Side Channel Cryptanalysis of Streebog

Gautham Sekar(✉)

Indian Statistical Institute, Chennai Centre,
SETS Campus, MGR Knowledge City,
Taramani, Chennai 600113, India
sgautham@isichennai.res.in

Abstract. Streebog is the cryptographic hash function standard of the Russian Federation. It comprises two hash functions corresponding to two digest sizes, 256 bits and 512 bits. This paper presents a side channel attack that uses processor flag information to speed up message recovery by a factor of 2. Success is nearly guaranteed if the flag is set; the probability is 0.668 otherwise.

Keywords: Cryptographic hash function · Streebog · Side channel cryptanalysis · Carry flag · Message recovery · HMAC

1 Introduction

A hash function F takes an arbitrarily long bit string m as input and outputs a fixed length bit string H (called *hash value* or *digest*). A cryptographic hash function is meant to satisfy certain security properties, the most important of which are the following.

- **(First) preimage resistance**: given H, it is computationally infeasible to find an m' such that $F(m') = H$.
- **Second preimage resistance**: given an m and $F(m)$, it is computationally infeasible to find an $m' \neq m$ such that $F(m') = F(m)$.
- **Collision resistance**: it is computationally infeasible to find an m and an $m' \neq m$ such that $F(m) = F(m')$.

The general model for cryptographic hash functions involves what is called a compression function. The function transforms a bit string of a fixed length into a shorter string of a fixed length. The arbitrarily long message is partitioned into blocks after a process called padding (described later in the context of Streebog). The blocks are then sequentially processed, with the compression function acting on every block until all the blocks are processed. The final output is the hash value. The general model is described in good detail in [9, Sect. 2.4.1].

Streebog is a set of two hash functions and a Russian cryptographic standard (GOST R 34.10–2012) [5]. It was developed by the Center for Information Protection and Special Communications of the Federal Security Service of the Russian Federation, with participation of the Open Joint-Stock Company

© Springer International Publishing Switzerland 2015
L. Chen and S. Matsuo (Eds.): SSR 2015, LNCS 9497, pp. 154–162, 2015.
DOI: 10.1007/978-3-319-27152-1_8

"Information Technologies and Communication Systems" (JSC "InfoTeCS") [5], following a demand for "a hash function to meet modern requirements for cryptographic strength" [5]. In 2012, Streebog replaced GOST R 34.11–94 as the national standard.

The hash functions comprising Streebog have 256 bits and 512 bits as their digest lengths. We shall call the hash functions "Streebog-256" and "Streebog-512", respectively. The compression function, common to both the versions, operates on 512-bit blocks in the Miyaguchi-Preneel mode, has 13 rounds, is based on a substitution-permutation network and uses a linear transformation.

In 2010–2011, open research workshops were organised by the Chinese Academy of Sciences to discuss cryptographic algorithms proposed for inclusion in the LTE/4G mobile standards. In a seemingly similar fashion, between 2013 and 2015, the Russian Technical Committee for Standardization "Cryptography and Security Mechanisms" (TC 26), with the participation of the Academy of Cryptography of the Russian Federation and support from the JSC InfoTeCS, held an open research competition for the analysis of Streebog. In this period, several results were reported, notably in [1–3,6,10].

In [1,10], the rebound attack is used to find (semi-free-start) collisions for reduced versions of the Streebog compression function; [2] presents integral distinguishers on up to 7 rounds of the compression function; [3] reports preimages for 6-round Streebog; and [6] describes second preimage attacks on the full Streebog-512. The drawback of the attacks in [6] is that they work well only with long messages. For instance, if the length of the message is at least 2^{188} bits, then 2^{342} compression function evaluations are required. The time complexity can be brought down to as low as $O(2^{266})$ provided that the message is at least 2^{268} bits in length. For shorter messages, of bit-length $\gamma < 2^{188}$ (but greater than 512 bits), the number of compression function evaluations is estimated at $(\log_2 \gamma - 9) \cdot 2^{522-\log_2 \gamma}$. We present in this paper the first side channel attack on the full Streebog. We also discuss the implications of our attack on the security of Streebog-based keyed-hash message authentication code (HMAC).

Processors have registers that store information on operations performed by their ALUs. For example, in the Intel IA-32 architecture, the *status flags* of the EFLAGS register indicate the result of arithmetic instructions such as ADD and DIV (divide) [7]. One of these flags, known as the *carry flag*, is a single bit that indicates an overflow in unsigned integer arithmetic. For instance, when two unsigned integers are added, the carry flag is set (to 1) if a carry is generated by the addition at the most significant bit position (we shall call this an *end carry*) and the flag is cleared (i.e., 0) otherwise. This may be exploited by an attacker as in [8] where Kelsey *et al.* use carry flag information to attack the block cipher RC5. In our side channel attack too we use the state of the carry flag. Our attack recovers a message block in about 2^{511} time with 99.9 % success rate (number of successful recoveries per 100 messages uniformly distributed at random) if the carry flag is set and 66.8 % otherwise. The only other attack known on the full Streebog is due to Guo *et al.* [6].

Table 1. Notation and conventions

Symbol/notation	Meaning
$\|W\|$	length of W in bits
$\Gamma_i(W)$	ith 64-bit word of W; $i = 0$ denotes the least significant word
$W_{(i)}$	ith bit of W; $i = 0$ denotes the least significant bit
$\|\|$	concatenation
$\oplus$	exclusive OR
fg, where f and g are functions	$f \circ g$ (composition of f and g)
LSB	least significant bit
MSB	most significant bit

The paper is organised as follows. Section 2 describes Streebog and Sect. 3 details our meesage recovery attack. We propose countermeasures to our attack in Sect. 4 and conclude in Sect. 5.

2 Description of Streebog

Table 1 lists the notation and conventions followed in the rest of this paper.

Streebog is a simple design that uses only a few elementary arithmetic operators such as XOR and modular addition, and simple functions such as substitution, permutation and linear transformation. The hash function accepts any message M of length less than 2^{512} bits and returns a digest of length 256 bits or 512 bits. The round function or compression function has 13 iterations, the first twelve of which involve a substitution-permutation layer. If $512 \nmid |M|$, then padding prefixes M with a bit string $pad{:=}\{0\}^{511-(|M| \mod 512)}\|1$. The padded message is then partitioned into $(k+1)$ 512-bit blocks $M_k, M_{k-1}, \ldots, M_0$; i.e., $pad\|M = M_k\|M_{k-1}\|\cdots\|M_0$. The compression function g that processes the message block M_i takes as additional inputs the chaining value H_i (of size 512 bits) and a length counter N_i, and outputs H_{i+1}. Algorithm 1 describes the working of Streebog. The IV in the algorithm is the initial value H_0 (Streebog-256 and Streebog-512 use different 512-bit IVs).

The substitution-permutation layer includes the following components.

- Substitution function S: The input, a 512-bit string, is first partitioned into bytes. Every byte is then substituted by a byte from a set π', which is a permutation of $\{0, 1, \ldots, 255\}$, and concatenated.
- Permutation function P: Partitions its 512-bit input into bytes, permutes the bytes (i.e., shuffles their positions) and concatenates them.
- Linear transformation L: This is also a 512-bit-to-512-bit mapping. If the input is W, then $L(W) = \ell(\Gamma_7(W))\|\ell(\Gamma_6(W))\|\cdots\|\ell(\Gamma_0(W))$, where ℓ is a 64-bit-to-64-bit linear transformation that outputs the right multiplication of its input with a constant matrix $\mathbf{A}$ over $GF(2)$.

Algorithm 1. The Streebog algorithm

Require: The message M, $|M| < 2^{512}$
Ensure: A 256-bit or a 512-bit digest
1: $M \to pad\|M \to M_k\|M_{k-1}\|\cdots\|M_0$;
2: $H_0 = IV$;
3: $N_0 = 0$;
4: **for** $i = 0$ to $(k-1)$ **do**
5: $\quad H_{i+1} = g(H_i, M_i, N_i)$;
6: $\quad N_{i+1} = N_i + 512 \mod 2^{512}$;
7: $\quad \Sigma \leftarrow \Sigma + M_i \mod 2^{512}$;
8: $H_{k+1} = g(H_k, M_k, N_k)$;
9: $N_{k+1} = N_k + \alpha \mod 2^{512}$, where $\alpha = 512 - |pad|$;
10: $\Sigma \leftarrow \Sigma + M_k \mod 2^{512}$;
11: $H_{k+2} = g(H_{k+1}, N_{k+1}, 0)$;
12: $H = g(H_{k+2}, \Sigma, 0)$;
13: Output H if Streebog-512, else output $H \gg 256$;

– The function $X[\cdot]$: If K and W are 512-bit strings, then $X[K](W) = K \oplus W$.

The compression function g is now given by:

$$g(H_i, M_i, N_i) = E(L(P(S(H_i \oplus N_i))), M_i) \oplus H_i \oplus M_i\,, \tag{1}$$

where

$$E(L(P(S(H_i \oplus N_i))), M_i) = X[K_{13}]LPSX[K_{12}]LPSX[K_{11}]\ldots \\ LPSX[K_1](M_i)\,, \tag{2}$$

(recall from Table 1 that $fg = f \circ g$) and

$$K_0 = LPS(H_i \oplus N_i)\,, \tag{3}$$

$$K_{j+1} = LPS(K_j \oplus C_j)\,, \quad \text{for } j = 0, 1, \ldots, 12, \text{ and constants } C_j. \tag{4}$$

The subkeys $K_1, K_2, \ldots, k_{13}$ are the round keys; in deriving them, K_0 is used as an initial value.

3 The Message Recovery Attack

The functions S and P do not involve modular addition or multiplication. The function X is a simple XOR operation. The linear transformation ℓ works as follows. Denoting its 64-bit input by $\beta{:=}\beta_{(63)}\|\beta_{(62)}\|\cdots\|\beta_{(0)}$, we have:

$$\ell(\beta) = \bigoplus_{i=0}^{63} \beta_{(63-i)} \odot A[i]\,,$$

where the product $\odot$ is defined as follows:

$$\beta_{(63-i)} \odot A[i] = \begin{cases} \{0\}^{64} & \beta_{(63-i)} = 0\,; \\ A[i] & \beta_{(63-i)} = 1\,. \end{cases}$$

Hence, from (1)–(4), it immediately follows that Streebog compression does not involve any operation, such as addition modulo 2^{512}, that can alter the state of the carry flag. This means that only steps 6, 7, 9 and 10 of Algorithm 1 can potentially affect the carry flag.[1]

Now, the maximum length of M is $2^{512} - 1$. Given a message of this length, the number of blocks will be $\lceil (2^{512} - 1)/512 \rceil = 2^{503}$.[2] If $k + 1 < 2^{503}$ (to simply calculations, this can be considered a sure event as it happens with a probability that is very close to 1 if $|M|$ is uniformly distributed at random over $\{0, 1, \ldots, 2^{512} - 1\}$), then $N_k = 512k$, $512k < N_{k+1} \leq 512(k+1)$, and the carry flag will be unaffected by steps 6 and 9. This leaves us with steps 7 and 10. Now,

$$\Sigma = \left(\sum_{i=0}^{k-1} M_i \right) \mod 2^{512} + M_k \mod 2^{512}\,. \tag{5}$$

$$= T_{k-1} + M_k \mod 2^{512}\,, \text{ say.} \tag{6}$$

Let $C{:=}[C_{(511)} C_{(510)} \cdots C_{(0)}]$ denote the vector of carries generated in (6) such that $C_{(0)}$ is the carry at the LSB position. When $k \geq 1$ (this can also be considered a sure event), we have the following attack.

Scenario 1: Suppose that the carry flag is set at the end of Algorithm 1. If $|pad| \geq 2 \Rightarrow M_{k(511)} = 0$, or $|pad| = 0$ and $M_{k(511)} = 0$, then $T_{k-1(511)} = C_{(511)} = 1$. If the attacker knows $M_0, M_1, \ldots, M_{k-2}$, and all but the MSB of M_{k-1}, then she can recover $M_{k-1(511)}$ from $T_{k-1(511)} = 1$ performing $k - 1 < 2^{503} - 2$ additions (recall (5) and (6)).

If $|pad| = 0$ and $M_{k(511)} = 1$, or $|pad| = 1 \Rightarrow M_{k(511)} = 1$, then there are three possibilities: *(i)* $T_{k-1(511)} = C_{(511)} = 1$, *(ii)* $T_{k-1(511)} = 0$ and $C_{(511)} = 1$, *(iii)* $T_{k-1(511)} = 1$ and $C_{(511)} = 0$. Assuming these cases to be equally likely,[3] the attacker can assume with 2/3 probability that $T_{k-1(511)} = 1$, and recover $M_{k-1(511)}$.

Table 2 lists the above cases and their probabilities assuming that *(i)* $|M_k|$ is uniformly distributed at random over $\{0, 1, \ldots, 511\}$, and *(ii)* every message

[1] The `for`-loop of Algorithm 1 is implemented differently in [5]. To obtain M_0, the least significant 512-bit word of the padded message is extracted. The leftover message replaces the padded message and its 512 LSBs are extracted as M_1. This process is repeated until all the message blocks have been extracted. The carry flag is evidently unaffected by the process.

[2] Therefore, even if we go with the `for`-loop implementation (Algorithm 1), it will have no bearing on the carry flag.

[3] Since the distribution of $|M_k|$ is uniform, given the padding scheme employed, the distribution of M_k is not uniform. This makes it tedious to compute the distribution of the carry vector C. Hence the assumption.

block other than M_k is uniformly distributed at random over $\{0, 1, \ldots, 2^{512} - 1\}$. The attack methodology is as follows. The attacker, knowing $M_0, M_1, \ldots, M_{k-2}$ and M_k, makes a guess for the 511 LSBs of M_{k-1}, obtains a value for the MSB of M_{k-1} (assuming that $T_{k-1(511)} = 1$), hashes $M_k \| M_{k-1} \| \cdots \| M_0$, and compares the digest with the given hash value. If the values do not agree, the guess is incorrect and the attacker makes another guess. The process is repeated until the hash values agree. The sum σ of $M_0, M_1, \ldots, M_{k-2}$ modulo 2^{512} can be precomputed (cost is $k - 2$); $\sigma + M_{k-1} \mod 2^{512}$ can be performed at each guess and, in doing so, can be avoided while computing the digest (i.e., $\sigma + M_{k-1} \mod 2^{512}$ can be stored and reused). To minimise memory usage, the storage element can be rewritten at the next guess. The probability of success is the probability that $T_{k-1(511)} = 1$ holds true. From Table 2, this probability is simply $510/512 + 1/768 + 1/1024 + 1/1536 \approx 0.999$. The attack requires 2^{511} hash function evaluations plus a precomputation cost of $k - 2 < 2^{503} - 3$. Memory requirements are negligible.

Table 2. Computing $Pr(T_{k-1(511)} = 1)$ when the carry flag is 1; the probability q is given the condition on $|pad|$ and r is given the conditions on $|pad|$ and $M_{k(511)}$

$\|pad\|$	Pr. (p)	$M_{k(511)}$	Cond. pr. (q)	$T_{k-1(511)}$	Cond. pr. (r)	Overall pr. (pqr)
≥ 2	510/512	0	1	1	1	510/512
1	1/512	1	1	1	2/3	1/768
0	1/512	0	1/2	1	1	1/1024
0	1/512	1	1/2	1	2/3	1/1536

Scenario 2: Suppose that the carry flag is 0 at the end of Algorithm 1. If $|pad| \geq 2 \Rightarrow M_{k(511)} = 0$, or $|pad| = 0$ and $M_{k(511)} = 0$, then at least one of $T_{k-1(511)}$ and $C_{(511)}$ is 0. Knowing $M_0, M_1, \ldots, M_{k-2}$, and all but the MSB of M_{k-1}, the attacker can recover $M_{k-1(511)}$ assuming that $T_{k-1(511)} = 0$. The assumption is valid in two out of the three possible cases: *(i)* $T_{k-1(511)} = C_{(511)} = 0$, *(ii)* $T_{k-1(511)} = 0$ and $C_{(511)} = 1$, *(iii)* $T_{k-1(511)} = 1$ and $C_{(511)} = 0$. Assuming that these cases are equally likely, $Pr(T_{k-1(511)} = 0) = 2/3$.

When $|pad| = 0$ and $M_{k(511)} = 1$ or when $|pad| = 1 \Rightarrow M_{k(511)} = 1$, then $T_{k-1(511)} = C_{(511)} = 0$.

Table 3 lists the above cases and their probabilities under the assumption that *(i)* $|M_k|$ is uniformly distributed at random over $\{0, 1, \ldots, 511\}$, and *(ii)* every message block other than M_k is uniformly distributed at random over $\{0, 1, \ldots, 2^{512} - 1\}$. The attack methodology is identical to that described under Scenario 1, except that the attacker here assumes that $T_{k-1(511)} = 0$. The probability of success is the probability that $T_{k-1(511)} = 0$ holds true. From Table 3, this probability is $170/256 + 1/512 + 1/1536 + 1/1024 \approx 0.668$. The time complexity and memory requirements are the same as that in Scenario 1.

Note: The probability that $T_{k-1} = 0$ given that the carry flag is 0 and $M_{k(511)} = 0$ is at least 1/2 since $Pr(\text{case } (i) \text{ or case } (ii)) = Pr(T_{k-1}) = 1/2$ (given the assumption that the message blocks other than M_k are uniformly distributed). Even if the conditional probability is 1/2, the success probability will be $255/512 + 1/512 + 1/2048 + 1/1024 > 1/2$ (see Table 3). The success probability calculated from Table 2 changes negligibly when 2/3 is replaced by 1/2. □

Table 3. Computing $Pr(T_{k-1(511)} = 0)$ when the carry flag is 0; the probability q is given the condition on $|pad|$ and r is given the conditions on $|pad|$ and $M_{k(511)}$

$\|pad\|$	Pr. (p)	$M_{k(511)}$	Cond. pr. (q)	$T_{k-1(511)}$	Cond. pr. (r)	Overall pr. (pqr)
≥ 2	510/512	0	1	0	2/3	170/256
1	1/512	1	1	0	1	1/512
0	1/512	0	1/2	0	2/3	1/1536
0	1/512	1	1/2	0	1	1/1024

In summary, by simply guessing $T_{k-1(511)}$ to be equal to the carry flag, the attacker is able to recover M_{k-1} with 2^{511} hash function evaluations and $k-2$ precomputations. The number of precomputations can be negligible in comparison to 2^{511} and even the maximum number of precomputations $(2^{503} - 4)$ is considerably smaller than 2^{511}. Moreover, each precomputation is only an addition of two 512-bit integers. Consequently, the precomputation cost can be ignored. The success probability is 0.668 if the carry flag is 0 and 0.999 otherwise. Arriving at a single value for the probability is involved given the difficulty in determining the distribution of the carry vector C. It is easy to see that the attack works for any $i \in \{0, 1, \ldots, k-2\}$ in place of $k-1$. In the ideal case, either 2^{512} hash function evaluations are required or the success probability is 1/2 for 2^{511} evaluations.[4] Since the compression functions of Streebog-256 and Streebog-512 are identical, our attack applies to both the hash functions.

3.1 Implications of Our Attack

Our attack may be particularly relevant to HMACs. Proposed by Bellare *et al.* [4] as a message integrity checking mechanism, a HMAC employs a hash function h in conjunction with a secret key K and generates a MAC value as follows:

$$HMAC(K, m) = h((K_0 \oplus opad) \| h((K_0 \oplus ipad) \| m)) \,,$$

where m is the message, *opad* and *ipad* are public constants, and K_0 is the secret key or a function of K. The lengths of K_0, *opad* and *ipad* equal the length of a

[4] This does not apply to M_k unless $|pad| = 0$. Knowing $|pad|$ and $M_0, M_1, \ldots, M_{k-1}$, the attacker can recover M_k in $2^{512-|pad|}$ time. Our attack is not intended to recover M_k.

message block. Given the HMAC value and $h((K_0 \oplus ipad)\|m)$, in certain cases, our attack appears to speed up the recovery of K_0 by a factor of 2. This is being further investigated.

4 Countermeasures

A simple way to preclude our attack is to introduce a low-cost arithmetic operation, after step 12 of Algorithm 1, that permanently sets or clears the carry flag. However, the approach fails if the attack model assumes that the attacker can determine the status of the carry flag after step 12.[5]

A faster and safer countermeasure is to implement the checksum using XOR; i.e., replace the addition modulo 2^{512} in steps 7 and 10 of Algorithm 1 with XOR.

5 Conclusions

In this paper, we have presented the first known side channel attack on Streebog. The attack speeds up message recovery by a factor of 2 with a probability that lies in $[0.668, 0.999]$. The attack is conjectured to be applicable to Streebog-based HMAC. We have also proposed some countermeasures.

It may be possible to improve the attack by recovering bits other than the MSB, but calculating the success probabilities is involved and beyond the scope of this paper. We leave it as a problem for future work. Use of other processor flags such as the parity flag is also worth investigating.

References

1. AlTawy, R., Kircanski, A., Youssef, A.M.: Rebound attacks on Stribog. In: Lee, H.-S., Han, D.-G. (eds.) ICISC 2013. LNCS, vol. 8565, pp. 175–188. Springer, Heidelberg (2014)
2. AlTawy, R., Youssef, A.M.: Integral distinguishers for reduced-round Stribog. Inf. Process. Lett. **114**(8), 426–431 (2014)
3. AlTawy, R., Youssef, A.M.: Preimage attacks on reduced-round Stribog. In: Pointcheval, D., Vergnaud, D. (eds.) AFRICACRYPT 2014. LNCS, vol. 8469, pp. 109–125. Springer, Heidelberg (2014)
4. Bellare, M., Canetti, R., Krawczyk, H.: Keying hash functions for message authentication. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 1–15. Springer, Heidelberg (1996)
5. Federal Agency on Technical Regulation and Metrology, "NATIONAL STANDARD OF THE RUSSIAN FEDERATION GOST R 34.11-2012" (English Version), 1 January 2013
6. Guo, J., Jean, J., Leurent, G., Peyrin, T., Wang, L.: The usage of counter revisited: second-preimage attack on new Russian standardized hash function. In: Joux, A., Youssef, A. (eds.) SAC 2014. LNCS, vol. 8781, pp. 195–211. Springer, Heidelberg (2014)

[5] A similar assumption is made in [8].

7. Intel, "IA-32 Intel Architecture Software Developer's Manual", vol. 1 (Basic Architecture), p. 426 (2003). http://flint.cs.yale.edu/cs422/doc/24547012.pdf
8. Kelsey, J., Schneier, B., Wagner, D., Hall, C.: Side channel cryptanalysis of product ciphers. J. Comput. Secur. **8**, 141–158 (2000)
9. Preneel, B.: Analysis and Design of Cryptographic Hash Functions, PhD thesis, Katholieke Universiteit Leuven (1993)
10. Wang, Z., Yu, H., Wang, X.: Cryptanalysis of GOST R hash function. Inf. Process. Lett. **114**(12), 655–662 (2014)