Rebound attacks on Stribog

Riham AlTawy, Aleksandar Kircanski and Amr M. Youssef

Concordia Institute for Information Systems Engineering,
Concordia University, Montréal, Quebéc, CANADA

Abstract. In August 2012, the Stribog hash function was selected as the new Russian hash standard (GOST R 34.11-2012). Stribog is an AES-based primitive and is considered as an asymmetric reply to the new SHA-3. In this paper we investigate the collision resistance of the Stribog compression function and its internal cipher. Specifically, we present a message differential path for the internal block cipher that allows us to efficiently obtain a 5-round free-start collision and a 7.75 free-start near collision for the internal cipher with complexities 2^8 and 2^{40}, respectively. Finally, the compression function is analyzed and a 7.75 round semi free-start collision, 8.75 and 9.75 round semi free-start near collisions are presented along with an example for 4.75 round 49 out of 64 bytes near colliding message pair.

Keywords: Cryptanalysis, Hash functions, Meet in the middle, Rebound attack, GOST R 34.11-2012, Stribog

1 Introduction

Wang *et al.* attacks on MD5 [18] and SHA-1 [17] followed by the SHA-3 competition [3] have led to a flurry in the area of hash function cryptanalysis where different design concepts and various attack strategies were introduced. Many of the proposed attacks were not only targeting basic properties but they also studied any non-ideal behaviour of the hash function, compression function, internal cipher, or the used domain extender.

Stribog [1] is proposed by Grebnev *et al.* and has an output length of 512/256-bit. The compression function employs a 12-round AES-like cipher with 8×8-byte internal state preceded with one round of nonlinear whitening of the chaining value. The compression function operates in Miyaguchi-Preneel mode and is plugged in Merkle-Damgård domain extender with a finalization step. Stribog officially replaces the previous standard GOST R 34.11-94 which has been theoretically broken in [11] and recently in [9].

The rebound attack is a differential attack [12] proposed by Mendel *et al.* during the SHA-3 competition to construct differential paths for AES-based hash functions. Previous literature related to the rebound attack includes Mendel *et al.* first proposal on the ISO standard Whirlpool and the SHA-3 finalist Grøstl [12, 13]. In particular, Mendel *et al.* presented a 4.5-round collision, 5.5-round semi free-start collision and 7.5-round near collision attacks on the Whirlpool compression function. As for Grøstl-256, a 6-round semi free-start collision is given.

Subsequently, rebound attacks have been applied to other AES-based hash functions such as LANE [8], JH [14], and Echo [5]. Various tweaks have been applied to the basic rebound attack in order to construct differential paths that cover more rounds such as merging multiple in-bounds [7], super Sbox cryptanalysis [4], extended 5-round inbound [7], and linearized match-in-the-middle and start-from-the-middle techniques [10]. Lastly, Sasaki *et al.* [15] presented a free-start collision and near collision attacks on Whirlpool by inserting difference in the intermediate keys to cancel the difference propagation in the message and thus creating local collisions every 4 rounds.

In this work, we investigate the security of the Stribog hash function, assessing its resistance to rebound attacks. We efficiently produce free-start collision and near collision for the internal cipher (E) reduced to 5 and 7.75 rounds by employing the concept of local collisions. Specifically, we present a message differential path such that a local collision is enforced every 2 rounds. Thus we bypass the complexity of the rebound matching in the message in-bounds by using the same differentials as in the key path. Consequently, in contrast to [15], finding one key satisfying the key path is practically sufficient for finding a message pair following the message path. Finally, we present a practical 4.75 round 49 (out of 64) bytes near colliding message pair for the compression function and show that it is vulnerable to semi free-start 7.75 round collision, 8.75 and 9.75 round near collision attacks. Examples for the internal cipher attacks and the 4.75 round compression function near-collision attack are provided to validate our results.
The rest of the paper is organized as follows. In the next section, the specification of the Stribog hash function along with the notation used throughout the paper are provided. A brief overview of the rebound attack is given in Section 3. Afterwards, in Sections 4 and 5, we provide detailed description of our attacks, differential patterns, and the complexities of the attacks. Finally, the paper is concluded in Section 6.

2 Specification of Stribog

Stribog outputs a 512 or 256-bit hash value and can process up to 2^{512}-bit message. The compression function iterates over 12 rounds of an AES-like cipher with an 8×8 byte internal state and a final round of key mixing. The compression function operates in Miyaguchi-Preneel mode and is plugged in Merkle-Damgård domain extender with a finalization step. The input message M is padded into a multiple of 512 bits by appending one followed by zeros. Given $M = m_n \| .. \| m_1 \| m_0$, the compression function g_N is fed with three inputs: the chaining value h_{i-1}, a message block m_{i-1}, and the block size counter $N_{i-1} = 512 \times i$. (see Figure 1). Let h_i be a 512-bit chaining variable. The first state is loaded with the initial value

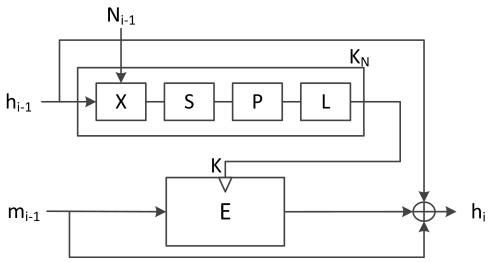

Fig. 1. Stribog's compression function g_N

IV and assigned to h_0. The hash value of M is computed as follows:

$$h_i \leftarrow g_N(h_{i-1}, m_{i-1}, N_{i-1}) \text{ for } i = 1, 2, .., n+1$$
$$h_{n+2} \leftarrow g_0(h_{n+1}, |M|, 0)$$
$$h(M) \leftarrow g_0(h_{n+2}, \sum(m_0, .., m_n), 0),$$

where $h(M)$ is the hash value of M. As depicted in Figure 1, the compression function g_N consists of:

- K_N: a nonlinear whitening round of the chaining value. It takes a 512-bit chaining variable h_{i-1} and the block size counter N_{i-1} and outputs a 512-bit key K.
- E: an AES-based cipher that iterates over the message for 12 rounds in addition to a finalization key mixing round. The cipher E takes a 512-bit key K and a 512-bit message block m as a plaintext. As shown in Figure 2, it consists of two similar parallel flows for the state update and the key scheduling.

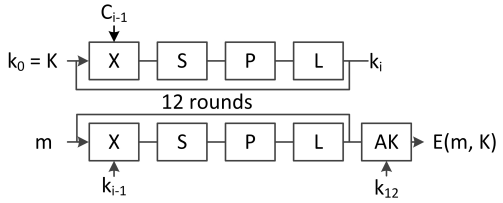

Fig. 2. The internal block cipher (E)

Both K_N and E operate on an 8×8 byte key state K. E updates an additional 8×8 byte message state M. In one round, a state is updated by the following sequence of transformations

- AddKey(X): XOR with either a round key, a constant, or a block size counter (N)
- SubBytes (S): A nonlinear byte bijective mapping.

– Transposition (P): Byte permutation.
– MixColumns (L): Left multiplication by an MDS matrix in GF(2).

Initially, state K is loaded with the chaining value h_{i-1} and updated by K_N as follows:

$$k_0 = L \circ P \circ S \circ X(N_{i-1})$$

Now K contains the key k_0 to be used by the cipher E. The message state M is initially loaded with the message block m and $E(k_0, m)$ runs the key scheduling function on state K to generate 12 round keys $k_1, k_2, .., k_{12}$ as follows:

$$k_i = L \circ P \circ S \circ X(C_{i-1}), \text{ for } i = 1, 2, .., 12,$$

where C_{i-1} is the i^{th} round constant. The state M is updated as follows:

$$M_i = L \circ P \circ S \circ X(k_{i-1}), \text{ for } i = 1, 2, ..., 12.$$

The final round output is given by $E(k_0, m) = M_{12} \oplus k_{12}$. The output of g_N in the Miyaguchi-Preneel mode is $E(K_N(h_{i-1}, N_{i-1}), m_{i-1}) \oplus m_{i-1} \oplus h_{i-1}$ as shown in Figure 1. For further details, the reader is referred to [1].

2.1 Notation

Let M and K be (8×8)-byte states denoting the message and key state, respectively. The following notation will be used throughout the paper:

– M_i: The message state at the beginning of round i.
– M_i^U: The message state after the U transformation at round i, where $U \in \{X, S, P, L\}$.
– $M_i[r, c]$: A byte at row r and column c of state M_i.
– $M_i[\text{row } r]$: Eight bytes located at row r of M_i state.
– $M_i[\text{col } c]$: Eight bytes located at column c of M_i state.
– $m \xrightarrow{r_i} n$: A transition from an m active bytes state at round i to an n active bytes state at round $i + 1$.
– $m \xleftarrow{r_i} n$: A transition from an n active bytes state at round $i + 1$ to an m active bytes state at round i.

Same notation applies to K.

3 The rebound attack

The rebound attack [12] is proposed by Mendel *et al.* for the cryptanalysis of AES-based hash functions. It is a differential attack that follows the inside-out or start from the middle approach which is used in the boomerang attack [16]. The

rebound attack is composed of three phases, one inbound and two outbounds. The compression function, internal block cipher or permutation of a hash function is divided into three parts. If C is a block cipher, then C is expressed as $C = C_{fw} \circ C_{in} \circ C_{bw}$. The middle part is the inbound phase and the forward and backward parts are the two outbound phases. In the inbound phase, a low probability XOR differential path is used and all possible degrees of freedom are used to satisfy the inbound path. In the two outbound phases, high probability truncated paths [6] are used. In other words, one starts from the middle satisfying C_{in}, then hash forward and backward to satisfy C_{fw} and C_{bw} probabilistically. For an 8×8 byte state, the basic rebound attack finds two states satisfying an inbound phase over two rounds $8 \xrightarrow{r_i} 64 \xrightarrow{r_{i+1}} 8$. The main idea is to pick random differences at each of the two eight active bytes sates. Then propagate both backward and forward until the output and input of the full active state Sbox, respectively. Using the Sbox differential distribution table (DDT), find values that satisfy input and output differentials. This process is further illustrated in Figure 3. The last step of the attack is called the Sbox matching phase and its complexity depends on the Sbox DDT. If the probability of differentials that have solutions is p, then the matching probability is given by p^8. In the following, we analyze the Sbox used in Stribog

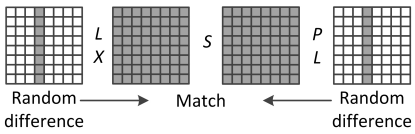

Fig. 3. The rebound attack.

and investigate how it affects the complexity of the rebound attack. The Stribog Sbox DDT has the following properties:

- Out of the 65536 differentials, there are 27300 possible non trivial differentials, i.e., nonzero (input, output) difference pairs that have solutions. Thus the probability that a randomly chosen differential is possible $\approx 0.42 = 2^{-1.3}$
- Each possible differential can have 2, 4, 6, or 8 solutions.
- A given input difference has a minimum of 98 and a maximum of 114 output differences.
- A given output difference has a minimum of 90 and a maximum of 128 input differences.
- For a given input (output) difference the average number of output (input) difference is 107.

From the analysis of the Sbox DDT, one can estimate the complexity of the inbound matching part of the rebound attack. Let us consider the basic inbound path $8 \xrightarrow{r_1} 64 \xrightarrow{r_2} 8$. One can find a pair of states satisfying this path as follows:

1. Compute the Sbox DDT.
2. Choose a random 8 differences for M_2^L active bytes.
3. Propagate the differences in M_2^L backwards until M_2^S (output difference).
4. for each row in M_1^P
   a. Choose a random difference for one active byte, propagate it forward to M_2^X (input difference). Propagating one active byte in M_1^P through the L transformation results in full active row in M_2^X.
   b. Using the Sbox DDT, determine if the corresponding row differences in M_2^X and M_2^S have solutions. If one byte differential pair is not possible, go to step 4.a.

One can repeat step (4.a) at most 2^8 times since we variate only one byte. However, the success probability of step 4.b. (finding solutions for the whole active row) is $2^{-1.3\times 8} \approx 2^{-10}$ which cannot be easily satisfied by randomizing one byte difference. One would often have to restart at step 2, i.e., pick another output difference. The same situation takes place when we move to the next row and pick a new output difference. In this case we have to start from row 0. As a result, the complexity of finding solutions to the 8 rows is not purely added [12]. Based on our experimental results, the complexity of this inbound path is in the order of 2^{18}. However finding this match means finding at least 2^{64} actual state values for M_2^X, such that both M_2^X and $M_2^X \oplus$ (input difference) follow the inbound path. Each value out of the 2^{64} values is a new starting point to satisfy the two outbound paths. In the following section, we present our attack on the internal block cipher of the Stribog compression function.

4 Attacks on the internal block cipher (E)

Verifying the ideal behaviour of the internal primitives of a hash function is important to evaluate its resistance to distinguishing attacks [2]. In this section we investigate the internal block cipher (E) and, by employing the idea of successive local collisions, we present a message differential path that collides every two rounds. This message differential path enables us to efficiently produce 5-round semi free-start collision and 7.75-round 40 bytes (out of 64) semi free-start near collision. The main idea of our approach is to first find a pair of keys that follows a given differential path and then use it to search for a pair of messages satisfying the message path. The approach of creating local collisions works perfectly if the key and the message flows are identical and the initial key is the input chaining value. To this end, one can keep similar differential patterns and the state message difference is cancelled after the X transformation. However, in the compression function of Stribog the key used in the internal cipher is the result of applying the K_N transformation on the input chaining value. Similar differential patterns

can be obtained when considering the internal block cipher. In our attack on the Stribog internal cipher, we present a message differential path such that a local collision is enforced every two rounds. Specifically, we first search for a pair of keys that satisfies the key differential path, then we use the Sbox differentials in the key path for the message path. Consequently, we bypass the complexity caused by the Sbox DDT matching in the message differential path and only one key pair is required to search for a message pair. In [15], Sasaki *et al.* presented a message differential path that creates local collisions every four rounds for the Whirlpool compression function and reported that they had to try 109 key pairs to search for a message pair that collides every 4 round. Furthermore, they estimated an increase in the message search complexity by a factor of 2^7 and attributed this to the imbalance of the Sbox DDT. Given the Stribog Sbox DDT, finding one key pair that follows the 8-round differential path takes up to two hours on an 8-core Intel i7 CPU running at 2.67GHz. Accordingly, it is important that the message differential path requires only one key pair to be satisfied. In what follows, we give the details of our approach.

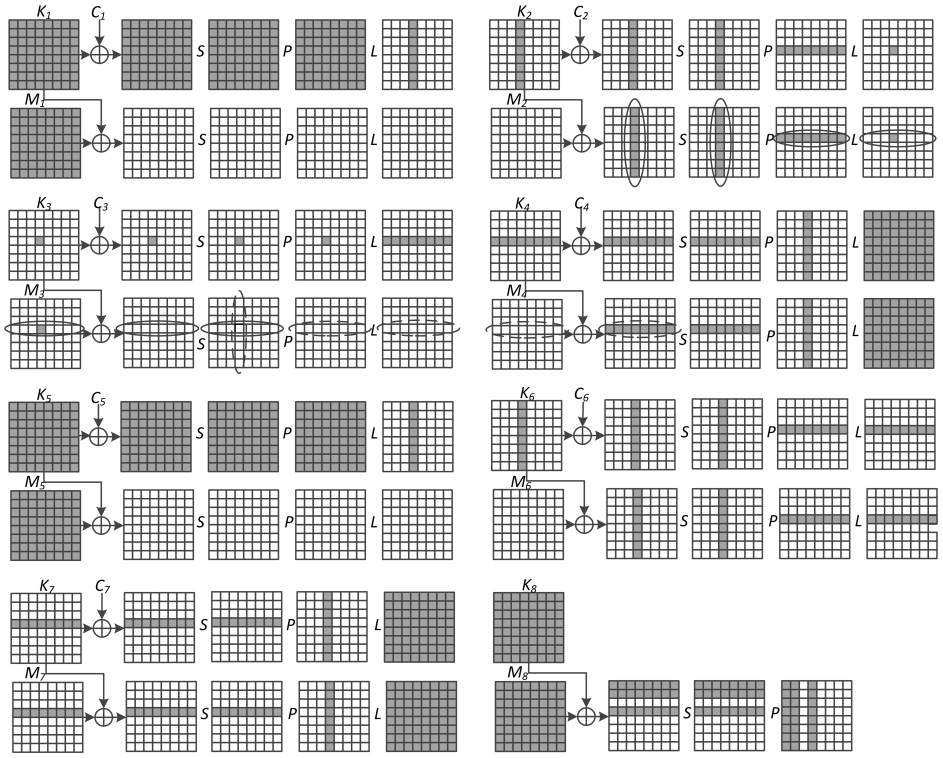

Fig. 4. 7.75 round differential path. Active bytes are coloured grey. Ellipses mark the row and column restricted by the two inbounds.

4.1 5-round free-start collision

Since the Stribog's Sbox DDT is biased with possible differential probability ≈ 0.42, we bypass the Sbox matching phase by using a message differential path such that local collisions are created every two rounds. The used key and message paths are given by:

$$\text{Key: } 64 \xrightarrow{r_1} 8 \xrightarrow{r_2} 1 \xrightarrow{r_3} 8 \xrightarrow{r_4} 64$$
$$\text{Message: } 64 \xrightarrow{r_1} 0 \xrightarrow{r_2} 1 \xrightarrow{r_3} 0 \xrightarrow{r_4} 64 \xrightarrow{r_5} 0$$

This message differential path allows us to bypass the rebound matching part completely in our message search because the same input and output Sbox differences in the key path are used for the message path. Thus the matching probability is 1. Unlike the differential paths in [15], our message differential path is satisfied practically using only one key pair. In this attack, we do not use the matching part of the rebound attack in either the key or the message; we only search for one byte value in the message to find a common solution between two rounds which can be considered as a meet in the middle approach. As depicted in Figure 4, the steps for finding a key pair can be summarized as follows:

1. Choose a random difference and a random value for byte $K_2^L[3, 3]$
2. Hash backward until K_1.
3. Hash forward until K_5.

Accordingly, we have a key pair following the given key path. Let the differences in M_2^X, M_2^S, M_4^X, and M_4^S be the same as the differences in K_2^X, K_2^S, K_4^X, and K_4^S, respectively. Having the same differences in the message states as in the key states implies that no differential matching is needed at the Sboxes of rounds 2 and 4, and guarantees that the differences in K_3 and M_3 cancel out. Similar observation applies to K_5 and M_5.

To search for a conforming message pair, we need to find a common solution between the Sboxes of rounds 2 and 4 possible solutions. This can be achieved as follows. Since M_2^X[col 3] and M_2^S[col 3] differentials are possible, then from the Sbox DDT there are at least 2^8 values for M_2^X[col 3] that satisfy the path until M_3^S. For all solution M_2^X[col 3], hash forward until M_3^S. Because M_2^X[col 3] is one column after the P, L, X, and S transformations, its transformed value becomes M_3^S[row 3] as indicated by the ellipse in Figure 4. We store all possible values of M_3^S[row 3] in a list L. As for M_4^X[row 3], and M_4^S[row 3], hashing all possible solutions backwards restricts the values of M_3^S[col 3]. However we do not store the results in a another list. Because the two restricted results intersect in only one byte $M_3^S[3, 3]$ (the intersection of the two ellipses in Figure 4), we compare byte [3, 3] of each backward result against byte [3, 3] from each entry in list L. The success probability for finding a one byte match is 2^{-8} which can be easily fulfilled

by the number of entries in L. Once a match is found, we assign the matching list row to M_3^S[row 3] and the backwards column to M_3^S[col 3]. The rest of the 49 unrestricted bytes are free and can be used to satisfy a longer outbound.

4.2 8-round collision and 7.75-round near collision attacks

Extending the 5 round path to 8 rounds adds complexity to the key search part because we need to use an improved version of the rebound attack to get a key pair following a longer differential path. We employ the following message and key differential paths:

$$\text{Key: } 64 \xrightarrow{r_1} 8 \xrightarrow{r_2} 1 \xrightarrow{r_3} 8 \xrightarrow{r_4} 64 \xrightarrow{r_5} 8 \xrightarrow{r_6} 8 \xrightarrow{r_7} 64$$
$$\text{Message: } 64 \xrightarrow{r_1} 0 \xrightarrow{r_2} 1 \xrightarrow{r_3} 0 \xrightarrow{r_4} 64 \xrightarrow{r_5} 0 \xrightarrow{r_6} 8 \xrightarrow{r_7} 8 \xrightarrow{r_8} 0$$

and use the start form the middle technique [10] to solve the key inbound phase between rounds 3 and 5. This approach finds states following a $1 \longrightarrow 8 \longrightarrow 64 \longrightarrow 8$ transition. Unlike the basic inbound that yields 2^{64} solutions, using this approach on Stribog results in only one solution. For AES Sboxes, a solution is expected in a time complexity of 2^8 and memory complexity of 2^8. However, for Stribog's biased Sbox DDT, one practical solution is found between 33 minutes to 2 hours on an 8-core Intel i7 CPU running at 2.67GHz. Accordingly, it is crucial that the key outbound phase has high probability if one is aiming for practical results and no rebound matching is used in the message search so that one key is enough to get a conforming message pair. In the following steps, we briefly describe the procedure we used for solving the $1 \longrightarrow 8 \longrightarrow 64 \longrightarrow 8$ key inbound phase. Figure 5 further illustrates the process.

1. Solve the basic inbound $8 \longrightarrow 64 \longleftarrow 8$ as explained in Section 3.
2. From the DDT, each byte difference in K_5^X has at least 2 and at most 8 values, such that any value satisfies the path from K_4^X to K_6.
3. To enforce the transition from 8 active bytes in K_4^X to 1 active byte in K_3^P, do the following:
   a. Create a table T_L of all possible 255 byte difference values d_3 (candidates for K_3^P[3, 3]) and their corresponding 8 byte difference values $L(d_3)$ (candidates for K_4^X[row 3]). These values are the result of applying the linear transformation L to a difference at column 3.
   b. Each candidate difference for $K_4^X[row\ 3]$ has 8 active bytes that can be manipulated independently. More precisely, to change the difference value of byte i in K_4^X[row 3], one has to switch between 2^8 or more possible values of K_5^X[row i]. As illustrated by the ellipses in Figure 5, a change in the values of K_5^X[row 0] is reflected on the difference value of byte 0 in K_4^X[row 3]

c. Go through the entries in table T_L and change the values of K_5^X rows one by one until a match is found, if not, restart from step 1.

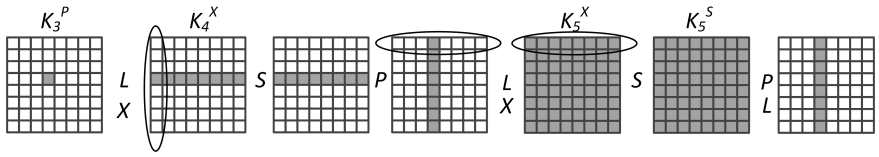

Fig. 5. Start from the middle approach.

Finally, by hashing the obtained key pair two rounds backward and two rounds forward, we get a conforming key pair that follows the key differential path. Once we have the key, we can directly get a message pair in the same way as explained in the previous section for the 5-round collision. This message pair satisfies the message differential path up until M_6^L. However, to have an 8-round collision, we need the difference in K_8 to cancel the difference in M_8 after the X transformation in round 8. Since both L and P transformations are linear, then this condition is satisfied if the 8 byte differences in K_7^S and M_7^S are equal. The difference in K_7^S is already set from the key search stage, so we randomize the 49 unrestricted bytes in M_3^S, hash forward till M_7^S and compare the resulting 8 differences with K_7^S. The probability that the 8 byte differences are equal is 2^{-64}. To verify the applicability of this attack, we have implemented a 7.75-round near collision attack where we were checking if only 5 out of 8 byte differences are equal in M_7^S and K_7^S. In Figure 4, the implemented 7.75-round differential pattern, with 2^{40} time and 2^8 state memory complexities is given. Table 2 shows an example for a free-start 5-round collision and 7.75-round near collision for the internal cipher (E). Both the 5-round semi free-start collision and the 7.75 semi free-start near collision are demonstrated by one example because the 7.75 semi free-start near collision path collides at round 5.

5 Attacks on Stribog compression function

As depicted in Figure 1, the compression function of Stribog employs a nonlinear whitening round K_N of the chaining value. This extra round randomizes the chaining value before being introduced as a key for the block cipher E. As long as there is no difference in the chaining value, most of the differential trails proposed for Whirlpool are also applicable on the Stribog compression function.

In what follows, we consider semi free-start collision attacks on the compression function. Several approaches are used to extend the inbound phase can be used

to construct collision paths for the compression function. The extended 5 round inbound presented in [7] finds a pair of states satisfying the $8 \xrightarrow{r_1} 64 \xrightarrow{r_2} 8 \xrightarrow{r_3} 8 \xrightarrow{r_4} 64 \xrightarrow{r_5} 8$ transition in 2^{64} time and 2^8 memory. The main idea is to solve two independent $8 \xrightarrow{r_1} 64 \xrightarrow{r_2} 8$ and $8 \xrightarrow{r_4} 64 \xrightarrow{r_5} 8$ inbounds and use the freedom to choose key values that connect the resulting 8 differences and 64 byte values. However, unlike the basic inbound, it provides only one solution or starting point for the outbound paths. Using different outbounds with the extended inbound, a semi free-start 7.75-round collision, and 7.75-round, 8.75-round, and 9.75-round near collisions are obtained.

7.75 round semi free-start collision. This is obtained by using two outbounds in the form of $8 \longrightarrow 1$. The probability of a transition from 8 active bytes to 1 active byte through L is $2^{-8\times 7} = 2^{-56}$. Given the following path:

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

one can produce a semi free-start collision. We need two transitions from 8 to 1 in both the forward and backward directions, and the one active byte in the first and last states to be equal so that they cancel out after the feedforward. Thus, one needs to try $2^{56+56+8}$ times to satisfy the outbound phase. In other words, we need 2^{120} inbound solutions. If the complexity of one inbound solution is 2^{64}, then the time complexity of 7.75 rounds semi free-start collision is 2^{184} and the memory complexity is 2^8, as we can pass one active byte through X, S and P transformations with probability one.

7.75 round semi free-start near collision. While aiming for collision requires both differences in the first and last states to be exactly in the same place so that they cancel out after the feedforward, near collision requires only few differences to cancel out. A 49-byte near collision is obtained by extending the 5-round inbound with two transitions from 8 to 8 in both directions with no additional cost. Using the following path:

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

one active byte would cancel out with probability 2^{-8} after feedforward. Consequently, The complexity of 7.75 rounds semi free-start 49-byte collision is 2^{72}. To demonstrate the correctness of the above concept, we have implemented a 4.75-round 49-byte near collision with a shorter practical inbound $8 \xrightarrow{r_2} 64 \xrightarrow{r_3} 8$ as shown in Figure 6. A 4.75-round near colliding pair is given in Table 1 using the $IV = 0$ and $N = 0$.

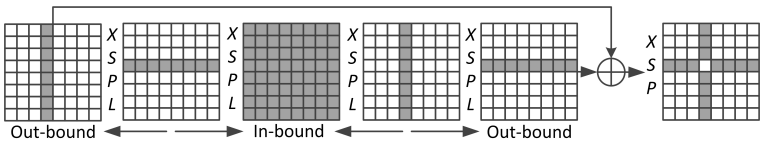

Fig. 6. 4.75 round near collision path

8.75 round semi free-start near collision. Using one transition from 8 to 1 in the forward outbound has a complexity of 2^{56} and results in the following path:

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

The probability that one active byte is cancelled by the feedforward is 2^{-8}. Consequently the complexity of 8.75 rounds semi free-start 49-byte collision is $2^{64+56+8} = 2^{120}$.

9.75 round semi free-start near collision. With a complexity of 2^{196}, a 9.75-round 49-byte near collision can be obtained with a lower complexity of 2^{184}. By adding two 8 to 1 transitions in both the forward and the backward directions for a complexity of 2^{112} and two 1 to 8 transitions in rounds one (backward) and nine (forward) for no additional cost, the following path:

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

results in a 49-byte near collision. Additional complexity of 2^8 is needed for a one byte cancellation after the feedforward.

6 Conclusion

In this paper, we have analyzed the Stribog compression function and its internal cipher. As for the internal cipher, we have proposed a new message differential path such that a local collision is enforced every two rounds. Accordingly, the Sbox matching complexity caused by its DDT bias is bypassed. As a result, we have efficiently produced free-start 5-round collision and 7.75-round near collision examples for the internal cipher. Moreover, the compression function is investigated and we have noted that the Stribog compression function key whitening round K_N enhances its resistance to free-start collision attacks. However, we have showed that the Stribog compression function is vulnerable to semi free-start 7.75 round collision, 8.75 and 9.75 round near collision attacks and presented an example for a 4.75 round 49-byte near colliding message pair.

m	m'	Difference at M_4
cd ed 17 46 d8 d7 f0 f3	cd ed 17 59 d8 d7 f0 f3	00 00 00 1f 00 00 00 00
3e d6 22 7a 99 4a c9 ea	3e d6 22 0c 99 4a c9 ea	00 00 00 76 00 00 00 00
cc 5d e2 f0 14 4f f0 3c	cc 5d e2 ea 14 4f f0 3c	00 00 00 1a 00 00 00 00
4b bc 31 41 dd 99 68 0d	4b bc 31 4d dd 99 68 0d	ba 38 7a 00 6f 93 95 37
b4 d1 27 0f 2d ed 55 28	b4 d1 27 58 2d ed 55 28	00 00 00 57 00 00 00 00
d8 ca c8 79 22 fa c8 14	d8 ca c8 f6 22 fa c8 14	00 00 00 8f 00 00 00 00
9f 06 fe 94 b3 3d 20 6a	9f 06 fe 80 b3 3d 20 6a	00 00 00 14 00 00 00 00
5a d6 10 10 51 4c a3 7a	5a d6 10 2b 51 4c a3 7a	00 00 00 3b 00 00 00 00

Table 1. Example of a 4.75-round near collision for the compression function.

m	m'	Difference at M_7^P
ba aa da d1 92 9e 95 f5	3b 16 1b b0 76 fe 1e 78	
3a 4a 35 2c 61 a8 84 f1	4c 03 4f 12 d1 a3 b4 bd	
44 38 38 e2 d2 fa 5e ec	c6 a7 81 ff 3a c7 3e 36	
27 00 09 05 4f 53 05 f2	6c 76 3e 0a d6 92 72 00	
cd 02 30 bb 3e b4 54 df	47 7e c6 e0 a4 6e 23 1a	
fc c6 de 98 54 4e 5c b6	28 a4 20 68 ee e1 01 11	d7 4d 00 c8 00 00 00 00
60 dc 52 73 dc c9 5d f1	43 20 0a 43 12 ba fe a0	ff 60 00 60 00 00 00 00
72 99 45 8d 9b c8 73 f2	8a d2 ff b3 19 f4 e4 25	15 3c 00 c9 00 00 00 00
		1b 49 00 ae 00 00 00 00
k	k'	03 81 00 42 00 00 00 00
		1a ed 00 ea 00 00 00 00
f4 d7 d6 42 05 a4 b9 7a	75 6b 17 23 e1 c4 32 f7	37 8e 00 60 00 00 00 00
2f 70 68 1a 2c 59 f4 4e	59 39 12 24 9c 52 c4 02	61 b8 00 f2 00 00 00 00
8b 7b 44 12 38 36 84 87	09 e4 fd 0f d0 0b e4 5d	
63 04 2f 7d de 3d b9 9f	28 72 18 72 47 fc ce 6d	
78 db 37 55 73 39 f7 30	f2 a7 c1 0e e9 e3 80 f5	
3f f2 8d fb 23 a9 6a 8a	eb 90 73 0b 99 06 37 2d	
20 18 3a e4 63 85 3a 81	03 e4 62 d4 ad f6 99 d0	
b5 58 8a e7 d3 34 20 4d	4d 13 30 d9 51 08 b7 9a	

Table 2. Example of a 5-round collision and 7.75-round near collision for the internal block cipher (E).

References

1. The National Hash Standard of the Russian Federation GOST R 34.11-2012. Russian Federal Agency on Technical Regulation and Metrology report, 2012. https://www.tc26.ru/en/GOSTR34112012/GOST_R_34_112012_eng.pdf.
2. Canteaut, A., Fuhr, T., Naya-Plasencia, M., Paillier, P., Reinhard, J.-R., and Videau, M. A unified indifferentiability proof for permutation- or block cipher-based hash functions. Cryptology ePrint Archive, Report 2012/363, 2012. http://eprint.iacr.org/2012/363.
3. Chang, S., Perlner, R., Burr, W.E., Turan, M., Kelsey, J., Paul, S. and Bassham, L.E. Third-round report of the SHA-3 cryptographic hash algorithm competition. `http://nvlpubs.nist.gov/nistpubs/ir/2012/NIST.IR.7896.pdf`,2012.
4. Gilbert, H., and Peyrin, T. Super-Sbox Cryptanalysis: Improved attacks for AES-like permutations. In *FSE* (2010), S. Hong and T. Iwata, Eds., vol. 6147 of *Lecture Notes in Computer Science*, Springer, pp. 365–383.
5. Jean, J., and Fouque, P.-A. Practical near-collisions and collisions on round-reduced ECHO-256 compression function. In *FSE* (2011), A. Joux, Ed., vol. 6733 of *Lecture Notes in Computer Science*, Springer, pp. 107–127.
6. Knudsen, L. R. Truncated and higher order differentials. In *FSE* (1995), B. Preneel, Ed., vol. 1008 of *Lecture Notes in Computer Science*, Springer, pp. 196–211.

7. Lamberger, M., Mendel, F., Rechberger, C., Rijmen, V., and Schläffer, M. Rebound distinguishers: Results on the full Whirlpool compression function. In *ASIACRYPT* (2009), M. Matsui, Ed., vol. 5912 of *Lecture Notes in Computer Science*, Springer, pp. 126–143.
8. Matusiewicz, K., Naya-Plasencia, M., Nikoli, I., Sasaki, Y., and Schläffer, M. Rebound attack on the full lane compression function. In *ASIACRYPT* (2009), M. Matsui, Ed., vol. 5912 of *Lecture Notes in Computer Science*, Springer, pp. 106–125.
9. Matyukhin, D., and Shishkin, V. Some methods of hash functions analysis with application to the GOST P 34.11-94 algorithm. *Mat. Vopr. Kriptogr 3* (2012), 71–89.
10. Mendel, F., Peyrin, T., Rechberger, C., and Schläffer, M. Improved cryptanalysis of the reduced Grøstl compression function, ECHO permutation and AES block cipher. In *Selected Areas in Cryptography* (2009), M. J. Jacobson Jr, V. Rijmen, and R. Safavi-Naini, Eds., vol. 5867 of *Lecture Notes in Computer Science*, Springer, pp. 16–35.
11. Mendel, F., Pramstaller, N., Rechberger, C., Kontak, M., and Szmidt, J. Cryptanalysis of the GOST hash function. In *CRYPTO* (2008), D. Wagner, Ed., vol. 5157 of *Lecture Notes in Computer Science*, Springer, pp. 162–178.
12. Mendel, F., Rechberger, C., Schlffer, M., and Thomsen, S. S. The rebound attack: Cryptanalysis of reduced Whirlpool and Grøstl. In *FSE* (2009), O. Dunkelman, Ed., vol. 5665 of *Lecture Notes in Computer Science*, Springer, pp. 260–276.
13. Mendel, F., Rechberger, C., Schlffer, M., and Thomsen, S. S. Rebound attacks on the reduced Grøstl hash function. In *CT-RSA* (2010), J. Pieprzyk, Ed., vol. 5985 of *Lecture Notes in Computer Science*, Springer, pp. 350–365.
14. Rijmen, V., Toz, D., and Varc, K. Rebound attack on reduced-round versions of JH. In *FSE* (2010), S. Hong and T. Iwata, Eds., vol. 6147 of *Lecture Notes in Computer Science*, Springer, pp. 286–303.
15. Sasaki, Y., Wang, L., Wu, S., and Wu, W. Investigating fundamental security requirements on Whirlpool: Improved preimage and collision attacks. In *ASIACRYPT* (2012), X. Wang and K. Sako, Eds., vol. 7658 of *Lecture Notes in Computer Science*, Springer, pp. 562–579.
16. Wagner, D. The boomerang attack. In *Fast Software Encryption* (1999), L. R. Knudsen, Ed., vol. 1636 of *Lecture Notes in Computer Science*, Springer, pp. 156–170.
17. Wang, X., Yin, Y. L., and Yu, H. Finding collisions in the full SHA-1. In *CRYPTO* (2005), V. Shoup, Ed., vol. 3621 of *Lecture Notes in Computer Science*, Springer, pp. 17–36.
18. Wang, X., and Yu, H. How to break MD5 and other hash functions. In *EUROCRYPT* (2005), R. Cramer, Ed., vol. 3494 of *Lecture Notes in Computer Science*, Springer, pp. 19–35.