Keyed Streebog is a secure PRF and MAC

Vitaly Kiryukhin

LLC «SFB Lab», JSC «InfoTeCS», Moscow, Russia
vitaly.kiryukhin@sfblaboratory.ru

Abstract

One of the most popular ways to turn a keyless hash function into a keyed one is the HMAC algorithm. This approach is too expensive in some cases due to double hashing. Excessive overhead can sometimes be avoided by using certain features of the hash function itself. The paper presents a simple and safe way to create a keyed cryptoalgorithm (conventionally called «Streebog-K») from hash function Streebog $\mathsf{H}(M)$. Let K be a secret key, then $\mathsf{KH}(K, M) = \mathsf{H}(K||M)$ is a secure pseudorandom function (PRF) and, therefore, a good message authentification code (MAC). The proof is obtained by reduction of the security of the presented construction to the resistance of the underlying compression function to the related key attacks (PRF-RKA). The security bounds of Streebog-K are essentially the same as those of HMAC-Streebog, but the computing speed doubles when short messages are used.

Keywords: Streebog, Streebog-K, PRF, MAC, HMAC, provable security

1 Introduction

The HMAC algorithm was proposed in 1996 [10] as an efficient way to construct a keyed transformation (and, most importantly, a secure message authentification code) from a keyless hash function $\mathsf{H}(M)$

$$\mathsf{HMAC}(K, M) = \mathsf{H}\left((\overline{K} \oplus opad)||\mathsf{H}(\overline{K} \oplus ipad||M)\right),$$

where $\overline{K}$ is obtained by padding the secret key K with zero bits, $opad$ and $ipad$ are different nonzero constants.

The security proof [10] explicitly assumes the use of a «plain» Merkle-Damgård [7, 8] cascade as an underlying hash function $\mathsf{H}(M)$: the message M is padded and splitted into b-bit blocks; the compression function g is iteratively applied to the previous n-bit state and b-bit block; the initial state IV is the predefined constant; the last state is the result of hashing. The result largely depends, among other things, on the weak collision resistance (WCR) of H in the «secret initial state» setting.

Collision resistance is broken in practice for several widely used hash functions, such as MD5 and SHA-1. In 2006, an updated proof was presented in [14], showing that the HMAC-MD5 and HMAC-SHA-1 nevertheless remain secure. The reduction shows that HMAC is a secure pseudorandom function (PRF) if g is a secure PRF (in the secret key and also in some restricted related-key settings). The proof was obtained via a non-uniform reduction (with «non-constructible» adversaries), leading to the insignificance of this result in practice [13].

In [13], along with a critique of the results [14], an alternative proof was also presented (the definition of PRF is slightly different). More precise bounds with the same initial requirements [14] and without the use of a «non-uniform computation model» were also obtained in the works [15, 17].

Russian hash function Streebog [1] can also be used in the HMAC [3, 6]. Streebog uses a modified Merkle-Damgård approach. Its compression function is based on a 12-rounds AES-like block cipher in Miyaguchi-Preneel mode. The internal state and the message block consist of $n = 512$ bits. The output length of hash function can be either 512 or 256-bit.

The most important differences from the «plain» cascade are the following:

– before processing the i-th block, the state is summed modulo 2 with the number of already hashed bits;

– the last call of the compression function is used to «mix» the checksum (modulo 2^n) of all message blocks.

It is important to note that the differences between Streebog and the Merkle-Damgård scheme do not allow direct use of the results [10, 14, 13, 15, 17] for HMAC-Streebog. The proof of the latter's security was, among other things, given in [16]. However, the reduction descends not to the properties of the compression function g, but to the properties of the hash function H itself.

Unfortunately, HMAC-Streebog has a significant overhead when working with short messages. We have at least 8 (resp. 9) calls of g for HMAC-Streebog-256 (resp. 512). However, the design of Streebog implicitly generates a more efficient solution.

The aforementioned features allow us to prove that «Streebog-K» («Keyed Streebog») $\mathsf{KH}(K, M) = \mathsf{H}(\overline{K}||M)$ is a secure pseudorandom function (PRF) under some plausible assumptions about the compression function g. Thus, processing a short message requires 4 computations of g: padded key, padded message, bit length, checksum. It is also easy to see that the proposed cryptoalgorithm *does not require any changes* in Streebog itself. Other methods of involving K, such as secret-IV [11], along with simplifying the finalization and a number of small changes can provide a more computationally efficient and no less secure solution. Unfortunately, all this requires edits in the formal description of the hash function and in many existing implementations. Therefore, we consider «Streebog-K» that is devoid of these disadvantages.

The security of Streebog against the length-extension attack (i.e. the particular case of PRF-security) is explicitly claimed in [5]. However, as far as we know, there are no publicly available formal proofs.

The analysis of Streebog-K was carried out in the paradigm of provable security [19, 18]. We start from high-level description of Streebog (section 3) and its equivalent representation [22] carefully considered in the proof. Next, in section 4 we present and discuss «hard-to-solve» problems: the indistinguishability of g from family of random functions under related key attacks (PRF-RKA) in two various settings. In the main part of the paper (section 5) we reduce the PRF properties of Streebog-K to the PRF-RKA properties of g. Roughly speaking, if there is an effective attack against Streebog-K, then there is an attack against g. The reduction gives us the upper bound on the probability of the adversary's success (for example, the forgery or the key recovery). The bound functionally depends on the capabilities of the adversary (amount of the computation resources, the number of adaptively chosen input-output pairs).

Two «beyond security bound» attacks against Streebog-K were also briefly considered (section 6). The first is the simple forgery attack, the second one is the key recovery attack, almost identical to the same against HMAC-Streebog [23].

Similar proofs can be also relatively easily obtained for HMAC-Streebog [3] and S3G [4]. The corresponding results are briefly discussed in section 7. The security bounds are almost the same in all cases, but Streebog-K requires a weaker notion of PRF-RKA-security from g.

The good security bounds in the PRF setting allow you to use Streebog-K as a secure MAC and key derivation function.

2 Notations and definitions

We use the following notations throughout the paper:

$n = 512$ – block size in bits; $k \leq 512$ – key size in bits; $\oplus$ – bitwise XOR operation; $\boxplus$, $\boxminus$ – addition and subtraction modulo $2^n = 2^{512}$;
$||$ – concatenation of binary strings;

V^* – the set of all binary strings of a finite length;

V^n – the set of all n-bit strings with naturally defined operations «$\oplus$» and «$\boxplus$»;

$V^{\leq L}$ – the set of binary strings of length no more than L bits;

$(V^n)^{\leq l}$ – the set of binary strings of length no more than $l \cdot n$ bits, the length of each string is a multiple of n;

$\mathsf{bin}(x)$ – n-bit representation of the integer x;

$\mathsf{sum}_{\boxplus}(M) = m_1 \boxplus m_2 \boxplus \ldots \boxplus m_l$ – the checksum (modulo 2^n) of blocks from l-block message $M = m_1||m_2||...|m_l$;

$\mathsf{sum}'_{\boxplus}(M) = m_1 \boxplus \ldots \boxplus m_{l-1}$ – the checksum of all blocks from the message $M = m_1||m_2||...|m_l$, except for the last block;

$\mathrm{Func}(\mathbf{X}, \mathbf{Y})$ – the set of all mappings from the set $\mathbf{X}$ to the set $\mathbf{Y}$;

$X \xleftarrow{\mathrm{R}} \mathbf{X}$ – uniform and random selection of element X from the set $\mathbf{X}$.

The adversary is modeled by an interactive probabilistic algorithm that has access to other algorithms (oracles). We denote by $\mathrm{Adv}_{\mathsf{F}}^{TM}(\mathcal{A})$ a quantitative characterization (advantage) of the capabilities of the adversary $\mathcal{A}$ in realizing a certain threat, defined by the model TM, for the cryptographic scheme F. The resources of $\mathcal{A}$ are measured in terms of time and query complexities. The time complexity t includes the description size of $\mathcal{A}$ in some computation model. The query complexity q is measured in the number of adaptively chosen input/output pairs. If F has a variable input length, the maximum length $l_{\max}$ of the query (in n-bit blocks) is also characteristic of the adversary's resources. Without loss of generality, we assume that $\mathcal{A}$ always uses exactly q unique queries (no redundancy and repetitions). The result of computations $\mathcal{A}$ after interacting with oracles $\mathcal{O}_1$, $\mathcal{O}_2$, ... $\mathcal{O}_w$, $w \in \mathbb{N}$ is some value x (usually binary), $\mathcal{A}^{\mathcal{O}_1, \mathcal{O}_2, \ldots \mathcal{O}_w} \Rightarrow x$.

The maximum of the advantage among all resource constrained adversaries is denoted by

$$\mathrm{Adv}_{\mathsf{Alg}}^{TM}(t, q, l_{\max}) = \max_{\mathcal{A}(t',q',l'):t' \leq t,\ q' \leq q,\ l' \leq l_{\max}} \mathrm{Adv}_{\mathsf{Alg}}^{TM}(\mathcal{A}).$$

The cryptoalgorithm Alg is called secure in the threat model TM with respect to adversaries limited by resources $(t, q, l_{\max})$ if $\mathrm{Adv}_{\mathsf{F}}^{TM}(t, q, l_{\max}) < \varepsilon$, where ε is some small value determined by the requirements for the strength of the cryptosystem.

To demonstrate the practical significance of the obtained results, we sometimes substitute heuristic estimates based on assumptions into derived security bounds. The resulting informal estimates are denoted by symbol «$\lessapprox$» meaning «less or equal if the assumptions are true».

Definition. The advantage of $\mathcal{A}$ in the model PRF (PRF-CMA – indistinguishability from a random function under chosen message attack) for the keyed cryptoalgorithm $\mathsf{F} : \mathbf{K} \times \mathbf{X} \to \mathbf{Y}$ is

$$\mathrm{Adv}_{\mathsf{F}}^{PRF}(\mathcal{A}) = \Pr\left(K \xleftarrow{R} \mathbf{K}; \mathcal{A}^{\mathsf{F}_K(\cdot)} \Rightarrow 1\right) - \Pr\left(\widetilde{\mathsf{F}} \xleftarrow{R} \mathrm{Func}(\mathbf{X}, \mathbf{Y}); \mathcal{A}^{\widetilde{\mathsf{F}}(\cdot)} \Rightarrow 1\right),$$

where $\mathbf{K}$, $\mathbf{X}$, $\mathbf{Y}$ are spaces of the keys, messages, and outputs respectively.

As the example, for a PRF with a fixed input length, we have $(\mathbf{K}, \mathbf{X}, \mathbf{Y}) = (V^n, V^n, V^n)$. For Streebog-K, $(\mathbf{K}, \mathbf{X}, \mathbf{Y}) = (V^k, V^{\leq L}, V^n)$.

3 Streebog and Streebog-K

Streebog hashes the message $M \in V^*$ as follows. The text is padded with bit string $\mathtt{10\ldots0}$. At least one bit is always added, even if the message bit length L is already divisible by n. The string $M' = M||\mathtt{10\ldots0}$ is divided into l blocks of $n = 512$ bits $m_1||m_2||\ldots||m_l$. The compression function is sequentially applied to the previous state, the block and the counter

$$h_{i+1} = \mathsf{g}(h_i, m_{i+1}, \mathbf{i}),\ i = 0, ..., l-1,\ h_0 = IV \in V^n,$$

where IV is a predefined constant which is different in both versions of the hash function, the counter $\mathbf{i} = \mathsf{bin}(i \cdot n) \in V^n$ is the number of already hashed bits.

Two more transformations are performed at the finalizing stage: the bit length L and the checksum $\Sigma = \mathsf{sum}_{\boxplus}(M')$ are «mixed» with the state

$$h_{l+1} = \mathsf{g}(h_l, L, \mathbf{0}),\ H = \mathsf{g}(h_{l+1}, \Sigma, \mathbf{0}).$$

If 256-bit hash function is used, the output H truncated to 256 bit.

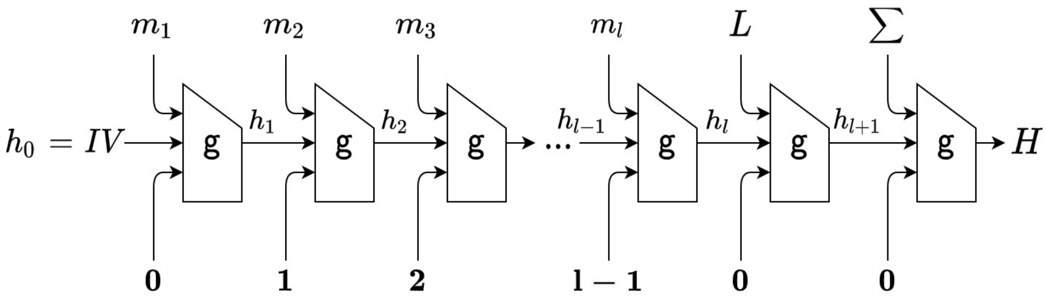

Figure 1: Keyless hash function Streebog-512.

The compression function is based on a 12-rounds AES-like block cipher E in Miyaguchi-Preneel mode

$$\mathsf{g}(h_i, m_{i+1}, \mathbf{i}) = \mathsf{E}(h_i \oplus \mathbf{i}, m_{i+1}) \oplus h_i \oplus m_{i+1} = h_{i+1}.$$

In [22], the equivalent representation was proposed (see the detailed figure in the Appendix)

$$h_{i+1} = \underbrace{\mathsf{E}(h_i \oplus \mathbf{i}, m_{i+1}) \oplus (h_i \oplus \mathbf{i}) \oplus m_{i+1}}_{\mathsf{g}'(h_i \oplus \mathbf{i}, m_{i+1})} \oplus \mathbf{i},$$
$$h_{i+1} = \mathsf{g}'(h_i \oplus \mathbf{i}, m_{i+1}) \oplus \mathbf{i},$$
$$h_{i+2} = \mathsf{g}'(\mathsf{g}'(h_i \oplus \mathbf{i}, m_{i+1}) \oplus \underbrace{\mathbf{i} \oplus (\mathbf{i} \boxplus \mathbf{1})}_{\Delta_i}, m_{i+2}) \oplus (\mathbf{i} \boxplus \mathbf{1}).$$

Adjacent counters are summed with each other. However, the last counter appears differently $h_l = \mathsf{g}'(h_{l-1} \oplus (\mathbf{l} \boxminus \mathbf{1}), m_l) \oplus \underbrace{(\mathbf{l} \boxminus \mathbf{1})}_{\widetilde{\Delta}_{l-1}}$.

Hence, $\mathsf{g}(h_i, m_{i+1}, \mathbf{i})$ is replaced by

$$\mathsf{g}(h_i, m_{i+1}) = \mathsf{E}(h_i, m_{i+1}) \oplus h_i \oplus m_{i+1} \oplus \Delta_i, \ i = 0, \dots, l-2,$$
$$\mathsf{g}(h_i, m_{i+1}) = \mathsf{E}(h_i, m_{i+1}) \oplus h_i \oplus m_{i+1} \oplus \widetilde{\Delta}_i, \ i = l-1,$$

and the sequence of unique counters $\mathbf{i}$ is replaced by a «quasi-periodic» one $\Delta_i = \mathbf{i} \oplus (\mathbf{i} \boxplus \mathbf{1})$, for example,

$$\Delta_0, \Delta_1, \dots, \widetilde{\Delta}_{15} = \mathbf{1}, \mathbf{3}, \mathbf{1}, \mathbf{7}, \mathbf{1}, \mathbf{3}, \mathbf{1}, \mathbf{15}, \mathbf{1}, \mathbf{3}, \mathbf{1}, \mathbf{7}, \mathbf{1}, \mathbf{3}, \mathbf{1}, \mathbf{15},$$
$$\Delta_0, \Delta_1, \dots, \Delta_{15}, \widetilde{\Delta}_{16} = \mathbf{1}, \mathbf{3}, \mathbf{1}, \mathbf{7}, \mathbf{1}, \mathbf{3}, \mathbf{1}, \mathbf{15}, \mathbf{1}, \mathbf{3}, \mathbf{1}, \mathbf{7}, \mathbf{1}, \mathbf{3}, \mathbf{1}, \mathbf{31}, \mathbf{16}.$$

Also, it is important that $\Delta_i \neq \widetilde{\Delta}_i \ \forall i = 0, ..., 2^n - 1$.

The keyed cryptoalgorithm Streebog-K defined as (fig. 2)

$$\mathsf{KH}(K, M) = \mathsf{H}(\overline{K}||M) = \mathsf{H}(K||\underbrace{0...0}_{n-k}||M), \ K \in V^k, \ \overline{K} \in V^n,$$

where $256 \leq k \leq 512 = n$ and K is padded with zero bits if necessary (as in [3]). Streebog-256 and Streebog-512 can be used as H without significant differences in properties. Note that due to the key's prepending, the last value $\widetilde{\Delta}_l = \mathbf{l}$ has the index l, and not $l - 1$. Further in the text, the compression function means $\mathsf{g}(h, m) = \mathsf{E}(h, m) \oplus h \oplus m$.

4 Related key attack settings

The security proof presented in the next section shows that if the adversary can break PRF-security of Streebog-K, then one of the following two problems is also easy to solve. However, we expect these problems to be hard – g successfully resists attacks using related keys in (at least) two settings. Therefore, the security of Streebog-K is also difficult to break.

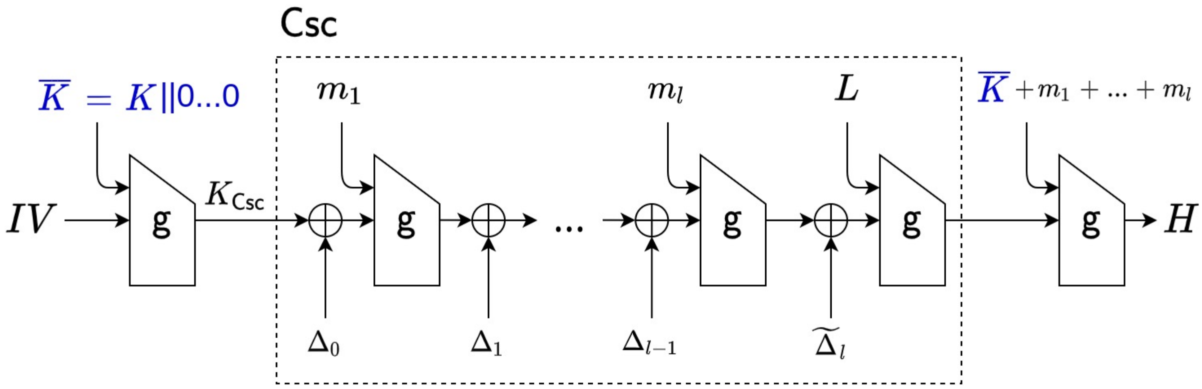

Figure 2: Streebog-K with the equivalent representation [22].

Problem 1. *PRF-RKA*$_\oplus$-security of $\mathsf{g}^{\triangleright}_K(\cdot) = \mathsf{g}(K, \cdot)$ in sense

$$\mathrm{Adv}^{PRF-RKA_\oplus}_{\mathsf{g}^{\triangleright}}(\mathcal{A}) = \Pr\left(K \xleftarrow{R} V^n,\ \mathcal{A}^{\mathsf{g}(K,\cdot),\mathsf{g}(K\oplus\phi,\cdot)} \Rightarrow 1\right) - \\ - \Pr\left(\mathsf{f}, \mathsf{f}' \xleftarrow{R} \mathrm{Func}(V^n, V^n),\ \mathcal{A}^{\mathsf{f}(\cdot),\mathsf{f}'(\cdot)} \Rightarrow 1\right)$$

– the pair of the compression functions (with the key K and with the related key $K \oplus \phi$) is indistinguishable from the pair of random functions. The value of $\phi \neq 0$ is chosen once by the adversary before the sequence of queries. The query consists of the block m and the binary flag «key K»/«key $K \oplus \phi$».

In the most favorable case, there are only two distinguishing methods: brute-force attack against two keys and birthday-paradox

$$\mathrm{Adv}^{PRF-RKA_\oplus}_{\mathsf{g}^{\triangleright}}(t, q) \lessapprox \frac{2 \cdot t}{2^n} + \frac{q^2}{2^{n+1}}.$$

Problem 2. *PRF-RKA*$_\boxplus$-security of $\mathsf{g}^{\triangledown}_K = \mathsf{g}(\cdot, K)$. The relation between the keys is modular addition

$$\mathrm{Adv}^{PRF-RKA_\boxplus}_{\mathsf{g}^{\triangledown}}(\mathcal{A}) = \Pr\left(K \xleftarrow{R} V^k; \mathcal{A}^{\mathsf{g}(\cdot,\overline{K}\boxplus\cdot)} \Rightarrow 1\right) - \\ - \Pr\left(K \xleftarrow{R} V^k; \mathsf{f}_{\overline{K}\boxplus\sigma} \xleftarrow{R} \mathrm{Func}(V^n, V^n),\ \forall\sigma \in V^n; \mathcal{A}^{\mathsf{f}_{\overline{K}\boxplus\cdot}(\cdot)} \Rightarrow 1\right).$$

The query consists of the block m and the value σ. The response is $\mathsf{g}^{\triangledown}_{\overline{K}\boxplus\sigma}(m) = \mathsf{g}(m, \overline{K} \boxplus \sigma)$ or $\mathsf{f}_{\overline{K}\boxplus\sigma}(m)$ correspondingly. We can hope that in the absence of specific vulnerabilities, the only possible attack is the parallel key guessing

$$\mathrm{Adv}^{PRF-RKA_\boxplus}_{\mathsf{g}^{\triangledown}}(t, q) \lessapprox \frac{t \cdot q}{2^{k-1}}.$$

The more related keys are used, the easier it is to carry out the attack.

The complexity of solving basic problems should be confirmed by constructive cryptanalysis of the compression function. The impossible differential single-key attack against $\mathsf{g}^{\triangleright}$ is presented in [24] and covers 6.75 out of

12 rounds. In [25], attacks on 7 rounds in the PRF model are proposed for $\mathsf{g}^{\triangleright}$ and $\mathsf{g}^{\triangledown}$. The attacks in the related-key settings against 10 and 11 rounds of $\mathsf{g}^{\triangledown}$ were recently proposed in [26].

Despite the many papers on the topic [27, 28, 29, 30, 31, 32, 33, 34], to the best of our knowledge, no effective full-round algorithms for constructing preimages and collisions of various types have been published. This is implicit evidence of the good cryptographic properties of g.

The existing results of cryptanalysis, as well as the conservative design of the underlying block cipher E and its key schedule, suggest that there are no special attacks on full-round versions of the compression function also in the PRF-RKA settings. In other words, the two basic problems under consideration are actually computationally hard.

The appearance of more efficient cryptographic methods of the compression function g will not render the presented security proof of Streebog-K incorrect. Specific attacks on g can be taken into account in the security bounds due to the absence of heuristic arguments in the proof.

5 Proof of PRF-security

Next, we show the reducibility of Streebog-K security to the problems discussed in the previous section. The equivalent representation (figure 2) is the start point $\mathsf{KH}^{(0)}(K, M) = \mathsf{KH}(K, M)$.

Step 1. We define the padding transformation $\mathsf{pad} : V^* \to (V^n)^{\leq l_{\max}}$ that sequentially adds to M:

– the nonempty binary string $\mathtt{10\ldots0}$ to achieve the multiplicity of the block length;

– the block $\mathsf{bin}(L)$ representing the length of M in bits.

Let M' consists of $l+1$ full-length blocks ($l \geq 1$), then

$$\mathsf{KH}^{(0)}(K, M) = \mathsf{KH}^{(1)}(K, \mathsf{pad}(M) = M||\mathtt{10\ldots0}||\mathsf{bin}(L)),\ M \in V^*,$$
$$\mathsf{KH}^{(1)}(K, M') = \mathsf{g}\left(\mathsf{Csc}(\mathsf{g}(IV, \overline{K}), M'),\ \overline{K} \boxplus \mathsf{sum}'_{\boxplus}(M')\right),\ M' \in (V^n)^{\leq l_{\max}},$$

and the «mixing» L is an implicit part of the cascade transformation

$$\mathsf{Csc}(K_{\mathsf{Csc}}, M') = \mathsf{g}(\ldots\mathsf{g}(\mathsf{g}(K_{\mathsf{Csc}} \oplus \Delta_0, m_1) \oplus \Delta_1, m_2)\ldots \oplus \widetilde{\Delta}_l, m_{l+1}),$$

where $K_{\mathsf{Csc}} = \mathsf{g}(IV, \overline{K})$ and $m_{l+1} = \mathsf{bin}(L)$.

The pad is injective, and hence if $\mathsf{KH}^{(1)}$ is secure with arbitrary block-length inputs, then $\mathsf{KH}^{(0)}$ is also an equally good PRF with $M \in V^*$.

Step 2. We replace $\mathsf{g}^{\triangledown}_K = \mathsf{g}(\cdot, \overline{K} \boxplus \cdot)$ (the first and last compression functions) with a family of true random functions $\mathsf{f}_{\overline{K}\boxplus\cdot}(\cdot)$ and obtain $\mathsf{KH}^{(2)}$.

Algorithm $\mathcal{A}$, which distinguishes $\mathsf{KH}^{(2)}$ from $\mathsf{KH}^{(1)}$, can be used to attack $\mathsf{g}_K^{\triangledown}$ in the model $PRF\text{-}RKA_{\boxplus}$. The corresponding algorithm $\mathcal{B}_{RKA}$ works as follows. To process requests from $\mathcal{A}$, one preparatory query $(IV, 0)$ to the oracle $\mathsf{g}(\cdot, \overline{K} \boxplus \cdot)$ is required. So, $\mathcal{B}_{RKA}$ obtains K_{Csc}. Each query $M \in (V^n)^{\leq l_{\max}}$ requires from $\mathcal{B}_{RKA}$ no more than $l_{\max}$ computations and one related-key query $(\mathsf{Csc}(K_{\mathsf{Csc}}, M), \sigma = \mathsf{sum}'_{\boxplus}(M))$. The result of work $\mathcal{A}$ is equal to the result of work $\mathcal{B}_{RKA}$ and the query complexity is $q_{\mathcal{B}} = 1 + q_{\mathcal{A}}$.

$$\Pr\left(\mathcal{A}^{\mathsf{KH}^{(1)}(\cdot)} \Rightarrow 1\right) - \Pr\left(\mathcal{A}^{\mathsf{KH}^{(2)}(\cdot)} \Rightarrow 1\right) \leq \mathrm{Adv}_{\mathsf{g}^{\triangledown}}^{PRF-RKA_{\boxplus}}(\mathcal{B}_{RKA}).$$

Step 3. The essence of the step is contained in the following statement.

Lemma. *The cascade* $\mathsf{Csc}(K_{\mathsf{Csc}}, M)$, $M \in (V^n)^{\leq l_{\max}}$ *is itself PRF-secure provided that* $\mathsf{g}^{\triangleright}$ *is secure in the* $PRF\text{-}RKA_{\oplus}$ *model*

$$\mathrm{Adv}_{\mathsf{Csc}}^{PRF}(t, q, l_{\max}) \leq q \cdot l_{\max} \cdot \mathrm{Adv}_{\mathsf{g}^{\triangleright}}^{PRF\text{-}RKA_{\oplus}}(t', q),$$

where $t' = t + O(q \cdot l_{\max})$.

The inequality presented above is similar to [9, Theorem 3.1] on the PRF-security of a «plain» cascade pCsc (i.e. without addition with Δ_0,...,$\widetilde{\Delta}_l$). The differences are as follows: the relevant threat model for $\mathsf{g}^{\triangleright}$ has been changed from PRF to $PRF\text{-}RKA_{\oplus}$; the sequence $\Delta_0, ..., \Delta_{l-1}, \widetilde{\Delta}_l$ is used; the prefix-free restriction is not imposed on the adversary's queries.

Obviously, the cascade isn't secure if the adversary can predict some output for non-queried input. If the value $\mathsf{pCsc}(K, M)$ is known for some message M, then $\mathsf{pCsc}(K, M||p)$ can also be easily computed for any block p (length extension attack). Hence, the PRF-security of pCsc is proved only for the case when none of the queried messages could be a prefix of any other.

Our situation is different. The value $\mathsf{Csc}(K, m_1|| \ldots ||m_{l+1})$ does not give a direct opportunity to compute $\mathsf{Csc}(K, m_1|| \ldots ||m_{l+1}||p)$ due to the $\Delta_l \neq \widetilde{\Delta}_l$. If m_{l+1} is the last block, then $\mathsf{g}(K_{\mathsf{g}} \oplus \widetilde{\Delta}_l, m_{l+1})$ is computed, otherwise we have $\mathsf{g}(K_{\mathsf{g}} \oplus \Delta_l, m_{l+1})$, where K_{g} is some intermediate state. The calculation is performed using related keys, and the relation is equal to $\phi_l = \widetilde{\Delta}_l \oplus \Delta_l$. We formalize this intuition using the $PRF\text{-}RKA_{\oplus}$ notion and give the proof in Appendix B.

Thus, the last call of the compression function with checksum mixing *is not necessary* to ensure the security of Streebog-K (of course, under the two assumptions about security of g). However, the presence of the checksum affects the resistance to specific key-recovery «beyound the bound» attacks.

Step 4. Consider a special threat model called $FINAL$

$$\mathrm{Adv}_{\mathsf{Csc}}^{FINAL}(\mathcal{A}) = \Pr\Big(K_{\mathsf{Csc}} \xleftarrow{R} V^n,\ \mathcal{A} \Rightarrow (M_1, \dots, M_q),$$
$$\exists i,j:\ \mathsf{Csc}(K_{\mathsf{Csc}}, M_i) = \mathsf{Csc}(K_{\mathsf{Csc}}, M_j),\ M_i \neq M_j \text{ OR}$$
$$\exists i: \mathsf{Csc}(K_{\mathsf{Csc}}, M_i) = IV\Big).$$

An adversary $\mathcal{A}$ which is effective in this model allows to construct the algorithm $\mathcal{B}_{FINAL}$ attacking Csc in the PRF model. $\mathcal{B}_{FINAL}$ runs the algorithm $\mathcal{A}$ and obtains q different messages $(M_1, \dots, M_q)$. For each message $\mathcal{B}_{FINAL}$ requests from its oracle the value $Y_i = \mathsf{F}(M_i)$ (resp. $Y_i = \mathsf{Csc}(K_{\mathsf{Csc}}, M_i)$). If $\mathcal{B}_{FINAL}$ obtains the collision or the value IV among $(Y_1, ..., Y_q)$ then the result is 1, otherwise 0. Hence

$$\Pr(\mathcal{B}_{FINAL}^{\mathsf{Csc}(K_{\mathsf{Csc}},\cdot)} \Rightarrow 1) = p_0 \geq \mathrm{Adv}_{\mathsf{Csc}}^{FINAL}(\mathcal{A}),$$
$$\Pr(\mathcal{B}_{FINAL}^{\mathsf{F}(\cdot)} \Rightarrow 1) = p_1 \leq \frac{q\cdot(q-1)}{2}\frac{1}{2^n} + \frac{q}{2^n},$$
$$\mathrm{Adv}_{\mathsf{Csc}}^{PRF}(\mathcal{B}_{FINAL}) = p_0 - p_1 \geq \mathrm{Adv}_{\mathsf{Csc}}^{FINAL}(\mathcal{A}) - \left(\frac{q\cdot(q-1)}{2}\frac{1}{2^n} + \frac{q}{2^n}\right),$$
$$\mathrm{Adv}_{\mathsf{Csc}}^{FINAL}(\mathcal{A}) \leq \mathrm{Adv}_{\mathsf{Csc}}^{PRF}(\mathcal{B}_{FINAL}) + \frac{q^2+q}{2^{n+1}}.$$

The last call of the compression function in $\mathsf{Streebog}$ «mixes» checksum with the state. At the second step, this transformation was replaced by a family of random functions $\mathsf{f}_{\overline{K}\boxplus\sigma}(\cdot) \xleftarrow{\mathrm{R}} \mathrm{Func}(V^n, V^n)$, $\forall \sigma \in V^n$. One query has already been made $K_{\mathsf{Csc}} = \mathsf{f}_{\overline{K}\boxplus 0}(IV)$.

The query M_i from $\mathcal{A}$ produces the pair of values

$$(Y_i, \sigma_i) = (\mathsf{Csc}(K_{\mathsf{Csc}}, M_i), \mathsf{sum}'_{\boxplus}(M_i))$$

If there are no collisions $(Y_i, \sigma_i) \neq (Y_j, \sigma_j)$ for all $i \neq j$ and for all i: $(Y_i, \sigma_i) \neq (IV, 0)$, then $\mathsf{f}_{\overline{K}\boxplus\cdot}(\cdot)$ is not requested twice with the same query. Thus, the result is indistinguishable from a random function.

The transformation $\mathsf{KH}^{(2)}$ is represented as follows.

Initialization: $K \xleftarrow{\mathrm{R}} V^k$; $H'_0, H'_1, \dots, H'_q \xleftarrow{\mathrm{R}} V^n$; $K_{\mathsf{Csc}} = H'_0 = \mathsf{f}_{\overline{K}\boxplus 0}(IV)$;

On query M_i, $i = 1, ..., q$ compute:

- $Y_i = \mathsf{Csc}(K_{\mathsf{Csc}}, M_i)$; $H_i = H'_i$; $\sigma_i = \mathsf{sum}'_{\boxplus}(M_i)$;
- (*) **if** $(Y_i, \sigma_i) = (Y_j, \sigma_j)$ for some $j < i$ **then** $H_i = H_j$;
- (*) **if** $(Y_i, \sigma_i) = (IV, 0)$ **then** $H_i = K_{\mathsf{Csc}}$;
- **return** H_i.

If rows marked with (*) are not executed, then the result is indistinguishable from a random function. Delete these rows and obtain $\mathsf{KH}^{(3)}$.

The probability of the conditions (*) being true does not exceed the probability of a successful attack in the $FINAL$ model on Csc (if we remove checksums, then it is essentially the same thing). Hence, by «fundamental game-playing lemma»

$$\Pr\left(\mathcal{A}^{\mathsf{KH}^{(3)}(\cdot)} \Rightarrow 1\right) - \Pr\left(\mathcal{A}^{\mathsf{KH}^{(2)}(\cdot)} \Rightarrow 1\right) \leq \mathrm{Adv}^{PRF}_{\mathsf{Csc}}(\mathcal{B}_{FINAL}) + \frac{q^2+q}{2^{n+1}}.$$

The set of transitions presented leads to the following theorem.

Theorem (PRF-security of Streebog-K). *For any adversary $\mathcal{A}$ with time complexity at most t that makes q queries, where the maximal message length is at most $(l_{\max}-1)$ blocks, there exist the adversaries $\mathcal{B}'$ and $\mathcal{B}''$ such that*

$$\mathrm{Adv}^{PRF}_{\mathsf{KH}}(\mathcal{A}) \leq \mathrm{Adv}^{PRF-RKA_{\boxplus}}_{\mathsf{g}^{\triangledown}}(\mathcal{B}') + q \cdot l_{\max} \cdot \mathrm{Adv}^{PRF-RKA_{\oplus}}_{\mathsf{g}^{\triangleright}}(\mathcal{B}'') + \frac{q^2+q}{2^{n+1}}.$$

The query complexity of $\mathcal{B}'$ and $\mathcal{B}''$ is $q+1$ and q correspondingly. The time complexity of both adversaries is $t' = t + O(ql_{\max})$.

Assuming $t \gg q \cdot l_{\max}$ and with the estimates of $\mathrm{Adv}^{PRF-RKA_{\boxplus}}_{\mathsf{g}^{\triangledown}}$ and $\mathrm{Adv}^{PRF-RKA_{\oplus}}_{\mathsf{g}^{\triangleright}}$ based on generic attacks

$$\mathrm{Adv}^{PRF}_{\mathsf{KH}}(t, q, l_{\max}) \lessapprox \frac{t \cdot q}{2^{k-1}} + \frac{t \cdot q \cdot l_{\max}}{2^{n-1}} + \frac{q^3 \cdot l_{\max}}{2^n}.$$

It should be noted that the bound presented in the theorem almost coincides with the corresponding one in HMAC and can be considered tight in some sense (see, for example [15]). At the same time, the approximate estimate, which was given for illustrative purposes, is not accurate and significantly exaggerates the capabilities of the adversary. Despite this, Streebog-K can be used in practice *without any restrictions* on the amount of data processed. Of course, the presented estimates do not consider threats that are outside the formal model, e.g. side-channel attacks and others.

For example, let Streebog-K be used as MAC with 256-bit key. The output is truncated to $\tau = 64$ bits, $q = 2^{48}$ messages are processed with one key, each message has a length of no more than $l_{\max} = 2^{64}$ blocks. The computing power of the adversary is about $t = 2^{128}$ operations. Hence, the probability of creating a forgery in one attempt is bounded by [12, Proposition 7.3] (SUF – Strong UnForgeablility)

$$\mathrm{Adv}^{SUF}_{\mathsf{KH}}(t, q, l_{\max}) \leq \mathrm{Adv}^{PRF}_{\mathsf{KH}}(t, q, l_{\max}) + \frac{1}{2^{\tau}} \lessapprox 2^{-63},$$

and the numerical value is close to the ideal $2^{-\tau} = 2^{-64}$.

6 Beyond the bound attacks

To complete the description of the properties and features of Streebog-K, we briefly present two attacks on it. Once again, we note that attacks have a significant probability of success only if the amount of material and computing resources of the adversary is greater than allowed according to the provable security bounds.

6.1 Existential forgery

The attack is carried out under the conditions of the adaptively chosen messages.

The set of l-block messages $M_i = \mathsf{bin}(i)||\mathsf{bin}(2^n - i)||C$, $i = 1, ..., q$ is prepared, where C contains arbitrary blocks. The checksums of all messages are the same $\mathsf{sum}_{\boxplus}(M_i) = \mathsf{sum}_{\boxplus}(M_j)$, $1 \le i, j \le q$.

The oracle is queried for the values of $H_i = \mathsf{KH}(K, M_i)$.

For simplicity, we assume that after processing of the two first blocks $\mathsf{bin}(i)||\mathsf{bin}(2^n - i)$, all intermediate states are different. Further transformations will be *identical* for each message.

Assuming that when processing the j-th block ($j = 3, \ldots, l$), a random mapping is applied to each intermediate state in parallel, the probability of a collision for an arbitrary pair is estimated by $\Pr(H_i = H_j, i \ne j) = \Theta\left(l \cdot 2^{-n}\right)$, see [23, Lemma 1]. Then the probability of a collision among q messages $\Pr(\exists i, j : H_i = H_j, i \ne j) = \Theta\left(q^2 \cdot l \cdot 2^{-n}\right)$.

A collision generated by a pair of messages M_i, M_j most likely occurs when processing one of the message blocks, and not when finalizing. Hence, we have

$$\mathsf{KH}(K, M_i||P) = \mathsf{KH}(K, M_j||P),\ P \in V^n.$$

The adversary uses the possibility of adaptive setting, obtains $H_{q+1} = \mathsf{KH}(K, M_i||P)$ and creates the forgery $M_j||P$ with the code H_{q+1}.

The attack is the same for both Streebog-K and HMAC-Streebog.

6.2 Key recovery

In [23], among other things, the key-recovery attack against HMAC-Streebog (with 512-bit key) was presented. The time complexity is at least $t = 2^{419}$ operations.

The attack consists of two phases. Both of them have almost the same time complexity.

The first phase is the state-recovery attack. This method is *generic* for HMAC with HAIFA-like [21] hash function. The *optimal* time complexity is about $t = 2^{419}$ operations. The oracle is queried about $q = 2^{358}$ times. The length of each query is at least $l = 2^{51}$ blocks. Other values of the q and l will result in more time complexity.

The target of the second phase is the secret key. This part of the attack can only be applied to HMAC-Streebog and similar cryptoalgorithms.

The state recovery attack can be used against Streebog-K without any modification. So, we omit its description and refer to [23]. As a result of the first phase, the adversary obtains the l-block message M and the corresponding secret state x (after processing the message and before finalization).

Key recovery phase is much easier for Streebog-K.

The adversary constructs 2^u-collision starting with the state x (the time complexity is about $u \cdot 2^{n/2}$ operations [20]). The value of the multicollision is $x^* = \mathsf{Csc}(K_{\mathsf{Csc}}, M||P_i||\mathsf{10..0}||\mathsf{bin}(L))$ and P_i contains exactly u blocks, $i = 1, ..., 2^u$. By queries to the oracle the values $H_i = \mathsf{KH}(\overline{K}, M||P_i)$ are collected and $\mathsf{g}(x^*, \overline{K} \boxplus \sigma_i) = H_i$, where $\sigma_i = \mathsf{sum}_{\boxplus}(M||P_i||\mathsf{10..0})$ (see fig. 3). We assume that almost all σ_i are different and the same is true for H_i.

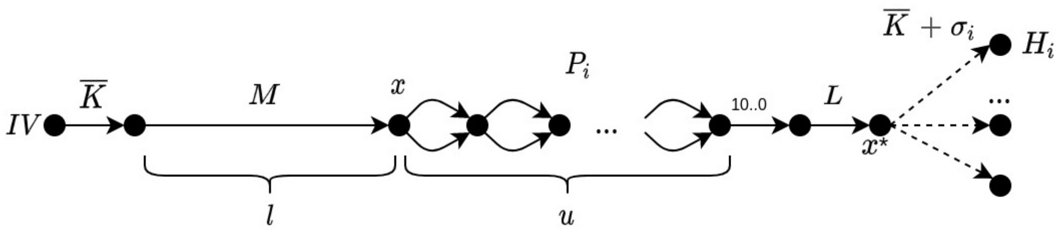

Figure 3: Key recovery attack (with known state x).

Thus, we need to guess $z_i = \overline{K} \boxplus \sigma_i$ for some i. We compute $\mathsf{g}(x^*, \widetilde{z}_j) = \widetilde{H}_j$ and check the match $\widetilde{H}_j = H_i$, $j = 1, ..., 2^v$. If $\widetilde{H}_j = H_i$ is true then $\overline{K} \boxplus \sigma_i = \widetilde{z}_j$ can also be true with high probability. If $2^u \cdot 2^v = 2^n$, we expect one true match to be found.

The time complexity of the key recovery phase is $t \approx l \cdot 2^u + u \cdot 2^{n/2} + 2^v$ and with $u = 230$, $l = 2^{51}$, $t \approx 2^{283}$, the query complexity is $q = 2^u$. Consequently, the complexity of the first phase is much greater than the second.

Thus, Streebog-K does not provide «512-bit security» in the sense of resistance to the key recovery. This is also true for HMAC-Streebog. We suggest using 256-bit keys in Streebog-K.

7 HMAC, S3G and GOST94

The obtained security proof for Streebog-K can be used with some modifications for a number of similar cryptoalgorithms.

HMAC-Streebog uses the key *four* times. The relation between keys is defined by two operations simultaneously. In the second step of the proof, the *stronger* model is used instead of $PRF\text{-}RKA_{\boxplus}$ (problem 2)

$$\mathrm{Adv}_{\mathsf{g}^{\triangledown}}^{PRF-RKA_{\oplus,\boxplus}}(\mathcal{A}) = \Pr\left(K \xleftarrow{R} V^k; \mathcal{A}^{\mathsf{g}(\cdot,(\overline{K}\oplus\cdot)\boxplus\cdot)} \Rightarrow 1\right) -$$
$$- \Pr\left(K \xleftarrow{R} V^k; \mathsf{f}_i \xleftarrow{R} \mathrm{Func}(V^n, V^n),\ \forall i \in V^n; \mathcal{A}^{\mathsf{f}_{(\overline{K}\oplus\cdot)\boxplus\cdot}(\cdot)} \Rightarrow 1\right),$$

the query is (m, ϕ, σ), the response is $y = \mathsf{g}(m, (\overline{K} \oplus \phi) \boxplus \sigma)$. The heuristic estimate of complexity for $PRF\text{-}RKA_{\oplus,\boxplus}$ is the same as for $PRF\text{-}RKA_{\boxplus}$. Problem 1 and the rest of the steps remain unchanged, but one more step is added. The collision after the first call of the hash function $\mathsf{H}((\overline{K} \oplus ipad)||M)$ is taken into account, as well as the collision between the last related key and the other three. The result is the following theorem (the proof is presented in Appendix).

Theorem (PRF-security of HMAC-Streebog). *For any adversary $\mathcal{A}$ with time complexity at most t that makes q queries, where the maximal message length is at most $(l_{\max} - 1)$ blocks, there exist adversaries $\mathcal{B}'$ and $\mathcal{B}''$ such that*

$$\mathrm{Adv}_{\mathsf{HMAC\text{-}Streebog}}^{PRF}(\mathcal{A}) \le \mathrm{Adv}_{\mathsf{g}^{\triangledown}}^{PRF-RKA_{\oplus,\boxplus}}(\mathcal{B}')+$$
$$+ q \cdot l_{\max} \cdot \mathrm{Adv}_{\mathsf{g}^{\triangleright}}^{PRF-RKA_{\oplus}}(\mathcal{B}'') + \frac{q^2+q}{2^{n+1}} + \frac{3(q^2+q)}{2^{\tau+1}},$$

where $\tau \in \{256, 512\}$ is the bit length of the output. The query complexity of $\mathcal{B}'$ and $\mathcal{B}''$ is $2 + 2 \cdot q$ and q correspondingly. The time complexity of both adversaries is $t' = t + O(ql_{\max})$.

Cryptoalgorithm S3G [4] is defined as $\mathsf{S3G}(K, M) = \mathsf{H}(K||M)$, but the k-bit key is not padded to 512 bits, $k \in \{128, 256\}$. The number of calls to g is always the same for the selected k. Yet another variant of the compression function is defined as $\mathsf{g}_K^{\triangledown\triangledown}(m, m') = \mathsf{g}(m, K||m')$, $m' \in V^{n-k}$ with the corresponding threat model

$$\mathrm{Adv}_{\mathsf{g}^{\triangledown\triangledown}}^{PRF-RKA_{\boxplus,||}}(\mathcal{A}) = \Pr\left(K \xleftarrow{R} V^k; \mathcal{A}^{\mathsf{g}(\cdot,(K||\cdot)\boxplus\cdot)} \Rightarrow 1\right) -$$
$$- \Pr\left(K \xleftarrow{R} V^k; \mathsf{f}_i \xleftarrow{R} \mathrm{Func}(V^n, V^n),\ \forall i \in V^n; \mathcal{A}^{\mathsf{f}_{(K||\cdot)\boxplus\cdot}(\cdot)} \Rightarrow 1\right),$$

the query is (m, m', σ), the response is $y = \mathsf{g}\,(m, (K||m') \boxplus \sigma)$. The second step of the proof changes accordingly. In the third step, the tree does not have a single root, the number of roots is arbitrary from 1 to q. The fourth step of the proof does not require significant changes.

Hash function $\mathsf{GOST94}$ [2] is based on the «plain» Merkle-Damgård scheme (the state and the message size $n = 256$ bit) with one exception: the last call of the compression function g_{94} «mixes» checksum of the message (modulo 2^n) to the state. The first step of the security proof $\mathsf{HMAC\text{-}GOST94}$ should take into account changes in the padding. The second step uses the same reduction as for $\mathsf{HMAC\text{-}Streebog}$ ($\mathsf{g}_{94}^{\triangledown}$ must be secure in the $PRF\text{-}RKA_{\oplus,\boxplus}$ model). The third step is simply using the previously known result [9, Th. 3.1] ($\mathsf{g}_{94}^{\triangleright}$ must be a secure PRF). At the fourth step we use a well-known «extension trick» as in the proof of HMAC [14, Sec. 7]. The last step is the same as for $\mathsf{HMAC\text{-}Streebog}$. However, it should be noted that the security of g_{94}, as far as we know, was not examined in the PRF model, unlike g [24, 25]. Therefore, in the case of $\mathsf{HMAC\text{-}GOST94}$, we cannot say without a doubt that the basic problems are really computationally hard.

8 Conclusion

The paper presents «$\mathsf{Streebog\text{-}K}$» («Keyed Streebog»)

$$\mathsf{KH}(K, M) = \mathsf{H}(\overline{K}||M), \; \overline{K} = K||0...0,$$

based on keyless hash function $\mathsf{Streebog}$. The proposed solution has almost the same cryptographic strength as $\mathsf{HMAC\text{-}Streebog}$. This is true both from the provable security point of view, and with regard to the applicability of attacks. At the same time, the speed is doubled when processing short texts.

The suggested proof shows that $\mathsf{Streebog\text{-}K}$ is a pseudorandom function (PRF) and, therefore, a secure message authentification code (MAC). The security is reduced to the resistance of the underlying compression function to the related key attacks (PRF-RKA). The existing results indicate that the compression function is indeed secure in the relevant threat models.

The obtained results can be slightly modified to create security proofs of $\mathsf{HMAC\text{-}Streebog}$, $\mathsf{HMAC\text{-}GOST94}$, $\mathsf{S3G}$ and similar cryptoalgorithms. Such changes were also briefly listed.

We also propose two open problems to consider:

1) Is it possible to replace problem 1 ($PRF\text{-}RKA_{\oplus}$) in the reduction with the simple PRF model?

2) Is there an attack in any model that would be more effective against $\mathsf{Streebog\text{-}K}$ than against $\mathsf{HMAC\text{-}Streebog}$?

9 Acknowledgements

The author sincerely thanks Evgeny Alekseev, Liliya Akhmetzyanova and Alexandra Babueva for the inspiration, motivation and useful discussions. The quality of the paper has been significantly improved thanks to the great help of Andrey Scherbachenko. Special thanks to the anonymous reviewer(s) for the thorough verification of the result and a lot of detailed comments and suggestions.

References

[1] *GOST R 34.11-2012 – National standard of the Russian Federation – Information technology – Cryptographic data security – Hash function*, 2012.

[2] *GOST R 34.11-94 – National standard of the Russian Federation – Information technology – Cryptographic data security – Hash function*, 1994.

[3] *R 50.1.113-2016 – Information technology – Cryptographic data security – Cryptographic algorithms accompanying the use of electronic digital signature algorithms and hash functions*, 2016.

[4] *R 1323565.1.003-2017 – Information technology – Cryptographic data security – Cryptographic algorithms for generating encryption keys and authentication vectors intended for implementation in hardware trust modules for use in mobile communication*, 2017.

[5] Grebnev S., Dmukh A., Dygin D., Matyukhin D., Rudskoy V., Shishkin V., "Asymmetrical Reply to SHA-3: Russian Hash Function Draft Standard", *CTCrypt 2012*, 2012.

[6] Smyshlyaev S., Alekseev E., Oshkin I., Popov V., Leontiev S., Podobaev V., Belyavsky D., "RFC 7836 - Guidelines on the Cryptographic Algorithms to Accompany the Usage of Standards GOST R 34.10-2012 and GOST R 34.11-2012", March 2016.

[7] Damgård I., "A design principle for hash functions", CRYPTO 1989, Lect. Notes Comput. Sci., **435**, 1990, 416–427.

[8] Merkle R., "One way wash functions and DES", CRYPTO 1989, Lect. Notes Comput. Sci., **435**, 1990, 428–446.

[9] Bellare M., Canetti R., Krawczyk H., "Pseudorandom Functions Revisited: The Cascade Construction and its Concrete Security", 37th FOCS, *IEEE*, 1996, 514–523.

[10] Bellare M., Canetti R., Krawczyk H., "Keying Hash Functions for Message Authentication", Crypto'96, Lect. Notes Comput. Sci., **1109**, 1996, 1–15.

[11] Preneel B., Paul C. van Oorschot, "On the Security of Iterated Message Authentication Codes", *IEEE Transactions on Information Theory*, **45** (1999), 188–199.

[12] Bellare M., Goldreich O., Mityagin A., "The power of verification queries in message authentication and authenticated encryption", *Cryptology ePrint Archive: Report 2004/304*, 2004.

[13] Koblitz N., Menezes A., "Another look at HMAC", *J. Math. Cryptol.*, **7:3** (2013), 225–251.

[14] Bellare M., "New proofs for NMAC and HMAC: security without collision-resistance", CRYPTO 2006, Lect. Notes Comput. Sci., **4117**, April 2014, 602–619.

[15] Gaži P., Pietrzak K., Rybár M., "The Exact PRF-Security of NMAC and HMAC", CRYPTO 2014, Lect. Notes Comput. Sci., **8616**, August 2014, 113–130.

[16] Alekseev E.K., Oshkin I.B., Popov V.O., Smyshlyaev S.V., "On the cryptographic properties of algorithms accompanying the applications of standards GOST R 34.11-2012 and GOST R 34.10-2012", *Mat. Vopr. Kriptogr.*, **7**:1 (2016), 5–38.

[17] Nandi M., "A New and Improved Reduction Proof of Cascade PRF", *Cryptology ePrint Archive: Report 2021/097*, 2021.

[18] Bellare M., Rogaway P., *Introduction to Modern Cryptography*, 2005.

[19] Goldreich O., *Foundations of Cryptography. Basic Tools*, 2004.

[20] Joux A., "Multicollisions in Iterated Hash Functions. Application to Cascaded Constructions", CRYPTO 2004, Lect. Notes Comput. Sci., **3152**, 2004, 306–316.

[21] Biham E., Dunkelman O., "A Framework for Iterative Hash Functions (HAIFA)", *Cryptology ePrint Archive, Report 2007/278*, 2007.

[22] Guo J., Jean J., Leurent G., Peyrin T., Wang L., "The usage of counter revisited: second-preimage attack on new Russian standardized hash function", SAC 2014, Lect. Notes Comput. Sci., **8781**, 2014, 195–211.

[23] Dinur I., Leurent G., "Improved generic attacks against hash-based MACs and HAIFA", CRYPTO 2014, Lect. Notes Comput. Sci., **8616**, 2014, 149–168.

[24] Abdelkhalek A., AlTawy R., Youssef A. M., "Impossible differential properties of reduced round Streebog", C2SI 2015, Lect. Notes Comput. Sci., **9084**, 2015, 274–286.

[25] Kiryukhin V., "Streebog compression function as PRF in secret-key settings", CTCrypt 2021, *Mat. Vopr. Kriptogr.*, **13**:2 (2022), 99–116, https://eprint.iacr.org/2022/118.pdf.

[26] Kiryukhin V., "Related-key attacks on the compression function of Streebog", *CTCrypt 2022, June 6-9, 2022, Novosibirsk, Russia*, https://eprint.iacr.org/2022/970.pdf.

[27] AlTawy R., Youssef A. M., "Preimage attacks on reduced-round Stribog", AFRICACRYPT 2014, Lect. Notes Comput. Sci., **8469**, 2014, 109–125.

[28] AlTawy R., Kircanski A., Youssef A. M., "Rebound attacks on Stribog", ICISC 2013, Lect. Notes Comput. Sci., **8565**, 2014, 175–188.

[29] Lin D., Xu S., Yung M., "Cryptanalysis of the round-reduced GOST hash function", Inscrypt 2013, Lect. Notes Comput. Sci., **8567**, 2014, 309–322.

[30] Ma B., Li B., Hao R., Li X., "Improved cryptanalysis on reduced-round GOST and Whirlpool hash function", ACNS 2014, Lect. Notes Comput. Sci., **8479**, 2014, 289–307.

[31] Wang Z., Yu H., Wang X., "Cryptanalysis of GOST R Hash Function", *Information Processing Letters*, **114** (2014), 655–662.

[32] Kölbl S., Rechberger C., "Practical attacks on AES-like cryptographic hash functions", LATINCRYPT 2014, Lect. Notes Comput. Sci., **8895**, 2014, 259–273.

[33] Ma B., Li B., Hao R., Li X., "Improved (pseudo) preimage attacks on reduced-round GOST and Grøstl-256 and studies on several truncation patterns for AES-like compression functions", IWSEC 2015, Lect. Notes Comput. Sci., **9241**, 2015, 79–96.

[34] Hua J., Dong X., Sun S., Zhang Z., Hu L., Wang X., "Improved MITM Cryptanalysis on Streebog", *Cryptology ePrint Archive, Paper 2022/568*, 2022.

A Detailed pictures

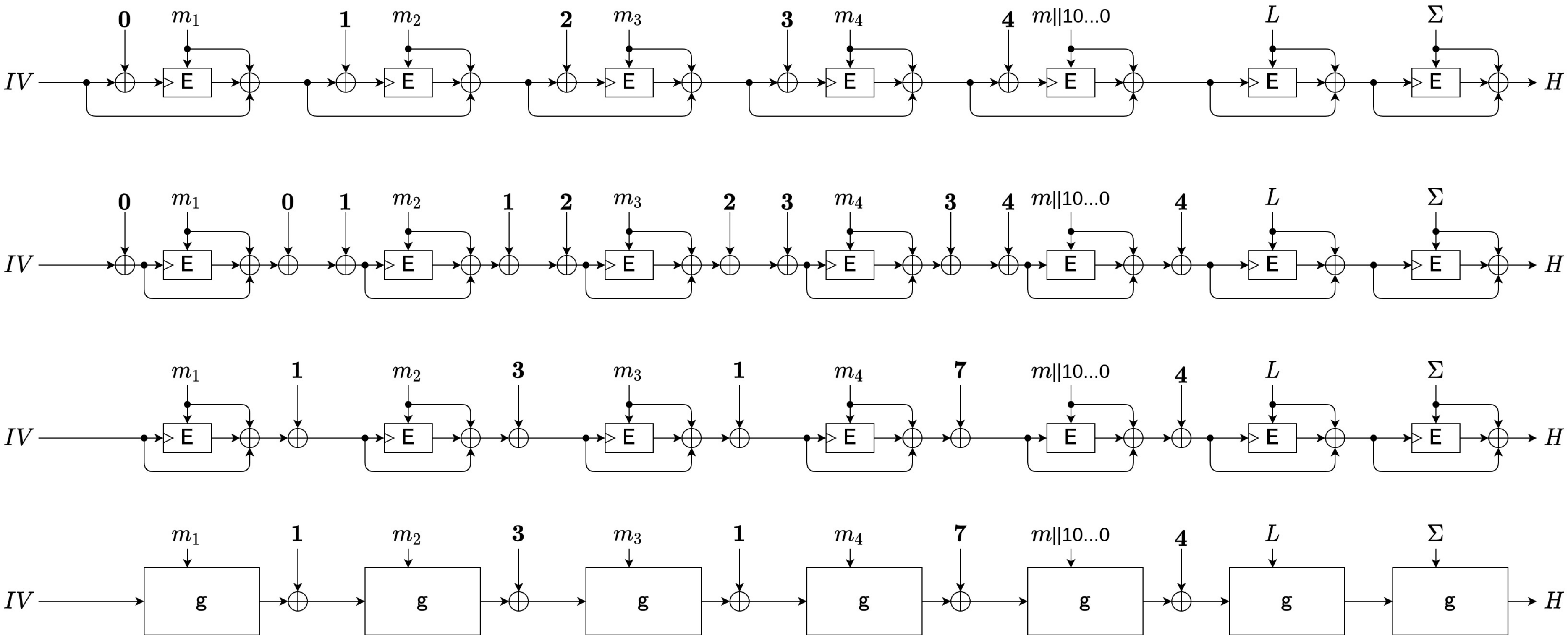

Figure 4: The equivalent representation of Streebog.

B Proof of the lemma

Lemma. *The cascade*

$$\mathsf{Csc}(K_{\mathsf{Csc}}, M) = \mathsf{g}(\ldots \mathsf{g}(K_{\mathsf{Csc}} \oplus \Delta_0, m_1) \ldots \oplus \widetilde{\Delta}_l, m_{l+1}),$$

where $M = m_1||m_2||...||m_{l+1} \in (V^n)^{\leq l_{\max}}$, $1 \leq l+1 \leq l_{\max}$, *is PRF-secure provided that* $\mathsf{g}^{\triangleright}$ *is secure in the PRF-RKA*$_{\oplus}$ *model*

$$\mathrm{Adv}_{\mathsf{Csc}}^{PRF}(t, q, l_{\max}) \leq q \cdot l_{\max} \cdot \mathrm{Adv}_{\mathsf{g}^{\triangleright}}^{PRF\text{-}RKA_{\oplus}}(t', q),$$

where $t' = t + O(q \cdot l_{\max})$.

Proof. Let's imagine queries to the cascade in the form of a tree:
the root v_0 is K_{Csc}; the nodes v_i are the intermediate states; the results are stored in leaves. Each edge of the tree is labeled with the the block m_i from the message M

$$K_{\mathsf{Csc}} = v_0 \overset{m_1}{\to} v_1 \overset{m_2}{\to} v_2 \overset{m_3}{\to} v_3 \ldots \overset{m_{l+1}}{\to} v_{l+1}, \ 1 \leq l+1 \leq l_{\max}.$$

At each level (height) after processing all requests, there will be no more than q nodes.

Consider an arbitrary node v_i, which is essentially an intermediate secret key. If m_{i+1} is not the last, then $\mathsf{g}(v_i \oplus \Delta_i, m_{i+1}) = v_{i+1}$ is computed. If the block m'_{i+1} is the last, then $\widetilde{\Delta}_i$ is used $\mathsf{g}(v_i \oplus \widetilde{\Delta}_i, m'_{i+1}) = v'_{i+1}$. The first secret key is $K_{\mathsf{g}} = v_i \oplus \Delta_i$, the second one is $K'_{\mathsf{g}} = v_i \oplus \widetilde{\Delta}_i$. The relation between the keys is defined as

$$\phi_i = K_{\mathsf{g}} \oplus K'_{\mathsf{g}} = \Delta_i \oplus \widetilde{\Delta}_i = (\mathbf{i} \oplus (\mathbf{i} \boxplus 1)) \ \oplus \ \mathbf{i} = \mathbf{i} \boxplus 1, \ i = 0, \ldots, l_{\max}.$$

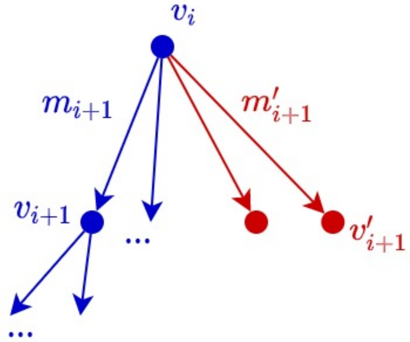

Figure 5: Node v_i of the tree. The internal (resp. external) edges and nodes are highlighted in blue (resp. red).

The adversary will never observe the values of the internal nodes of the tree. Hence, all adversary's queries will be *independent* of these values. For a «plain» cascade from [9], this was ensured by limiting the queries (none of the queried message could be a prefix of any other). In our case, the mentioned independence is provided essentially by two different functions that compute internal and external nodes, respectively.

We use a «hybrid argument» for tree levels (from 1 to $l_{\max}$) and for nodes of each level (from 1 to q).

Denote by Csc_i the cascade transformation starting from level i, $\mathsf{Csc}_0 = \mathsf{Csc}$. At level i, we define the hybrid game and the corresponding oracle $\mathcal{C}_i$ as follows:

– Initialization: $\mathsf{F} \xleftarrow{R} \mathrm{Func}\left((V^n)^i, V^n\right)$; $\mathsf{F}' \xleftarrow{R} \mathrm{Func}\left((V^n)^{\leq i}, V^n\right)$;

– On query $M = (m_1, ..., m_l) \in (V^n)^l$, $1 \leq l \leq l_{\max}$ from $\mathcal{A}$ compute:
if $l \leq i$ **then** $y = \mathsf{F}'(M)$; **return** y;
if $l > i$ **then** $M_{pre} = (m_1, \ldots, m_i)$; $M_{suff} = (m_{i+1}, \ldots, m_l)$;
return $y = \mathsf{Csc}_i(\mathsf{F}(M_{pre}), M_{suff})$.

All internal (resp. external) nodes of the tree are calculated using F (resp. F'). The oracle $\mathcal{C}_0(\cdot)$ is identical to the $\mathsf{Csc}(K_{\mathsf{Csc}}, \cdot)$. Indeed, if $i = 0$ then M_{pre} is an empty string, $M_{suff} = M$, $\mathsf{F}(M_{pre}) = K_{\mathsf{Csc}}$ and $y = \mathsf{Csc}_0(K_{\mathsf{Csc}}, M)$. The algorithm $\mathcal{C}_{l_{\max}}(\cdot)$ is essentially a random function

$$\mathrm{Adv}_{\mathsf{Csc}}^{PRF}(\mathcal{A}) = \Pr\left(\mathcal{A}^{\mathcal{C}_{l_{\max}}(\cdot)} \Rightarrow 1\right) - \Pr\left(\mathcal{A}^{\mathcal{C}_0(\cdot)} \Rightarrow 1\right).$$

Let $\mathcal{A}$ be able to effectively distinguish $\mathcal{C}_0(\cdot)$ and $\mathcal{C}_1(\cdot)$, then it is possible to construct $\mathcal{B}_0$ distinguishing the compression function from the random function in the $PRF\text{-}RKA_{\oplus}$ model. Really, for the one-block message $M = m_1$ algorithm $\mathcal{B}_0$ queries the value $\mathsf{f}'(m_1)$ (this is either the value of the second random function, or $\mathsf{g}(K \oplus \phi_1, m_1)$). For any multi-block query $M = (m_1, m_2, \ldots, m_l)$ received from $\mathcal{A}$, algorithm $\mathcal{B}_0$ asks its oracle for the value $\mathsf{f}(m_1)$ (resp. $\mathsf{g}(K, m_1)$). Recall that the secret random key K is essentially the value $K = K_{\mathsf{Csc}} \oplus \Delta_0$ (also distributed uniformly on V^n), and therefore, $\mathcal{B}_0$ correctly emulates the beginning of the cascade. Next, the value $\mathsf{Csc}_1(\mathsf{f}(m_1), (m_2, \ldots, m_l))$ is computed and sent to $\mathcal{A}$ without queries to the oracle. The result of $\mathcal{B}_0$ is equal to the result of $\mathcal{A}$ and

$$\Pr\left(\mathcal{A}^{\mathcal{C}_1(\cdot)} \Rightarrow 1\right) - \Pr\left(\mathcal{A}^{\mathcal{C}_0(\cdot)} \Rightarrow 1\right) \leq \mathrm{Adv}_{\mathsf{g}^{\triangleright}}^{PRF-RKA_{\oplus}}(\mathcal{B}_0).$$

Consider the general case. Let $\mathcal{A}$ be able to effectively distinguish $\mathcal{C}_i(\cdot)$ and $\mathcal{C}_{i-1}(\cdot)$ (figure 6).

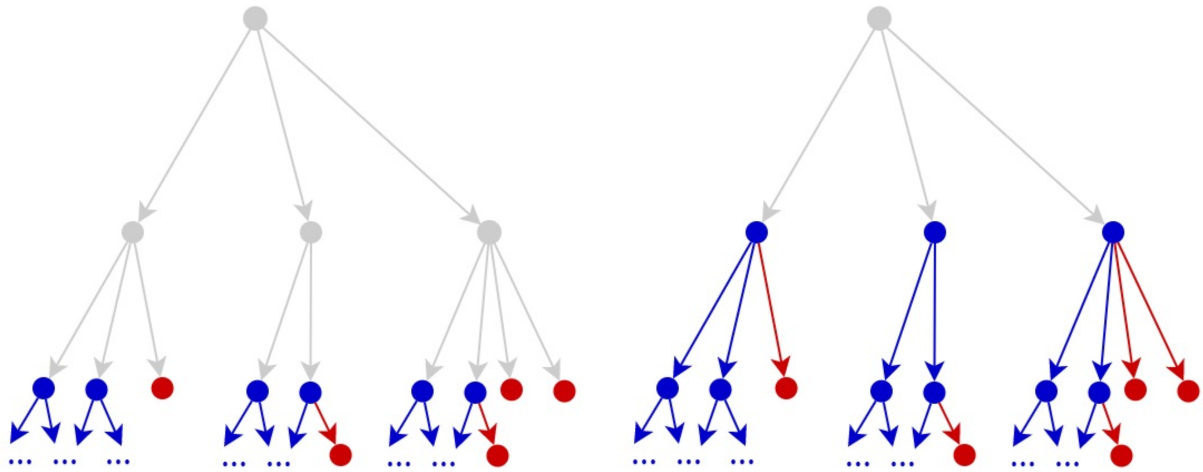

Figure 6: Trees formed by queries to $\mathcal{C}_i(\cdot)$ (left) and $\mathcal{C}_{i-1}(\cdot)$ (right).

We turn to the case of q parallel games in the PRF-$RKA_{\oplus}$. The corresponding adversary $\mathcal{B}_i$ has access to q pairs of oracles (q pairs of random functions $(\mathsf{f}_j(\cdot), \mathsf{f}'_j(\cdot))$ or q pairs of $(\mathsf{g}^{\triangleright}_{K_j}(\cdot), \mathsf{g}^{\triangleright}_{K_j \oplus \phi_i}(\cdot))$, $j = 1, .., q$). The query from $\mathcal{B}_i$ consists of $(j, b \in \{1, 2\}, m \in V^n)$, where b specifies the first or the second oracle of the j-th pair. Due to the independence of the pairs, the hybrid argument is straightforward, and we have

$$\Pr\left(\mathcal{B}_i^{\mathsf{f}_\cdot(\cdot),\mathsf{f}'_\cdot(\cdot)} \Rightarrow 1\right) - \Pr\left(\mathcal{B}_i^{\mathsf{g}^{\triangleright}_{K_\cdot}(\cdot),\mathsf{g}^{\triangleright}_{K_\cdot \oplus \phi_i}(\cdot)} \Rightarrow 1\right) \leq \sum_{j=1}^{q} \mathrm{Adv}_{\mathsf{g}^{\triangleright}}^{PRF-RKA_{\oplus}}(\mathcal{B}_{i,j}).$$

The value of $\phi_i = \mathbf{i} \boxplus 1$ is chosen equally by all PRF-$RKA_{\oplus}$-adversaries $\mathcal{B}_{i,j}$, $j = 1, ...q$. The algorithm of $\mathcal{B}_{i,j}$ is similar to that of $\mathcal{B}_0$. In fact, the latter processes queries that affect the root of the tree, and $\mathcal{B}_{i,j}$ uses messages that depend on the j-th node at depth i.

$M = (m_1, \ldots, m_l)$ is the query from $\mathcal{A}$ to the oracle $\mathcal{C}_i(\cdot)$ or $\mathcal{C}_{i-1}(\cdot)$. The algorithm $\mathcal{B}_i$ must perfectly simulate both of them:

- Initialization: $\mathrm{PrefixMap}[P] = \emptyset$, $\forall P \in (V^n)^{i-1}$; $\mathrm{max_j} = 1$;
- On query $M = (m_1, m_2, \ldots, m_l) \in (V^n)^l$, $1 \leq l \leq l_{\max}$ from $\mathcal{A}$ compute:
  - **if** $l < i$ **then return** $y \xleftarrow{\mathrm{R}} V^n$; (recall that there are no duplicate queries M);
  - **if** $\mathrm{PrefixMap}[(m_1, \ldots, m_{i-1})] = \emptyset$ **then**
    $\mathrm{PrefixMap}[(m_1, \ldots, m_{i-1})] = \mathrm{max_j}$;
    $\mathrm{max_j} = \mathrm{max_j} + 1$;
  - $j = \mathrm{PrefixMap}[(m_1, \ldots, m_{i-1})]$;

– **if** $l = i$ **then** $y = \mathsf{f}'_j(m_i)$; (resp. $y = \mathsf{g}^{\triangleright}_{K_j \oplus \phi_i}(m_i)$) by query $(j, 2, m_i)$; **return** y;
– **if** $l > i$ **then** $z = \mathsf{f}_j(m_i)$; (resp. $z = \mathsf{g}^{\triangleright}_{K_j}(m_i)$) by query $(j, 1, m_i)$; $M_{suff} = (m_{i+1}, \ldots, m_l)$; $y = \mathsf{Csc}_i(z, M_{suff})$; **return** y.

First of all, we note that requests shorter than l blocks always generate a random response (both $\mathcal{C}_i$ and $\mathcal{B}_i$). Different prefixes $(m_1, \ldots, m_{i-1})$ generate queries to different oracles. PrefixMap is used to store all queried prefixes $(m_1, ..., m_{i-1})$. Initially, there is not a single entry in PrefixMap. If the prefix has not been queried before, a new entry is created in PrefixMap, otherwise, the j corresponding to the prefix is extracted. After all interactions, we have $1 \leq \max_\mathrm{j} \leq q$. In other words, PrefixMap stores at least one and at most q elements. If $\max_\mathrm{j} = 1$, then all queries had the same prefix. If $\max_\mathrm{j} = q$, then the prefixes of all queries were different.

Let $\mathcal{B}_i$ interact with q pairs of random functions. Therefore, if $l = i$ then the response is really random (as $y = \mathsf{F}'(M)$ in the case of $\mathcal{C}_i$). If $l > i$ then it is also truly random (as the intermediate value $\mathsf{F}(M_{pre})$). Further computation of the cascade is identical in both cases. So, $\mathcal{B}_i$ simulates $\mathcal{C}_i(\cdot)$ for the adversary $\mathcal{A}$.

Let $\mathcal{B}_i$ interact with q pairs of the compression functions $(\mathsf{g}^{\triangleright}_{K_j}(\cdot), \mathsf{g}^{\triangleright}_{K_j \oplus \phi_i}(\cdot))$. Imagine that $M_{pre} = (m_1, \ldots, m_{i-1})$ and instead of requesting a random function $\mathsf{F}(M_{pre})$, we implicitly use secret keys from the j-th pair of oracles. Next, the computations $y = \mathsf{Csc}_i(\mathsf{g}(K_j, m_i), (m_{i+1}, \ldots, m_l))$ are equivalent to the Csc_{i-1} cascade, and the perfect simulation of $\mathcal{C}_{i-1}(\cdot)$ is also constructed.

Thus, we consistently replace $\mathcal{C}_{i-1}(\cdot)$ with $\mathcal{C}_i(\cdot)$ $(i = 1, ..., l_{\max})$, and summing up the advantages, we obtain the statement of the lemma

$$\mathrm{Adv}^{PRF}_{\mathsf{Csc}}(\mathcal{A}) \leq \sum_{i=1}^{l_{\max}} \left(\Pr\left(\mathcal{A}^{\mathcal{C}_i(\cdot)} \Rightarrow 1\right) - \Pr\left(\mathcal{A}^{\mathcal{C}_{i-1}(\cdot)} \Rightarrow 1\right)\right) \leq$$
$$\leq \sum_{i=1}^{l_{\max}} \sum_{j=1}^{q} \mathrm{Adv}^{PRF-RKA_{\oplus}}_{\mathsf{g}^{\triangleright}}(\mathcal{B}_{i,j}).$$

C Adaptation of the proof for HMAC-Streebog

Recall that the HMAC is represented as

$$\mathsf{HMAC}(K, M) = \mathsf{H}\left((\overline{K} \oplus opad)||\mathsf{H}(\overline{K} \oplus ipad||M)\right),$$

where $ipad, opad \in V^n$, $ipad \neq opad$, $\overline{K} = (K||0...0) \in V^n$. We consider the case when H is Streebog-256 or Streebog-512 (fig. 7 and 8).

Denote by $\tau \in \{256, 512\}$ the bit length of the hash function output. The intermediate output is $H^I = \mathsf{H}((\overline{K} \oplus ipad)||M) \in V^\tau$ and the keys for cascades are $K^I_{\mathsf{Csc}} = \mathsf{g}(IV, \overline{K} \oplus ipad)$, $K^O_{\mathsf{Csc}} = \mathsf{g}(IV, \overline{K} \oplus opad)$.

The proof of the PRF-security for HMAC-Streebog is similar to the corresponding one for Streebog-K. Next, we describe the changes.

The **first step** remains the same. We proceed to the analysis when the message consists of n-bit blocks ($\mathsf{HMAC}^{(1)}$).

In the **second step**, the *stronger* model is used instead of $PRF\text{-}RKA_{\boxplus}$ (problem 2)

$$\mathrm{Adv}^{PRF-RKA_{\oplus,\boxplus}}_{\mathsf{g}^{\triangledown}}(\mathcal{A}) = \Pr\left(K \xleftarrow{\mathrm{R}} V^k; \mathcal{A}^{\mathsf{g}(\cdot,(\overline{K}\oplus\cdot)\boxplus\cdot)} \Rightarrow 1\right) - \\ - \Pr\left(K \xleftarrow{\mathrm{R}} V^k; \mathsf{f}_i \xleftarrow{\mathrm{R}} \mathrm{Func}(V^n, V^n),\ \forall i \in V^n; \mathcal{A}^{\mathsf{f}_{(\overline{K}\oplus\cdot)\boxplus\cdot}(\cdot)} \Rightarrow 1\right),$$

the query is the triple (m, ϕ, σ), the response is $y = \mathsf{g}(m, (\overline{K} \oplus \phi) \boxplus \sigma)$.

We replace $\mathsf{g}^{\triangledown}_K = \mathsf{g}(\cdot, (\overline{K} \oplus \cdot) \boxplus \cdot)$ (the first and last compression functions in the inner and outer hash functions) with a family of true random functions $\mathsf{f}_{(\overline{K}\oplus\cdot)\boxplus\cdot}(\cdot)$ and obtain $\mathsf{HMAC}^{(2)}$.

Algorithm $\mathcal{A}$, which distinguishes $\mathsf{HMAC}^{(2)}$ from $\mathsf{HMAC}^{(1)}$, can be used to attack $\mathsf{g}^{\triangledown}_K$ in the model $PRF\text{-}RKA_{\oplus,\boxplus}$. The corresponding algorithm $\mathcal{B}_{RKA}$ works as follows. To process requests from $\mathcal{A}$, two preparatory queries $(IV, ipad, 0)$ and $(IV, opad, 0)$ to the oracle $\mathsf{g}(\cdot, (\overline{K} \oplus \cdot) \boxplus \cdot)$ are required (K^I_{Csc} and K^O_{Csc} are obtained). Each query $M \in (V^n)^{\leq l_{\max}}$ requires from $\mathcal{B}_{RKA}$ no more than $O(l_{\max})$ computations and two related-key queries

$$(\mathsf{Csc}(K^I_{\mathsf{Csc}}, M), ipad, \sigma^I = \mathsf{sum}'_{\boxplus}(M)),$$
$$(\mathsf{Csc}(K^O_{\mathsf{Csc}}, H^I||10...0||\mathsf{bin}(n + \tau)), opad, \sigma^O = \mathsf{sum}_{\boxplus}(H^I||10...0)).$$

The result of work $\mathcal{A}$ is equal to the result of work $\mathcal{B}_{RKA}$ and the query complexity is $q_{\mathcal{B}} = 2 + 2 \cdot q_{\mathcal{A}}$.

$$\Pr\left(\mathcal{A}^{\mathsf{HMAC}^{(1)}(\cdot)} \Rightarrow 1\right) - \Pr\left(\mathcal{A}^{\mathsf{HMAC}^{(2)}(\cdot)} \Rightarrow 1\right) \leq \mathrm{Adv}^{PRF-RKA_{\oplus,\boxplus}}_{\mathsf{g}^{\triangledown}}(\mathcal{B}_{RKA}).$$

The **third** and **fourth steps** also remain the same.

The **fifth step**.

As a result of four steps we construct $\mathsf{HMAC}^{(3)}$ as a truly random function generating H^I. The last compression function is defined in $\mathsf{HMAC}^{(3)}$ as a family of random functions $\mathsf{f}_{(\overline{K} \oplus opad) \boxplus \sigma^O}(x)$. The value of H^I affects both inputs (σ^O and x). We concern only about σ^O and ignore x. The adversary $\mathcal{A}$ makes q queries to $\mathsf{HMAC}^{(3)}$ and thereby generates $2 + 2 \cdot q$ queries to $\mathsf{f}_{(\overline{K} \oplus \cdot) \boxplus \cdot}(\cdot)$. During the entire interaction, the following keys will be generated:

$$\begin{aligned}
&\overline{K} \oplus ipad, \\
&\overline{K} \oplus opad, \\
K_1^I =& (\overline{K} \oplus ipad) \boxplus \sigma_1^I, \\
K_1^O =& (\overline{K} \oplus opad) \boxplus \sigma_1^O, \\
&\ldots \\
K_q^O =& (\overline{K} \oplus ipad) \boxplus \sigma_q^I, \\
K_q^O =& (\overline{K} \oplus opad) \boxplus \sigma_q^O.
\end{aligned}$$

We have two possible «bad» events:

1) collision in the sequence $\mathbf{K}^O = \{K_1^O, ..., K_q^O\}$;

2) nonempty intersection of $\mathbf{K}^O$ and $\mathbf{K}^I = \{K_1^I, ..., K_q^I, \overline{K} \oplus ipad, \overline{K} \oplus opad\}$.

If no bad events have occurred, then the last compression function never queried with the coinciding arguments and hence output of $\mathsf{HMAC}^{(3)}$ is indistinguishable from a true random function.

The collision in $\mathbf{K}^O$ is guaranteed to generate the collision in the output of $\mathsf{HMAC}^{(3)}$. Therefore, we are obliged to consider them.

Collisions in $\mathbf{K}^I$ are not unsafe. This is taken into account in step 4, if the same keys are used, then the inputs are probably different.

The nonempty intersection of $\mathbf{K}^O$ and $\mathbf{K}^I$ may not lead to the reuse of random functions values, but to simplify the analysis, we consider only the worst case and assume that x value is the same for the same keys.

The probability of the first «bad» event is equal to the probability of the collision $H^I \in V^\tau$

$$\Pr(\exists i \neq j : K_i^O = K_j^O) = \Pr(\exists i \neq j : \sigma_i^O = \sigma_j^O) \leq \frac{q \cdot (q-1)}{2^{\tau+1}}.$$

The set $\mathbf{K}^I$ contains at most $(q+2)$ elements. Due to the bijectivity of modular addition, we have at most $(q+2)$ values σ^O at which $K^O \in \mathbf{K}^I$. Hence, there are at most $(q+2)$ values H^I that lead to «bad» event 2

$$\Pr(\mathbf{K}^O \cap \mathbf{K}^I \neq \emptyset) \leq \frac{q \cdot (q+2)}{2^\tau}.$$

and by the union bound and «fungamental game-playing lemma»

$$\mathrm{Adv}^{PRF}_{\mathsf{HMAC}^{(3)}}(q, l_{\max}) \leq \frac{q \cdot (q-1)}{2^{\tau+1}} + \frac{q \cdot (q+2)}{2^{\tau}} = \frac{3(q^2+q)}{2^{\tau+1}}.$$

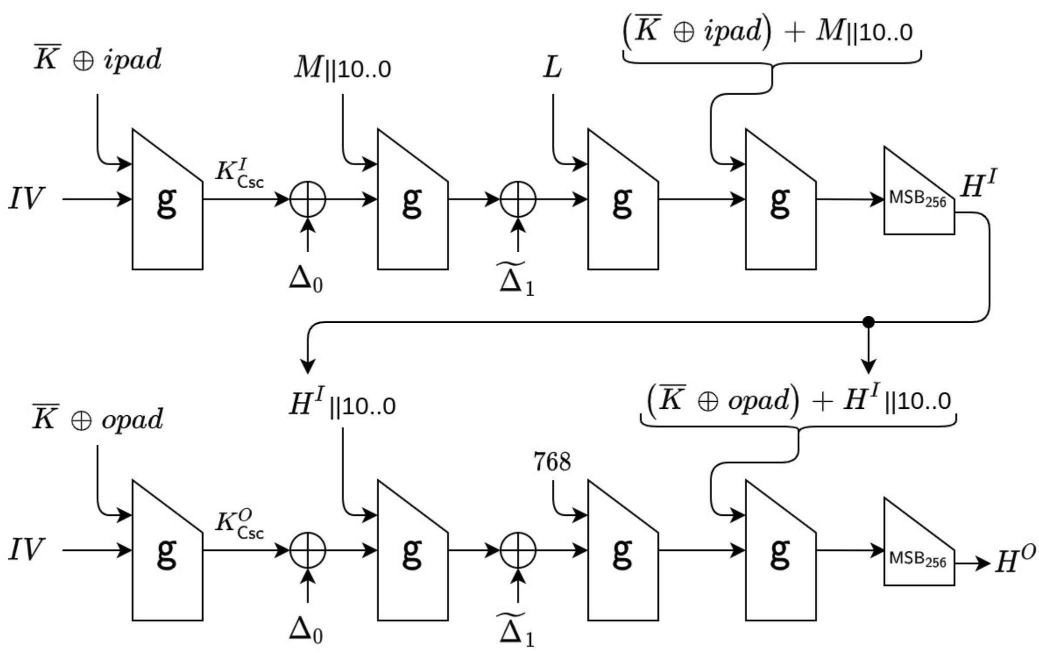

Figure 7: HMAC-Streebog-256 with equivalent representation. The message M consists of $L < 512$ bits.

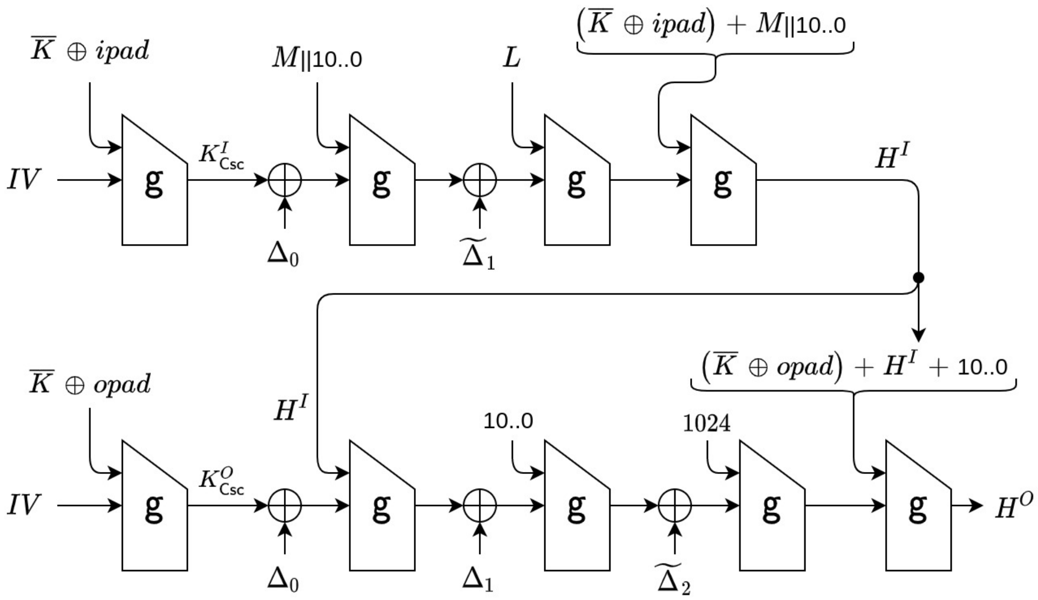

Figure 8: HMAC-Streebog-512 with equivalent representation. The message M consists of $L < 512$ bits.