Impossible Differential Properties of Reduced Round Streebog

Ahmed Abdelkhalek, Riham AlTawy, and Amr M. Youssef

Concordia Institute for Information Systems Engineering
Concordia University, Montréal, Québec, Canada

Abstract. In this paper, we investigate the impossible differential properties of the underlying block cipher and compression function of the new cryptographic hashing standard of the Russian federation Streebog. Our differential trail is constructed in such a way that allows us to recover the key of the underlying block cipher by observing input and output pairs of the compression function which utilizes the block cipher in Miyaguchi-Preneel mode. We discuss the implication of this attack when utilizing Streebog to construct a MAC using the secret-IV construction.
Keywords: Cryptanalysis, Hash functions, MAC, Secret-IV, Miss-in-the-middle, Impossible Differential, GOST R 34.11-2012, Streebog.

1 Introduction

In late 2012, Streebog [2] was announced as the new Russian cryptographic hashing standard GOST R 34.11-2012. It officially replaced GOST R 34.11-94 which has been theoretically broken in [25] and further analyzed in [24,23]. The output length of the Streebog hash function can be either 512 or 256-bit. Its compression function is based on a 12-rounds AES-like block cipher with 8×8-byte internal state, followed by an XOR operation with a whitening key. The compression function operates in Miyaguchi-Preneel mode and is plugged in Merkle-Damgård domain extender with a finalization step [19].

Literature related to the cryptanalysis of Streebog includes the analysis of the collision resistance of its compression function and internal cipher by AlTawy *et al.* [3], and Wang *et al.* [27]. An integral analysis of the compression function has been presented by AlTawy and Youssef where integral distinguishers for the reduced compression function was proposed [4]. Moreover, preimage attacks on the reduced hash function have been independently proposed by Altawy and Youssef [5], and Zou *et al.* [28], and later the attacks were improved by Bingka *et al.* [22]. Also, Kazymyrov and Kazymyrova presented an analysis of the algebraic aspects of the function [19], and a long second preimage attack was proposed by Guo *et al.* [17]. Finally, a malicious version of the whole hash function was presented in [7], and a differential fault analysis of the function when used in different MAC schemes was proposed [6].

A Message Authentication Code (MAC) [8] is a symmetric-key construction that provides mutual entity authentication and data integrity. Two common approaches are used to construct MAC schemes. The first approach employs a block cipher or a permutation, e.g., Cipher Block Chaining (CBC)-MAC [1], PELICAN-MAC [14], and ALPHA-MAC [13]. The second approach is based on hash functions where a secret key shared between the communicating parties is processed in a specific construction by the hash function which is consequently viewed as a keyed hash function. Examples of this approach include simple prefix MAC [26], secret-IV MAC [26], NMAC [8], and the internationally standardized HMAC [8]. Attacks on MAC schemes usually aim to investigate their resistance against forgery attacks and key recovery attacks. The latter attack is more devastating since it directly grants the attacker the ability to impersonate any of the communicating parties and consequently forge any given message. As a result, analyzing hash-based MACs with respect to key recovery attacks has been the main aspect of many proposed works [18,16].

When considering a hash function in a given MAC scheme, the first step is to analyze the security of the underlying primitives operating in the secret key model against key recovery attacks. Consequently, key recovery attacks on the underlying primitives has been considered as a valuable analytic model for the hash function. Such model has been adopted by Bouillaguet *et al.* in their analysis of the SHA-3 submission Lesamnta [11], where they presented a key recovery attack on the internal cipher reduced to 22 rounds. Additionally, the cryptanalysis of the SHA-3 submission EDON-R [21], where Laurent presented a key recovery attack on the function used in the Secret-IV MAC. One of the prospective applications of Streebog, as any other hash function, is using it in MAC schemes. Though both the simple prefix and the secret-IV MACs are vulnerable to length extension attacks, and the nested HMAC construction is internationally standardized, Streebog is by design not vulnerable to length extension attacks. This property may tempt users to adopt simpler MAC constructions such as the secret-IV setting. In this approach, the standard initial value is replaced by the secret key in the iterative construction of the hash function. More formally, $MAC(M) = H(K, M)$, where $H(K, M)$ is the hash value of the message M using the secret key K as the IV. Indeed, the designers of the NIST SHA-3 hash function, keccak [9,12], state on their website that since keccak is not vulnerable to length extension attacks, it does not need HMAC and propose that MAC computation can be done by concatenating the key with the message [20]. It should also be noted that the proof of security of the Miyaguchi-Preneel mode assumes that the underlying block cipher is ideal and must exhibit no distinguishing property. Accordingly, the results presented in this work are also interesting from this perspective since they are relevant to these indistinguishability claims.

In 2000, Biham and Keller presented a 4-round impossible differential property of AES [10] that was the basis for all the succeeding impossible differential attacks on AES. This property specifies that given an input pair at round i which has just one non-zero difference byte (in the literature, this is usually referred to

as an active byte), the corresponding pair at round $i+3$ cannot be equal in any of the four columns after applying the ShiftRow transformation. This 4-round impossible differential property consists of two deterministic differentials; one 2-round forward differential and the other is a 2-round backward differential that contradict with each other.

In this work, we provide a security evaluation of Streebog's compression function when used in the secret key model where the IV is replaced by a secret key. More precisely, we present an impossible differential (ID) property of the underlying block cipher and compression function and employ it to recover the secret-IV of the compression function. Table 1 provides a summary of current cryptanalytic results on the Streebog hash function.

Target	#Rounds	Time	Memory	Data	Attack	Reference
Internal cipher	5	2^8	2^8	-	Free-start collision	[3]
	8	2^{64}	2^8	-		
	3.75	-	-	-	ID distinguisher	Sec. 3
Internal permutation	6.5	2^{64}	-	2^{64}	Distinguisher	[4]
	7.5	2^{120}	-	2^{120}		
Compression function	7.75	2^{184}	2^8	-	Semi free-start collision	[3]
	4.75	2^8	-	-		
	7.75	2^{72}	2^8	-	Semi free-start near collision	
	8.75	2^{128}	2^8	-		
	9.75	2^{184}	2^8	-		
	6.75	$2^{399.5}$	2^{349}	$2^{483.1}$	Secret-IV recovery	Sec. 4
	6	2^{64}	-	2^{64}	Distinguisher	[4]
	7	2^{120}	-	2^{120}		
Hash function	5	2^{122}	2^{64}	-	Collision	[28]
	6	2^{496}	2^{64}	-	Preimage	[22]
	12	-	2^{14}	-	Differential fault analysis	[6]
	12	2^{266}	–	-	Long second preimage	[17]

Table 1. Summary of the current cryptanalytic results on Streebog.

The rest of the paper is organized as follows. In the next section, the specification of the Streebog hash function along with the notation used throughout the paper are provided. A brief overview of impossible differential cryptanalysis is

given in Section 3. Afterwards, in Section 4, we provide detailed description of the impossible differential attack on the underlying block cipher and the complexity of the attack. Finally, the paper is concluded in Section 5.

2 Specification of Streebog

Streebog 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

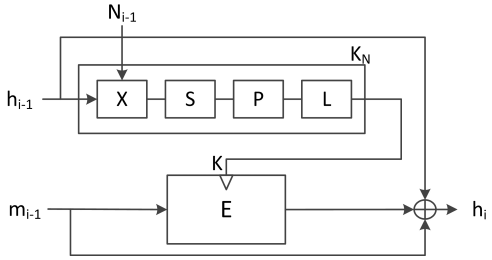

Fig. 1. Streebog's compression function g_N

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: a chaining value h_{i-1}, a message block m_{i-1}, and a 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 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 a 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.

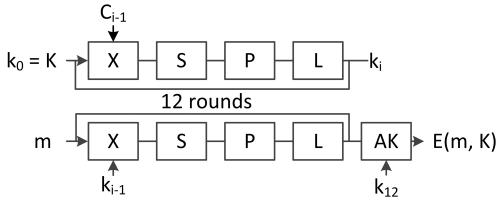

Fig. 2. The internal block cipher (E)

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, the 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.
- LinearTransformation (L): Left multiplication by an MDS matrix in GF(2).

Initially, the key 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 $E(K_N(h_{i-1}, N_{i-1}), m_{i-1}) \oplus m_{i-1} \oplus h_{i-1}$. For further details, the reader is referred to [2].

2.1 Notation

Let M be (8×8)-byte states denoting an input message state. The following notation will be used throughout the paper:

- M_i^I: A state at the beginning of round i.
- M_i^X, M_i^S, M_i^P and M_i^O: The message state at round i after the application of AddKey, SubBytes, Transposition and Linear Transformation, respectively. intuitively, $M_{i-1}^O = M_i^I$ for $i >= 2$.
- $M_i[r, c]$: A byte at row r and column c of state M_i. Another representation of state bytes is an enumeration $0, 1, 2, 3, ..., 63$ as shown in Figure 3.
- M_i[row r]: Eight bytes located at row r of M_i state.
- M_i[col c]: Eight bytes located at column c of M_i state.

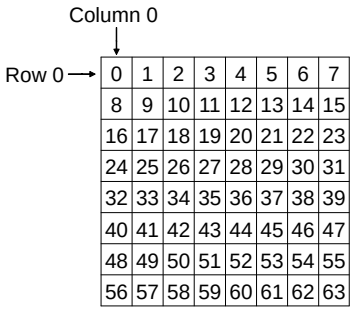

Fig. 3. The 8×8 state of Streebog

3 Impossible Differential Cryptanalysis of the Compression Function

Although Streebog's compression function employs an AES-like cipher, applying the commonly used 4-round impossible differential of AES as is on Streebog's compression function would not be of value as in this case, we would recover the key of the last round of the block cipher masked by the chaining value (recall that Streebog's compression function works in Miyaguchi-Preneel mode). Therefore, we opted to reverse the impossible differential as detailed below to help recover the key of the first round, i.e., k_0. Since the key scheduling is bijective, once k_0 is recovered, we can recover the secret chaining value in the case of a secret-IV MAC construction when applied only at the level of the compression function. We note that the impossible differential property of Streebog's compression function would be limited to 3.75-rounds because, unlike AES, in the Streebog underlying block cipher, the linear transformation in the last round is not omitted. As depicted in Figure 4, this impossible differential property states that given a pair of M_i^I with any 7 active bytes in the same arbitrary row (row 0 is chosen in the figure for illustration purposes), M_{i+3}^P cannot have only one active byte (similarly, that active byte can be any byte out of the 64-byte state). The deterministic differentials in this property are different than those of the AES property; on top of being swapped, the forward differential is just 1-round while the backward differential is 2.75 rounds. The property is rationalized as follows: any 7 active bytes in the same row of M_i^I give 56 active bytes by M_i^O with one entire row being equal (which row will depend on the position of the zero-difference byte in the input). On the other hand, one active byte in M_{i+3}^P leads to a full active state where all 64 bytes are active in M_{i+1}^I, which means that the middle states contradict with each other as illustrated in Figure 4. As explained above, this impossible differential holds regardless of the row and also the positions within that row. Figure 5 gives an example for impossible input and output patterns for the compression function. When the compression function message input has specific non-zero difference at bytes 0 to 6 (δ_0 to δ_6 in the figure) and zero difference in all the other bytes, then after the feedforward

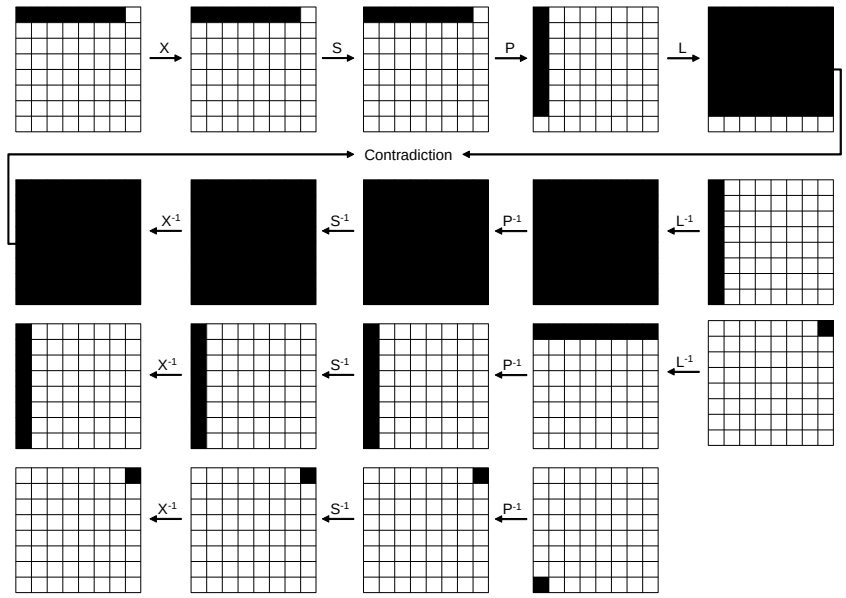

Fig. 4. Impossible differential property of the internal block cipher

the output difference cannot have the same values as the input difference at bytes 0 to 6 (δ_0 to δ_6) and a non-zero difference at byte 56 (δ_7). Such input and output difference patterns are impossible on the compression function level. It is to be noted that there exists $8 \times 8 \times 255^7$ such input patterns (the input differences can be in any row and the inactive byte can be at any column of that row and each of the differences can take $2^8 - 1$ possible values. There are also $((7 \times 8) + 1) \times 255 = 57 \times 255$ contradicting output patterns (the non-zero output difference byte can be at any column of 7 rows, i.e., all but the input differences row and takes only the position of the inactive byte on the input differences row).

4 Impossible Differential Attack on 6.75 rounds of the Compression Function

Considering the aforementioned impossible differential property, a 6.75-rounds attack on Streebog's compression function can be mounted as detailed hereafter. In our attack model, we assume access to the keyed Streebog's reduced compression function oracle which allows us to query the keyed oracle with chosen messages and get the corresponding compression function outputs.

4.1 Attack Algorithm

The attack is illustrated in Figure 6. In what follows, we give its details.

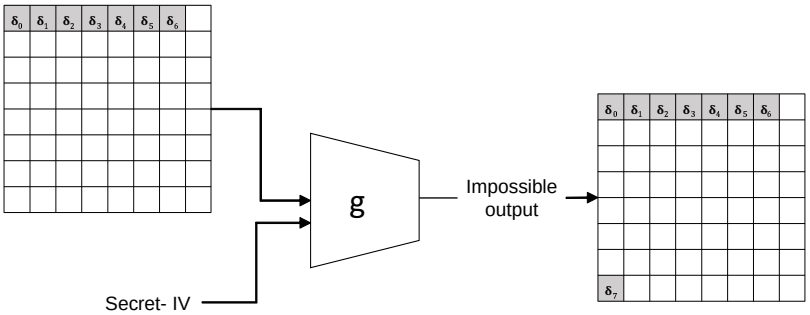

Fig. 5. Example of impossible differentials for the 3.75 round reduced compression function

1. The keyed compression function oracle is fed with 2^n structures where each structure consists of 2^{256} messages having the same value in columns 4, 5, 6 and 7 and assuming all possible 2^{256} values in columns 0, 1, 2 and 3. Accordingly, each structure offers $2^{256} \times 2^{256} \times 1/2 \simeq 2^{511}$ pairs of messages. Thus we have a total of 2^{n+256} messages, and 2^{n+511} message pairs for the 2^n structures.

2. Since we have access to the output of the compression function which operates in Miyaguchi-Preneel mode as depicted in Figure 1, the output h_i that we observe is $h_{i-1} \oplus m_{i-1} \oplus c_{i-1}$ where c_{i-1} is the corresponding output of the reduced variant of the block cipher when its input is m_{i-1}. Therefore, for each message query, we first XOR the compression function output with the corresponding input message, i.e., $h_i \oplus m_{i-1}$, to get $c_{i-1} \oplus h_{i-1}$ and keep only the pairs that have non-zero difference in just one column. Consequently, it is expected to have $2^{n+511} \times 2^{-448} = 2^{n+63}$ pairs.

3. Next, we randomly assume a 256-bit value of the first round key at columns 0, 1, 2 and 3, i.e., $k_0[\text{col } 0, 1, 2, 3]$, partially encrypt these 4 columns of the message pairs corresponding to the remaining ciphertext pairs, i.e., we compute $M_1^O[\text{row } 0, 1, 2, 3] = L \circ P \circ S[M_1^I[\text{col } 0, 1, 2, 3] \oplus k_0[\text{col } 0, 1, 2, 3]]$ and we choose the pairs which have just one non-zero difference byte at column 0 of the 4 rows, as shown in Figure 6. The probability of such combination is $q_1 = (2^{-8})^7 \times (2^{-8})^7 \times (2^{-8})^7 \times (2^{-8})^7 = 2^{-224}$. Accordingly, $2^{n+63} \times 2^{-224} = 2^{n-161}$ message pairs are expected to pass this step.

4. Afterwards, we assume a 32-bit value for the bytes 0, 1, 2 and 3 of column 0 of the key k_1, i.e., bytes 0, 8, 16, 24 as in Figure 3, partially encrypt these bytes through the second round to compute $M_2^O[\text{row } 0] = L \circ P \circ S[M_2^I[(0, 1, 2, 3), 0] \oplus k_1[(0, 1, 2, 3), 0]$. We choose the pairs which have only one zero-difference byte at any column of that row. The probability of such pairs is $q_2 = 8 \times 2^{-8} = 2^{-5}$. So after this step, we have $2^{n-161} \times 2^{-5} = 2^{n-166}$ message pairs.

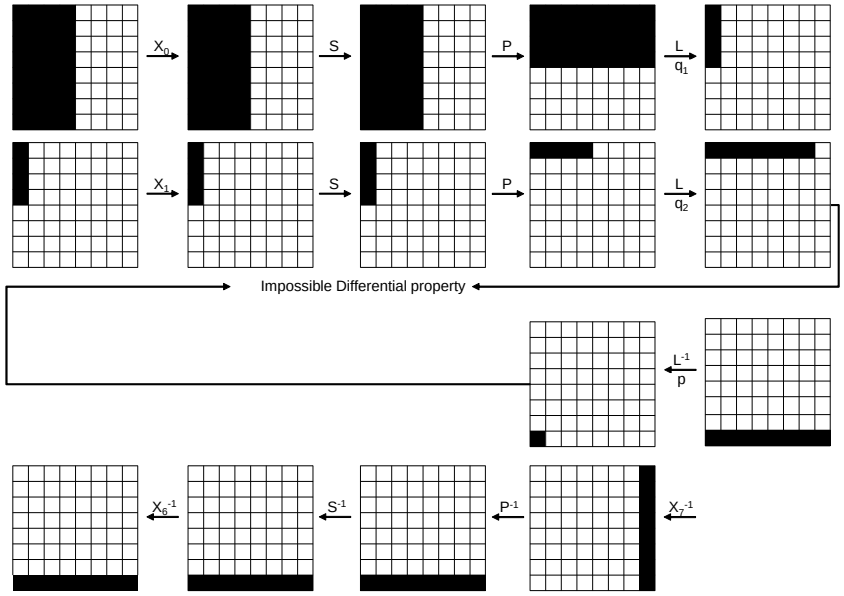

Fig. 6. 6.75-rounds impossible differential attack on the compression function

5. Then, we assume a 64-bit value for the last column of $k_7[\text{col } 7] \oplus h_{i-1}[\text{col } 7]$ so that we end up with the block cipher output (Note: h_{i-1} is the targeted secret-IV to be recovered and it has the same value for all the pairs we have). Specifically, we calculate $M_7^P[\text{col } 7] = (c_{i-1}[\text{col } 7] \oplus h_{i-1}[\text{col } 7]) \oplus (k_7[\text{col } 7] \oplus h_{i-1}[\text{col } 7])$ for all the pairs we have so far. The former value is the output we get from step 2, while the latter is the value we just assumed.

6. For each of the filtered pairs, we partially decrypt column 7. In other words, we compute $L^{-1} \circ ((S^{-1} \circ P^{-1}(M_7^P[\text{col } 7])) \oplus (S^{-1} \circ P^{-1}(M_7^{*P}[\text{col } 7]))$ and we choose the pairs which have only one non-zero difference byte at any position of row 7, which happens with probability $p = 8 \times (2^{-8})^7 = 2^{-53}$. This difference is impossible, hence each key (or to be exact each $k_7 \oplus h_{i-1}$ i.e., $k_7 \oplus const$) that results in such a difference is a wrong key. Therefore, after analyzing 2^{n-166} pairs, only $2^{64} \times (1-2^{-53})^{2^{n-166}} \simeq 2^{64} \times (e^{-1})^{2^{n-219}} \simeq 2^{64} \times 2^{-1.4 \times (2^{n-219})}$ wrong values of the last column of $k_7 \oplus h_{i-1}$ remains.

To be able to find the correct partial keys, we discard the 64-bit values for $k_7 \oplus h_{i-1}$ unless the initial guess of the 256-bit value of k_0 and the 32-bit value of k_1 is correct. The wrong values (k_0, k_1, k_7) remain with probability: $2^{(256+32)} \times 2^{64} \times 2^{-1.4 \times (2^{n-219})} = 2^{352-1.4 \times (2^{n-219})}$ which should be made as small as possible, e.g., less than 2^{-30} (that value is chosen to maximize the probability of finding the correct tuple without having a significant impact on the number of messages needed) which means $2^{352-1.4 \times (2^{n-219})} < 2^{-30}$ resulting in $n > 227.09$. Accordingly, when we start with $2^{227.1}$ structures and there

remains a value of $k_7 \oplus h_{i-1}$, we consider the assumed 256-bit value for k_0 is correct and the probability of wrong value of (k_0, k_1, k_7) is $2^{-32.1}$.

4.2 Attack Complexity

With n set to 227.1, the attack requires $2^{n+256} = 2^{483.1}$ chosen messages. The time complexity of the attack is calculated as follows:

- In step 3, row 0 requires $2 \times 2^{64} \times 2^{n+63} \times 1/8 = 2^{n+125}$ one round encryptions, row 1 requires $2 \times 2^{64} \times 2^{64} \times 2^{n+7} \times 1/8 = 2^{n+133}$ one round encryptions, row 2 requires $2 \times 2^{64} \times 2^{64} \times 2^{64} \times 2^{n-49} \times 1/8 = 2^{n+141}$ one round encryptions and row 3 requires $2 \times 2^{64} \times 2^{64} \times 2^{64} \times 2^{64} \times 2^{n-105} \times 1/8 = 2^{n+149}$ one round encryptions.
- In step 4, row 0 requires $2 \times 2^{256} \times 2^{32} \times 2^{n-161} \times 1/16 = 2^{n+124}$ one round encryptions.
- In step 6, column 7 decryption requires $2 \times 2^{256} \times 2^{32} \times 2^{64} \times (1 + (1 - 2^{-53}) + (1 - 2^{-53})^2 + ... + (1 - 2^{-53})^{2^{n-166}}) \times 1/16 \simeq 2^{402}$ one round encryptions.
- For $n = 227.1$, the overall complexity of the attack is about $(2^{352.1} + 2^{360.1} + 2^{368.1} + 2^{376.1} + 2^{351.1} + 2^{402})/6.75 \simeq 2^{399.5}$ encryptions to recover 256 bits of k_0.

Then, the other half of k_0 can be found by an exhaustive search. Hence the whole k_0 can be recovered with time complexity of $2^{399.5} + 2^{256} \simeq 2^{399.5}$ queries. Once k_0 is recovered, we can easily recover the secret-IV h_{i-1}. Finally, the memory requirement is dominated by the memory needed to store the list of the deleted key tuples (k_0,k_1,k_7), so we need $2^{352}/2^3 = 2^{349}$ bytes.

5 Conclusion

In this paper, we have analyzed Streebog's compression function in the secret key model. More precisely, we have proposed Secret-IV recovery attack, at the level of the compression function, based on impossible differential properties of the compression function. The attack requires $2^{483.1}$ messages, has a time complexity equivalent to $2^{399.5}$ queries to the compression function reduced to 6.75 rounds and needs 2^{349} bytes of memory.

Finally, it should be noted that this attack does not directly contradict the security claims of Streebog and does not present any immediate practical threat to its security. However, it helps as a cautionary note for using Streebog in this mode since it might be tempting to do so because the finalization stage of Streebog is strengthened against length extension attacks which are the main reasons for not using secret-IV or secret-prefix MAC constructions. It is interesting to note that, while the results in [15] show that the modular sum finalization stage weakens the function when used in HMAC construction, extending our attack to the full hash function remains a challenge and requires further investigation of the compression function one wayness properties.

References

1. ISO/IEC 9797-1. Information Technology-Security techniques-Data integrity mechanism using a cryptographic check function employing a block cipher algorithm. International Organizatoin for Standards.
2. 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/GOSTR3411\discretionary-2012/GOST_R_34_11\discretionary-2012_eng.pdf.
3. AlTawy, R., Kircanski, A., and Youssef, A. M. Rebound Attacks on Stribog. In *ICISC* (2013), H.-S. Lee and D.-G. Han, Eds., vol. 8565 of *Lecture Notes in Computer Science*, Springer, pp. 175–188.
4. AlTawy, R., and Youssef, A. M. Integral Distinguishers for Reduced-round Stribog. *Information Processing Letters 114*, 8 (2014), 426 – 431.
5. AlTawy, R., and Youssef, A. M. Preimage Attacks on Reduced-Round Stribog. In *AFRICACRYPT* (2014), D. Pointcheval and D. Vergnaud, Eds., vol. 8469 of *Lecture Notes in Computer Science*, Springer, pp. 109–125.
6. AlTawy, R., and Youssef, A. M. Differential Fault Analysis of Streebog. In *ISPEC* (2015), J. Lopez and Y. Wu, Eds., vol. 9065 of *Lecture Notes in Computer Science*, Springer, pp. 35–49.
7. AlTawy, R., and Youssef, A. M. Watch your Constants: Malicious Streebog. *IET Information Security* (2015). *(to appear)*.
8. Bellare, M., Canetti, R., and Krawczyk, H. Keying Hash Functions for Message Authentication. In *Advances in Cryptology CRYPTO 96* (1996), N. Koblitz, Ed., vol. 1109 of *Lecture Notes in Computer Science*, Springer, pp. 1–15.
9. Bertoni, G., Daemen, J., Peeters, M., and Van Assche, G. Keccak sponge function family main document. *Submission to NIST (Round 2) 3* (2009).
10. Biham, E., and Keller, N. Cryptanalysis of Reduced Variants of Rijndael. In *3rd AES Conference, New York, USA* (2000).
11. Bouillaguet, C., Dunkelman, O., Leurent, G., and Fouque, P.-A. Attacks on Hash Functions based on Generalized Feistel - Application to Reduced-Round Lesamnta and SHAvite-3_{512}. Cryptology ePrint Archive, Report 2009/634, 2009. http://eprint.iacr.org/2009/634.pdf.
12. Chang, S.-j., Perlner, R., Burr, W. E., Turan, M. S., Kelsey, J. M., Paul, S., and Bassham, L. E. *Third-round report of the SHA-3 cryptographic hash algorithm competition.* Citeseer, 2012.
13. Daemen, J., and Rijmen, V. A New MAC Construction ALRED and a Specific Instance ALPHA-MAC. In *Fast Software Encryption* (2005), H. Gilbert and H. Handschuh, Eds., vol. 3557 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 1–17.
14. Daemen, J., and Rijmen, V. The Pelican MAC Function. Cryptology ePrint Archive, Report 2005/088, 2005. http://eprint.iacr.org/2005/088.pdf.
15. Dinur, I., and Leurent, G. Improved Generic Attacks against Hash-Based MACs and HAIFA. In *CRYPTO* (2014), J. Garay and R. Gennaro, Eds., vol. 8616 of *Lecture Notes in Computer Science*, Springer, pp. 149–168.
16. Fouque, P.-A., Leurent, G., and Nguyen, P. Full Key-Recovery Attacks on HMAC/NMAC-MD4 and NMAC-MD5. In *Advances in Cryptology - CRYPTO 2007* (2007), A. Menezes, Ed., vol. 4622 of *Lecture Notes in Computer Science*, Springer, pp. 13–30.

17. Guo, J., Jean, J., Leurent, G., Peyrin, T., and Wang, L. The Usage of Counter Revisited: Second-Preimage Attack on New Russian Standardized Hash Function. In *SAC* (2014), A. Joux and A. Youssef, Eds., vol. 8781 of *Lecture Notes in Computer Science*, Springer, pp. 195–211.
18. Handschuh, H., and Preneel, B. Key-Recovery Attacks on Universal Hash Function Based MAC Algorithms. In *Advances in Cryptology CRYPTO 2008* (2008), D. Wagner, Ed., vol. 5157 of *Lecture Notes in Computer Science*, Springer, pp. 144–161.
19. Kazymyrov, O., and Kazymyrova, V. Algebraic Aspects of the Russian Hash Standard GOST R 34.11-2012. In *CTCrypt* (2013), pp. 160–176. Available at: http://eprint.iacr.org/2013/556.pdf.
20. Keccak team. "Strengths of Keccak - Design and security". http://keccak.noekeon.org/. Last Accessed: 2014-02-20.
21. Leurent, G. Practical Key Recovery Attack against Secret-IV Edon-R. In *Topics in Cryptology - CT-RSA 2010* (2010), J. Pieprzyk, Ed., vol. 5985 of *Lecture Notes in Computer Science*, Springer, pp. 334–349.
22. Ma, B., Li, B., Hao, R., and Li, X. Improved Cryptanalysis on Reduced-Round GOST and Whirlpool Hash Function. In *Applied Cryptography and Network Security* (2014), I. Boureanu, P. Owesarski, and S. Vaudenay, Eds., vol. 8479 of *Lecture Notes in Computer Science*, Springer, pp. 289–307.
23. 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.
24. Mendel, F., Pramstaller, N., and Rechberger, C. A (second) Preimage Attack on the GOST Hash Function. In *FSE* (2008), K. Nyberg, Ed., vol. 5086 of *Lecture Notes in Computer Science*, Springer, pp. 224–234.
25. 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.
26. Preneel, B., and van Oorschot, P. C. On the Security of Iterated Message Authentication Codes. *IEEE Transactions on Information Theory 45*, 1 (1999), 188–199.
27. Wang, Z., Yu, H., and Wang, X. Cryptanalysis of GOST R Hash Function. *Information Processing Letters 114*, 12 (2014), 655–662.
28. Zou, J., Wu, W., and Wu, S. Cryptanalysis of the Round-Reduced GOST Hash Function. In *Information Security and Cryptology* (2014), D. Lin, S. Xu, and M. Yung, Eds., Lecture Notes in Computer Science, Springer, pp. 309–322.