack on 9.5 rounds. This collision attack use the following sequence of active bytes:

$$8 \xrightarrow{r_1} 1 \xrightarrow{r_2} 8 \xrightarrow{r_3} 64 \xrightarrow{r_4} 8 \xrightarrow{r_5} 8 \xrightarrow{r_6} 64 \xrightarrow{r_7} 8 \xrightarrow{r_8} 1 \xrightarrow{r_9} 8$$

In the following, we will describe the outbound phase of the 9.5 rounds collision in detail.

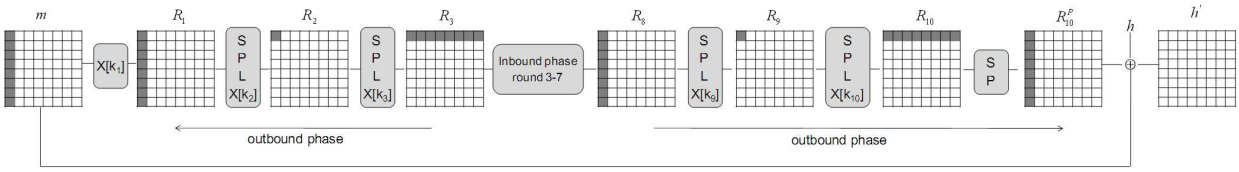

Figure 6: A schematic view of the attack on 9.5 rounds GOST R compression function.

3.6.1. Outbound Phase.

In contrast to the outbound phase for 7.5 rounds, we use truncated differentials here. After the inbound phase, the generated values lead the 8-byte-difference of R_3 and R_8 as shown in Fig.6. For both direction, we want to fulfill a $8 \rightarrow 1$ truncated differential trail. The probability of the truncated differential trail is 2^{-56}. The 1-byte difference of R_2 and R_9 will always result in 8 active bytes for both R_1 and R_{10}. So the probability of outbound phase is 2^{-112}. Hence we have to generate 2^{112} times more starting points.

Since a collision on the compression function reduce to 9.5 rounds require the same differences in m and R_{10}^P, we have to generate in total $2^{112+64} = 2^{176}$ starting points. As described in Sect.3.5, we expect to find a staring points with time complexity of 2^{64}. So the time complexity of finding a 9.5 rounds collision is about $2^{64+176} = 2^{240}$ and the memory requirement is 2^{16} bytes. By using a lookup table with 2^{128} entries, the time complexity is about 2^{176} while the memory requirement is 2^{128} bytes.

4. Limited-Birthday Distinguisher for 10 Rounds

In this section, we present a limited-birthday distinguisher for the compression function of GOST R reduced to 10 rounds.

Produced by Gilbert et.al. in [7], the limited-birthday distinguisher can be summarized as follow: for a random function which perform a b-bit permutation, mapping an input difference from a subspace of size I to an output difference from a subspace of size J requires only $\max\{\sqrt{2^b/J}, 2^b/(I \cdot J)\}$ calls to the function[2].

By applying L and $X[k_{11}]$ to previous result on 9.5 rounds, we extend the differential trail to 10 rounds. Even though R_{11} is fully active in terms of truncated differentials, the differences in R_{11} still belong to a subspace of dimension at most 64. Since the input differences of the compression function is in a subspace of size 2^{64}, the output differences belong to a subspace of 2^{128}. For a random function, we need $2^{512-(64+128)} = 2^{320}$ computations to map this kind of input/output differences. But for the compression function reduced to 10 rounds, the time complexity is only 2^{176} with a 2^{16} requirement of memory or 2^{128} with a 2^{128} requirement of memory. The time complexity required for the compression function is much less than that for a random function. This property can be used to distinguish 10-round GOST R compression function from random function.

5. Multi-collision on GOST R Hash Function

Now, we consider the security of the structure of GOST R hash function. For this kind of structure, we give a method to construct a k-collision. The time complexity is much lower than constructing a k-collision for an ideal structure. In other words, we prove that this kind of structure is not an ideal one.

For an ideal hash function with n-bits output, the time complexity to find a pairwise collision is about $2^{n/2}$ while $2^{n\times(k-1)/k}$ to find a k(multi)-collision. However, basing on the pairwise collisions, Joux proposed a method to construct 2^t-collisions with complexity of only $t \times 2^{n/2}$ for the iterated structure at Crypto'04 [12]. As shown in Fig.7, the attacker first generate t different pairwise collisions $\{(B_1, B_1^*), (B_2, B_2^*), \ldots, (B_t, B_t^*)\}$. Then, the attacker can directly output 2^t-collision of the form $(b_1, b_2, \ldots, b_t)$ where b_i is one of the two blocks B_i and B_i^*.

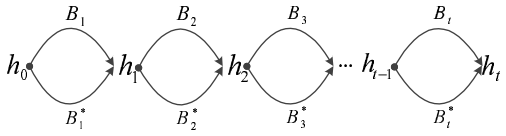

Figure 7: Joux's 2^t-collision construction. The 2^t messages are of the form $(b_1, b_2, \ldots, b_t)$ where b_i is one of the two blocks B_i and B_i^*

Although the structure of GOST R is not an iterated one, we could also construct a k-collision for it. The method is described as follows:

1. As shown in Fig.8, generate 2^t messages which lead to a same value at h_t:
   (a) Let h_0 be equal to the initial value IV of GOST R.
   (b) For i from 1 to t do:
   - Find B_i and B_i^* such that $g_N(h_{i-1}, B_i) = g_N(h_{i-1}, B_i^*)$ where B_i and B_i^* are both of the form $0^{256} \parallel \{0,1\}^{256}$. Since there are in total 2^{256} this kind of messages, we suppose to find a collision using generalized birthday paradox [13].
   - Pad and output 2^t messages of the form $(b_1, \ldots, b_t)$ where b_i is in one of the two blocks B_i and B_i^*.

[2]Without lose of generality, we assume that $I \leq J$

2. From the 2^t messages generated in step 1, try to find k collisions in Σ and output these k messages. Note that all 2^t messages have the same value of N, these k messages always lead to a k-collision of GOST R hash function.

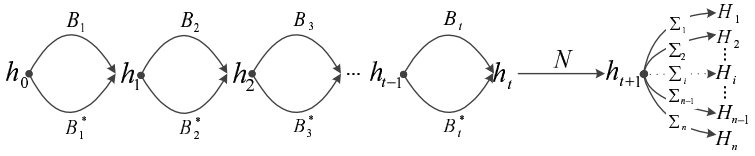

Figure 8: A schematic view of construct k collisions of GOST R.

Since all message blocks in step 1 of the form $0^{256} \parallel \{0,1\}^{256}$ and $\Sigma = b_1 \boxplus b_2 \boxplus \ldots \boxplus b_t$, there are at most $\log_2 t + 256$ significant bits in Σ. Under an ideal model, step 2 needs $2^{(\log_2 t + 256) \times (k-1)/k}$ messages to construct k collisions of Σ which require the following inequality, where $k, t \in Z^+$:

$$t < 2^{256} \tag{16}$$

$$k \geq 3 \tag{17}$$

$$2^{(\log_2^t + 256) \times (k-1)/k} \leq 2^t \tag{18}$$

Solving the above inequality we get

$$176 \leq t \leq 2^{256}$$

and

$$\begin{cases} 3 \leq k \leq (\log_2^t + 256)/(\log_2^t + 256 - t) & , \quad 176 \leq t \leq 264 \\ k \geq 3 & , \quad t \geq 265 \end{cases}$$

In other words, for t-block messages where $176 \leq t \leq 2^{256}$, we can find a k-collision of GOST R hash function with only $2^{(\log_2 t + 256) \times (k-1)/k}$ computations. This time complexity is much less than that of finding a k-collision for an ideal hash function structure.

6. Conclusion

In this paper, we presented some cryptanalytic results of GOST R. By applying rebound attack technique, we first explained our attack on 4.5 rounds of GOST R compression function. This result was further improved to 5.5 rounds by using super-Sbox technique. Then, degrees of freedom in key schedule were used to achieve attacks on 7.5 rounds and 9.5 rounds. More over, using the result of 9.5-round attack, we presented a 10-round distinguisher for the compression function of GOST R. At the end of this paper we presented a method to construct k collisions of GOST R hash function which show the weakness of the structure used in GOST R.

References

[1] X. Wang, H. Yu, How to Break MD5 and Other Hash Functions, in: Advances in Cryptology–EUROCRYPT 2005, Springer, 2005, pp. 19–35.

[2] X. Wang, Y. L. Yin, H. Yu, Finding Collisions in the Full SHA-1, in: Advances in Cryptology–CRYPTO 2005, Springer, 2005, pp. 17–36.

[3] V. Dolmatov, Gost R 34.11-94: Hash function algorithm.

[4] P. Barreto, V. Rijmen, The Whirlpool Hashing Function, in: First open NESSIE Workshop, Leuven, Belgium, Vol. 13, 2000, p. 14.

[5] F. Mendel, C. Rechberger, M. Schläffer, S. S. Thomsen, The Rebound Attack: Cryptanalysis of Reduced Whirlpool and Grøstl, in: Fast Software Encryption, Springer, 2009, pp. 260–276.
[6] M. Lamberger, F. Mendel, C. Rechberger, V. Rijmen, M. Schläffer, Rebound Distinguishers: Results on the Full Whirlpool Compression Function, in: Advances in Cryptology–ASIACRYPT 2009, Springer, 2009, pp. 126–143.
[7] H. Gilbert, T. Peyrin, Super-sbox Cryptanalysis: Improved Attacks for AES-like Permutations, in: Fast Software Encryption, Springer, 2010, pp. 365–383.
[8] O. Dunkelman, N. Keller, A. Shamir, Improved Single-Key Attacks on 8-Round AES-192 and AES-256, in: Advances in Cryptology-ASIACRYPT 2010, Springer, 2010, pp. 158–176.
[9] F. Mendel, T. Peyrin, C. Rechberger, M. Schläffer, Improved Cryptanalysis of the Reduced Grøstl Compression Function, Echo Permutation and AES Block Cipher, in: Selected Areas in Cryptography, Springer, 2009, pp. 16–35.
[10] D. Khovratovich, I. Nikolić, C. Rechberger, Rotational Rebound Attacks on Reduced Skein, in: Advances in Cryptology-ASIACRYPT 2010, Springer, 2010, pp. 1–19.
[11] A. Duc, J. Guo, T. Peyrin, L. Wei, Unaligned Rebound Attack: Application to Keccak, in: Fast Software Encryption, Springer, 2012, pp. 402–421.
[12] A. Joux, Multicollisions in Iterated Hash Functions: Application to Cascaded Constructions, in: Advances in Cryptology–CRYPTO 2004, Springer, 2004, pp. 306–316.
[13] D. Wagner, A Generalized Birthday Problem, in: Advances in Cryptology–CRYPTO 2002, Springer, 2002, pp. 288–304.