Provable Security of BLAKE with Non-Ideal Compression Function

Elena Andreeva, Atul Luykx and Bart Mennink

Dept. Electrical Engineering, ESAT/COSIC and IBBT
Katholieke Universiteit Leuven, Belgium
{elena.andreeva, bart.mennink}@esat.kuleuven.be

Abstract. We analyze the security of the SHA-3 finalist BLAKE. The BLAKE hash function follows the HAIFA design methodology, and as such it achieves optimal preimage, second preimage and collision resistance, and is indifferentiable from a random oracle up to approximately $2^{n/2}$ assuming the underlying compression function is ideal.
In our work we show, however, that the compression function employed by BLAKE exhibits a non-random behavior and is in fact differentiable in only $2^{n/4}$ queries. Our attack on the indifferentiability of the BLAKE compression function undermines the provable security strength of BLAKE not only with respect to its overall indifferentiability, but also its collision and (second) preimage security in the ideal model.
Our next contribution is the restoration of the security results for BLAKE in the ideal model by refining the level of modularity and assuming that BLAKE's underlying block cipher is an ideal cipher. We prove that BLAKE is optimally collision, second preimage, and preimage secure (up to a constant). We go on to show that BLAKE is still indifferentiable from a random oracle up to the old bound of $2^{n/2}$ queries, albeit under a weaker assumption: the ideality of its block cipher.

Keywords. SHA-3, BLAKE, collision resistance, (second) preimage resistance, indifferentiability.

1 Introduction

Hash functions are a main building block for numerous cryptographic applications. Due to a series of attacks on the widely deployed SHA-1 hash function by Wang et al. [19, 20], the US National Institute for Standards and Technology (NIST) recommended the replacement of SHA-1 by the SHA-2 hash function family and announced a call for the design of a new SHA-3 hashing algorithm in 2007 [16]. Five candidates, BLAKE [3], Grøstl [12], JH [21], Keccak [5] and Skein [11], made it to the third and final round of the competition. Evaluating the security and performance of the remaining five candidates is crucial in the ongoing process for the selection of the finalist hash function.

The focus of this work is the security of the BLAKE hash function candidate. To assess the security of BLAKE, we follow the security criteria listed by NIST in their call for a new SHA-3 hash function: collision, second preimage, preimage security and resistance to the length extension attacks. The BLAKE hash function is designed by Aumasson et al. [3], and follows the HAIFA design methodology of Biham and Dunkelman [6]. Its underlying compression function f employs internally a block cipher E and exhibits some similarities with the Davies-Meyer compression function [7]. As described in the SHA-3 provable security survey by Andreeva et al. [1], BLAKE inherits preimage, second preimage, collision, and indifferentiability security guarantees of the HAIFA design, assuming ideality of the underlying compression function. More precisely, due to the specific HAIFA counter BLAKE is indifferentiable from a random oracle in the indifferentiability framework of Maurer et al. [14]. The advantage of an adversary against the collision and (second) preimage security of BLAKE is upper bounded by approximately $q^2/2^n$ and $q/2^n$, respectively. While all of these results are true in the ideal compression function model, no concrete (in)differentiability results are known for the BLAKE compression function, which is the main motivation for this work.

Our Contributions. Firstly, in Sect. 3 we present an attack on the BLAKE compression function f, which shows that f is differentiable from a random oracle in $2^{n/4}$ queries. This is less than ideally expected, and as a consequence the existing BLAKE indifferentiability bound, together with the collision and (second)

preimage security bounds from [1], are reduced by a square root. These findings yield a provable security level that is not compliant with the NIST security requirements. The indifferentiability attack is a serious motivation for carrying out further security analysis of the BLAKE hash function in a way that restores its security guarantees. One approach in this direction is to refine the level of modularity in the security analysis and to investigate security properties of the BLAKE hash function under the assumption that the underlying block cipher E, rather than the compression function, is ideal.

This brings us to our second contribution, which is presented in Sect. 4. In the ideal cipher model, we conclude optimal (up to a constant) collision and (everywhere) preimage security of f. This result is important to establish a strong confidence in the security of f in the sense that even if f exhibits some non-ideal behavior, its collision and preimage security are not compromised when E behaves close to ideal. Furthermore, due to the collision and everywhere preimage resistance preservation of the HAIFA design [2], the BLAKE hash function inherits the optimal security of f with respect to both properties.

Next, in Sect. 5 we reconsider the second preimage resistance of BLAKE. As a HAIFA design, BLAKE does not preserve second preimage resistance [2], and proving second preimage security of BLAKE's compression function does not directly translate to the second preimage security of BLAKE. To assess the second preimage security of BLAKE, we therefore analyze directly the BLAKE hash function in the ideal cipher model and prove it optimally (everywhere) second preimage resistance, up to a constant. This result confirms BLAKE's resistance against the second preimage attacks of Dean [10] and Kelsey and Schneier [13], even when the non-ideal compression function of BLAKE is employed.

Finally, in Sect. 6 we restore the indifferentiability result of BLAKE to the old bound of approximately $2^{n/2}$ queries by giving a proof with an ideal underlying block cipher E. We show that despite the differentiability of BLAKE's compression function f, in the ideal cipher model the BLAKE hash function does not suffer structural design flaws. We summarize our results on BLAKE in Table 1.

Our results amount to an important contribution to the security analysis of the SHA-3 finalist BLAKE in a way that addresses all the security criteria of NIST listed in their call for a new SHA-3 hash function. We provide a thorough investigation of these security properties for BLAKE in the ideal block cipher model.

Table 1: A summary of our results on the BLAKE hash function $\mathcal{H}$ and its compression function f. The bounds denote the required number of queries to forge an attack. All results in this table are in the ideal cipher model.

	preimage	second preimage	collision	indifferentiability
f	$\Theta(2^n)$ Sect. 4	–	$\Theta(2^{n/2})$ Sect. 4	$O(2^{n/4})$ Sect. 3
$\mathcal{H}$	$\Theta(2^n)$ Sect. 4	$\Theta(2^n)$ Sect. 5	$\Theta(2^{n/2})$ Sect. 4	$\Omega(2^{n/2})$ Sect. 6

2 Preliminaries

For $n \in \mathbb{N}$, let $\{0,1\}^n$ denote the set of bit strings of length n and let $\{0,1\}^*$ denote the set of bit strings of arbitrary length. For two bit strings x, y, $x\|y$ denotes their concatenation and $x \oplus y$ their bitwise XOR. By $[x]_2 = x\|x$ we denote the concatenation of two copies of x. If x is of even length, then x^l and x^r denote its left and right halves where $|x^l| = |x^r|$. For natural m, n, $\langle m \rangle_n$ is the encoding of m as an n-bit string. We denote by $\mathrm{Bloc}(2n)$ the set of all block ciphers $E : \{0,1\}^{2n} \times \{0,1\}^{2n} \to \{0,1\}^{2n}$, where the first input corresponds to the key input. A random oracle [4] is a function which provides a random output for each new query. A random $2n$-bit block cipher is a block cipher randomly sampled from $\mathrm{Bloc}(2n)$. A random primitive will also be called "ideal".

2.1 BLAKE

In accordance with the SHA-3 hash function specification, BLAKE [3] supports outputs of size $n = 224$, 256, 384, and 512 bits. In this work we focus on the variants $n = 256, 512$, as the 224- and 384-variants are simply chopped versions of these.

BLAKE takes as input a salt s of $n/2$ bits (chosen by the user), and a message M of arbitrary length. The evaluation of $\mathcal{H}(s, M)$ is done as follows. Firstly, the message M is padded into message blocks $m_1, \ldots, m_k$ of $2n$ bits, where the padding function pad is defined as $\mathsf{pad}(M) = M \| 10^{-|M|-n/2-2 \bmod 2n} 1 \| \langle |M| \rangle_{n/2}$. Along with these message blocks, counter blocks $t_1, \ldots, t_k$ of length $n/4$ bits are generated. This counter keeps track of the number of message bits hashed so far and equals 0 if the i-th message block contains no message bits[1]. Starting from an initial state value $h_0 \in \{0,1\}^n$, the message blocks m_i and counter blocks t_i are compressed iteratively into the state using a compression function $f : \{0,1\}^n \times \{0,1\}^{n/2} \times \{0,1\}^{2n} \times \{0,1\}^{n/4} \to \{0,1\}^n$. Here, the second input to f denotes the salt s. The output of the BLAKE hash function is defined as its final state value $\mathcal{H}(s, M) = h_k$.

The compression function f internally uses a block cipher $E : \{0,1\}^{2n} \times \{0,1\}^{2n} \to \{0,1\}^{2n}$. f is depicted in Fig. 1 and is defined as follows. Here, $C \in \{0,1\}^n$ is a constant.

procedure $f(h_{i-1}, s, m_i, t_i)$
  $v_i \leftarrow (h_{i-1} \| s \| [t_i^l]_2 \| [t_i^r]_2) \oplus (0^n \| C)$
  $w_i \leftarrow E(m_i, v_i)$
  $h_i \leftarrow w_i^l \oplus w_i^r \oplus h_{i-1} \oplus [s]_2$
  return h_i
end procedure

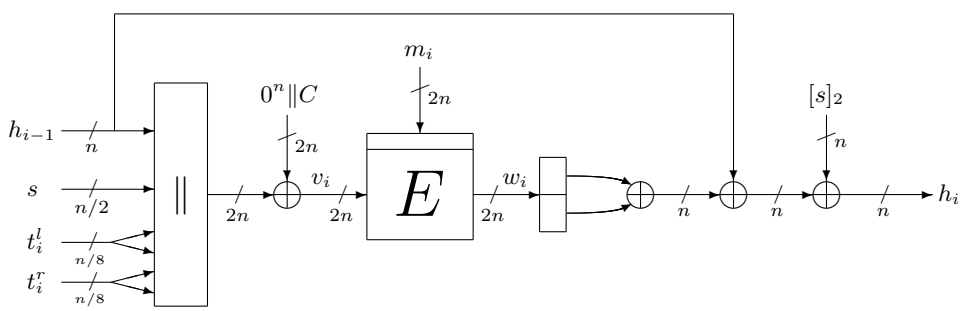

Fig. 1: The BLAKE compression function f of Sect. 2.1.

2.2 Preimage, Second Preimage and Collision Security

An adversary $\mathcal{A}$ is a probabilistic algorithm with oracle access to a randomly sampled block cipher $E \xleftarrow{\$} \mathrm{Bloc}(2n)$. In this work, we consider information-theoretic adversaries only. This type of adversary has unbounded computational power, and its complexity is measured by the number of queries made to his oracle. The adversary can make queries to E and its inverse E^{-1}. These queries are stored in a query history $\mathcal{Q}$ as elements of the form (m_j, v_j, w_j), where j is the query index, m_j is the key input to the block cipher (note that for BLAKE the message input to f is the key input to E), and v_j and w_j denote the plaintext and ciphertext, respectively. Associated to query (m_j, v_j, w_j) we define the value $x_j = w_j^l \oplus w_j^r \oplus h_j \oplus [s_j]_2$ as the output of the compression function f, where we parse $h_j \| s_j \| t_j^{(1)} \| t_j^{(2)} \| t_j^{(3)} \| t_j^{(4)} \leftarrow v_j \oplus (0^n \| C)$. In the remainder, we assume that $\mathcal{Q}$ always contains the queries required for the attack and that the adversary never makes queries to which it knows the answer in advance.

[1] In more detail, $t_i = \langle i2n \rangle_{n/4}$ if $i2n \leq |M|$, $t_i = \langle |M| \rangle_{n/4}$ if $(i-1)2n < |M| \leq i2n$, and $t_i = \langle 0 \rangle_{n/4}$ if $|M| \leq (i-1)2n$.

Let $F : \{0,1\}^p \to \{0,1\}^n$ for $p \geq n$ be a compressing function instantiated with a randomly chosen block cipher $E \xleftarrow{\$} \mathrm{Bloc}(2n)$. In this work, F will either be the BLAKE hash function $\mathcal{H}$ or its compression function f. For the preimage and second preimage security analysis in this work, we consider the notion of everywhere preimage and second preimage resistance [18]. In the ideal model setting (where randomness is gained from the ideal primitive rather than from the use of an explicit random key), these notions are the strongest options because they guarantee preimage (resp. second preimage) security for every range (resp. domain) point.

Definition 1. *Let $p, n \in \mathbb{N}$ with $p \geq n$ and let $F : \{0,1\}^p \to \{0,1\}^n$ be a compressing function employing block cipher $E \in \mathrm{Bloc}(2n)$. The advantage of an everywhere preimage finding adversary $\mathcal{A}$ is defined as*

$$\mathbf{Adv}_F^{\mathrm{epre}}(\mathcal{A}) = \max_{y \in \{0,1\}^n} \mathbf{Pr}\left(\begin{array}{c} E \xleftarrow{\$} \mathrm{Bloc}(2n),\ z \leftarrow \mathcal{A}^{E,E^{-1}}(y)\ : \\ F(z) = y \end{array} \right).$$

We define by $\mathbf{Adv}_F^{\mathrm{epre}}(q)$ the maximum advantage of any adversary making q queries to its oracles.

Definition 2. *Let $p, n \in \mathbb{N}$ with $p \geq n$ and let $F : \{0,1\}^p \to \{0,1\}^n$ be a compressing function employing block cipher $E \in \mathrm{Bloc}(2n)$. Let $\lambda \leq p$. The advantage of an everywhere second preimage finding adversary $\mathcal{A}$ is defined as*

$$\mathbf{Adv}_F^{\mathrm{esec}[\lambda]}(\mathcal{A}) = \max_{z' \in \{0,1\}^\lambda} \mathbf{Pr}\left(\begin{array}{c} E \xleftarrow{\$} \mathrm{Bloc}(2n),\ z \leftarrow \mathcal{A}^{E,E^{-1}}(z')\ : \\ z \neq z' \ \wedge\ F(z) = F(z') \end{array} \right).$$

We define by $\mathbf{Adv}_F^{\mathrm{esec}[\lambda]}(q)$ the maximum advantage of any adversary making q queries to its oracles.

In case F denotes the BLAKE compression function f of Sect. 2.1, its domain points are of the form $z = (h, s, m, t)$. If F is the BLAKE hash function $\mathcal{H}$, its domain points are parsed as $z = (s, M) \in \{0,1\}^{n/2} \times \{0,1\}^*$, where in the second preimage notion λ is required to be of length at least $n/2$ bits.

We define the collision security of a compressing function F as follows.

Definition 3. *Let $p, n \in \mathbb{N}$ with $p \geq n$ and let $F : \{0,1\}^p \to \{0,1\}^n$ be a compressing function employing block cipher $E \in \mathrm{Bloc}(2n)$. Fix a constant $h_0 \in \{0,1\}^n$. The advantage of a collision finding adversary $\mathcal{A}$ is defined as*

$$\mathbf{Adv}_F^{\mathrm{col}}(\mathcal{A}) = \mathbf{Pr}\left(\begin{array}{c} E \xleftarrow{\$} \mathrm{Bloc}(2n),\ z, z' \leftarrow \mathcal{A}^{E,E^{-1}}\ : \\ z \neq z' \ \wedge\ F(z) \in \{F(z'), h_0\} \end{array} \right).$$

We define by $\mathbf{Adv}_F^{\mathrm{col}}(q)$ the maximum advantage of any adversary making q queries to its oracles.

As before, in case F denotes the compression function f the strings z and z' are of the form (h, s, m, t) and (h', s', m', t'), and if F is the BLAKE hash function, z and z' are parsed as (s, M) and (s', M').

2.3 Indifferentiability

The indifferentiability framework introduced by Maurer et al. [14] is an extension of the classical notion of indistinguishability. It proves that if a construction $\mathcal{C}^{\mathcal{G}}$ based on an ideal subcomponent $\mathcal{G}$ is indifferentiable from an ideal primitive $\mathcal{R}$, then $\mathcal{C}^{\mathcal{G}}$ can replace $\mathcal{R}$ in any system. Although recent results by Ristenpart et al. [17] show that indifferentiability does not capture all properties of a random oracle, indifferentiability still remains the best way to rule out structural attacks for a large class of hash function applications.

Definition 4. *A Turing machine $\mathcal{C}$ with oracle access to an ideal primitive $\mathcal{G}$ is called $(t_D, t_S, q, \varepsilon)$ indifferentiable from an ideal primitive $\mathcal{R}$ if there exists a simulator $\mathcal{S}$, such that for any distinguisher $\mathcal{D}$ we have*

$$\mathbf{Adv}_{\mathcal{C}}^{\mathrm{pro}}(\mathcal{D}) = \left| \mathbf{Pr}\left(\mathcal{D}^{\mathcal{C}^{\mathcal{G}},\mathcal{G}} = 1\right) - \mathbf{Pr}\left(\mathcal{D}^{\mathcal{R},\mathcal{S}^{\mathcal{R}}} = 1\right) \right| < \varepsilon.$$

The simulator has oracle access to $\mathcal{R}$ and runs in time at most t_S. The distinguisher runs in time at most t_D and makes at most q queries.

In the remainder, we refer to $\mathcal{C}^{\mathcal{G}}, \mathcal{G}$ as the "real world", and to $\mathcal{R}, \mathcal{S}^{\mathcal{R}}$ as the "simulated world"; the distinguisher $\mathcal{D}$ converses either with the real or the simulated world and its goal is to tell both worlds apart. $\mathcal{D}$ can query both its "left oracle" L (either $\mathcal{C}$ or $\mathcal{R}$) and its "right oracle" R (either $\mathcal{G}$ or $\mathcal{S}$).

For the purpose of the presented indifferentiability results, $\mathcal{G}$ throughout denotes a random block cipher $E \xleftarrow{\$} \mathrm{Bloc}(2n)$. $\mathcal{C}$ is either the BLAKE hash function $\mathcal{H}$ or its compression function f, and $\mathcal{R}$ will be a random oracle RO with the same domain and range as $\mathcal{C}$.

3 Differentiability of f

We consider the indifferentiability of the BLAKE compression function f from a random oracle RO (with the same domain and range as f), when the underlying block cipher E is sampled uniformly at random $E \xleftarrow{\$} \mathrm{Bloc}(2n)$. In more detail, we construct a distinguisher $\mathcal{D}$, such that for any simulator $\mathcal{S}$, $\mathcal{D}$ differentiates (f, E) from $(RO, \mathcal{S})$ in about $2^{n/4}$ queries, hence significantly faster than expected.

The differentiability attack considers fixed-points for f, by which we here mean values (h, m, s, t) such that $f(h, m, s, t) = h \oplus [s]_2$. The presence of fixed-points of these form have already been pointed out in [3, Sect. 5.2.4]. However, not every block cipher evaluation renders a valid compression function evaluation, due to the duplication of t in the block cipher input. The simulator may be able to take advantage of this, which makes the attack proof more complicated.

Theorem 1. *Let $E \xleftarrow{\$} \mathrm{Bloc}(2n)$, and let $RO : \{0,1\}^{n+n/2+2n+n/4} \to \{0,1\}^n$ be a random compression function. For any simulator $\mathcal{S}$ that makes at most $q_{\mathcal{S}} \leq 2^{n-3}$ queries to RO, there exists a distinguisher $\mathcal{D}$ that makes at most $2^{n/4} + 1$ queries to its oracles, such that*

$$\mathbf{Adv}_f^{\mathrm{pro}}(\mathcal{D}) \geq 1 - e^{-1} - \frac{q_{\mathcal{S}}}{2^n} \geq 0.5.$$

Proof. Let $\mathcal{S}$ be any simulator making at most $q_{\mathcal{S}}$ queries to RO. We construct a distinguisher $\mathcal{D}$ that differentiates (f, E) from $(RO, \mathcal{S})$ with a significant probability. $\mathcal{D}$ has query access to (L, R, R^{-1}) (either (f, E, E^{-1}) or $(RO, \mathcal{S}, \mathcal{S}^{-1})$) and operates as follows. Let $\alpha = 2^{n/4}$.

1. For $j = 1, \ldots, \alpha$, $\mathcal{D}$ sets $m_j \leftarrow \langle j \rangle_{2n}$, queries $v_j \leftarrow R^{-1}(m_j, 0)$, and parses $h_j \| s_j \| t_j^{(1)} \| t_j^{(2)} \| t_j^{(3)} \| t_j^{(4)} \leftarrow v_j \oplus (0^n \| C)$;[2]
2. If for all $j \in \{1, \ldots, \alpha\}$ we have $t_j^{(1)} \| t_j^{(3)} = t_j^{(2)} \| t_j^{(4)}$, $\mathcal{D}$ guesses $(L, R) = (RO, \mathcal{S})$ and halts;
3. Otherwise, let $j \in \{1, \ldots, \alpha\}$ be such that $t_j^{(1)} \| t_j^{(3)} = t_j^{(2)} \| t_j^{(4)}$. The distinguisher queries

$$h \leftarrow L(h_j, s_j, m_j, t_j^{(1)} \| t_j^{(3)}),$$

and guesses $(L, R) = (f, E)$ if and only if $h = h_j \oplus [s_j]_2$.

The distinguisher guesses his oracles correctly *except if* one of the following events occur:

$$\begin{array}{llll} \mathsf{E}_1 : & \forall\, j \in \{1, \ldots, \alpha\} : & t_j^{(1)} \| t_j^{(3)} = t_j^{(2)} \| t_j^{(4)} & (L, R) = (f, E); \\ \mathsf{E}_2 : & \exists\, j \in \{1, \ldots, \alpha\} : & t_j^{(1)} \| t_j^{(3)} = t_j^{(2)} \| t_j^{(4)} \text{ and } h = h_j \oplus [s_j]_2 & (L, R) = (RO, \mathcal{S}). \end{array}$$

In particular, $\mathbf{Adv}_f^{\mathrm{pro}}(\mathcal{D}) \geq 1 - \mathbf{Pr}(\mathsf{E}_1) - \mathbf{Pr}(\mathsf{E}_2)$. We start with $\mathbf{Pr}(\mathsf{E}_2)$, and we suppose $(L, R) = (RO, \mathcal{S})$. E_2 in fact covers the event that $\mathcal{S}$ finds a fixed-point for RO, namely inputs h_j, s_j, m_j, t_j such that $RO(h_j, s_j, m_j, t_j) = h_j \oplus [s_j]_2$. As $\mathcal{S}$ makes at most $q_{\mathcal{S}}$ queries, he can find such fixed-point with probability at most $q_{\mathcal{S}}/2^n$. Next, we consider $\mathbf{Pr}(\mathsf{E}_1)$, and we suppose $(L, R) = (f, E)$. As E is a random block cipher and the message blocks m_j are all different, the probabilities $\mathbf{Pr}\ \ t_j^{(1)} \| t_j^{(3)} = t_j^{(2)} \| t_j^{(4)}$ are independent for different indices j, and satisfy

$$\mathbf{Pr}\ \ t_j^{(1)} \| t_j^{(3)} = t_j^{(2)} \| t_j^{(4)} \ = 1 - \mathbf{Pr}\ \ t_j^{(1)} \| t_j^{(3)} = t_j^{(2)} \| t_j^{(4)} \ = 1 - 1/2^{n/4}.$$

Therefore, $\mathbf{Pr}(\mathsf{E}_1) = \left(1 - 1/2^{n/4}\right)^{\alpha}$. For $\alpha = 2^{n/4}$, this bound is at most e^{-1}. We thus obtain $\mathbf{Adv}_f^{\mathrm{pro}}(\mathcal{D}) \geq 1 - e^{-1} - q_{\mathcal{S}}/2^n \geq 0.5$ for $q_{\mathcal{S}} \leq 2^{n-3}$. □

[2] The only requirement on the m_j's is that they are distinct.

4 Collision and Preimage Resistance of f and $\mathcal{H}$

In this section, we analyze the collision and (everywhere) preimage resistance of the BLAKE compression function f. We achieve optimal security (up to a constant). As the HAIFA mode of operation preserves collision and everywhere preimage resistance [2], these results directly carry over to the BLAKE hash function $\mathcal{H}$.

Theorem 2. *Let $n \in \mathbb{N}$. The advantage of any adversary $\mathcal{A}$ in finding a collision for f after $q < 2^{2n-1}$ queries can be upper bounded by*

$$\mathbf{Adv}_f^{\mathrm{col}}(q) \leq \frac{q(q+1)}{2^n}.$$

Proof. Let $j = 1, \ldots, q$. We consider the probability that the j-th query results in a collision. We distinguish between *forward* and *inverse* queries.

Collision by a forward query. The adversary makes an encryption query of the form (m_j, v_j) to receive a ciphertext w_j such that $E(m_j, v_j) = w_j$. Parse $h_j \| s_j \| t_j^{(1)} \| t_j^{(2)} \| t_j^{(3)} \| t_j^{(4)} \leftarrow v_j \oplus (0^n \| C)$. If $t_j^{(1)} \| t_j^{(3)} = t_j^{(2)} \| t_j^{(4)}$ the block cipher query does not correspond to a compression function evaluation and a collision is obtained with probability 0. Hence, we assume $t_j^{(1)} \| t_j^{(3)} = t_j^{(2)} \| t_j^{(4)}$. In this case, the block cipher query corresponds to the compression function evaluation $f(h_j, s_j, m_j, t_j^{(1)} \| t_j^{(3)}) = w_j^l \oplus w_j^r \oplus h_j \oplus [s_j]_2$. The query renders a collision for f only if

$$w_j^l \oplus w_j^r \oplus h_j \oplus [s_j]_2 \in \{x_i \mid i = 1, \ldots, j-1\} \cup \{h_0\}, \tag{1}$$

where the x_i are defined as in Sect. 2.2. Let x denote any of the j elements from the set at the right hand side of (1). The j-th query collides with x with probability $\mathbf{Pr}\ \ w_j^l \oplus w_j^r = x \oplus h_j \oplus [s_j]_2$, where h_j and s_j are fixed by the adversarial input v_j to E. As w_j is generated from a set of size at least $2^{2n} - q$, and at most 2^n values w_j satisfy the equation, this probability is upper bounded by $\frac{2^n}{2^{2n}-q}$. Considering any choice of x, then the j-th query results in a collision with probability at most $\frac{j2^n}{2^{2n}-q}$.

Collision by an inverse query. The adversary makes a decryption query of the form (m_j, w_j) to receive a plaintext v_j. We parse this plaintext as $h_j \| s_j \| t_j^{(1)} \| t_j^{(2)} \| t_j^{(3)} \| t_j^{(4)} \leftarrow v_j \oplus (0^n \| C)$. This query only renders a collision if v_j is consistent with the definition of f, i.e. it satisfies $t_j^{(1)} \| t_j^{(3)} = t_j^{(2)} \| t_j^{(4)}$. Additionally, the query constitutes a collision for f only if

$$w_j^l \oplus w_j^r \oplus h_j \oplus [s_j]_2 \in \{x_i \mid i = 1, \ldots, j-1\} \cup \{h_0\}. \tag{2}$$

Let x denote any of the j elements from the set at the right hand side of (2). The j-th query results in a collision with this x with probability

$$\begin{aligned} &\mathbf{Pr}\ \ h_j \oplus [s_j]_2 = x \oplus w_j^l \oplus w_j^r \ \wedge\ t_j^{(1)} \| t_j^{(3)} = t_j^{(2)} \| t_j^{(4)} \quad , \\ &= \sum_{s \in \{0,1\}^{n/2}} \sum_{t \in \{0,1\}^{n/4}} \mathbf{Pr} \begin{pmatrix} h_j = x \oplus w_j^l \oplus w_j^r \oplus [s]_2 \ \wedge\ s_j = s \\ \wedge\ t_j^{(1)} \| t_j^{(2)} \| t_j^{(3)} \| t_j^{(4)} = [t^l]_2 \| [t^r]_2 \end{pmatrix}, \end{aligned}$$

where the equality holds simply by conditioning on the values attained by s_j and the t_j's. As v_j is generated from a set of size at least $2^{2n} - q$, for any fixed s and t this probability is upper bounded by $\frac{1}{2^{2n}-q}$. Considering any choice of x, s and t, then the j-th query results in a collision with probability at most $\frac{j2^{n/2}2^{n/4}}{2^{2n}-q} = \frac{j2^{3n/4}}{2^{2n}-q}$.

A collision for the compression function f is generated by either a forward or inverse query, and the j-th query thus renders a collision with probability at most $\max\left\{\frac{j2^n}{2^{2n}-q}, \frac{j2^{3n/4}}{2^{2n}-q}\right\} = \frac{j2^n}{2^{2n}-q}$. Summing over all q queries, we obtain

$$\mathbf{Adv}_f^{\mathrm{col}}(q) \leq \sum_{j=1}^{q} \frac{j2^n}{2^{2n} - q} \leq \frac{q(q+1)2^n}{2(2^{2n} - q)}.$$

For $q < 2^{2n-1}$, we have $\frac{1}{2^{2n}-q} \leq \frac{2}{2^{2n}}$, which completes the proof. $\sqcap$

Theorem 3. *Let $n \in \mathbb{N}$. The advantage of any adversary $\mathcal{A}$ in finding a preimage for f after $q < 2^{2n-1}$ queries can be upper bounded by*

$$\mathbf{Adv}_f^{\mathrm{epre}}(q) \leq \frac{2q}{2^n}.$$

Proof. Let $y \in \{0,1\}^n$ be any point to be inverted, as specified in Def. 1. The proof follows the proof of Thm. 2 with the only difference that the j-th query (for $j = 1, \ldots, q$) needs to hit this particular value y, rather than any value x from a set of size j (cf. (1-2)). More detailed, the j-th query needs to satisfy $w_j^l \oplus w_j^r \oplus h_j \oplus [s_j]_2 = y$, and results in a preimage for y with probability at most $\frac{2^n}{2^{2n}-q}$. When summing over all queries, we obtain

$$\mathbf{Adv}_f^{\mathrm{col}}(q) \leq \sum_{j=1}^{q} \frac{2^n}{2^{2n}-q} \leq \frac{q2^n}{2^{2n}-q} \leq \frac{2q}{2^n},$$

where the last inequality holds as $q < 2^{2n-1}$. ⊓

5 Second Preimage Resistance of $\mathcal{H}$

Due to the lack of second preimage security preservation [2] of the BLAKE hash function $\mathcal{H}$, we investigate the second preimage security of $\mathcal{H}$ directly, rather than its compression function (as in the collision and preimage cases). Our proof shows similarities with the second preimage proof for HAIFA by Bouillaguet and Fouque [8]. Our proof, however, is realized in the ideal cipher (rather than ideal compression function) model.

Theorem 4. *Let $n \in \mathbb{N}$, and $\lambda \geq n/2$. The advantage of any adversary $\mathcal{A}$ in finding a second preimage for $\mathcal{H}$ after $q < 2^{2n-1}$ queries can be upper bounded by*

$$\mathbf{Adv}_{\mathcal{H}}^{\mathrm{esec}[\lambda]}(q) \leq \frac{4q}{2^n}.$$

Proof. Let $(s', M') \in \{0,1\}^{n/2} \times \{0,1\}^{\lambda-n/2}$ be the target preimage. Denote $\mathsf{pad}(M') = m_1' \| \cdots \| m_{k'}'$ and denote by $t_1', \ldots, t_{k'}'$ the corresponding counter values. Note that by construction, $t_{i'}' = \langle i'2n \rangle_{n/4}$ for $i' \in \{1, \ldots, k'-2\}$, $t_{k'}' = \langle k'2n \rangle_{n/4}$, and $t_{k'-1}'$ may or may not be of the form $\langle (k'-1)2n \rangle_{n/4}$. The block cipher executions corresponding to this hash function evaluation are given to the adversary for free. That is, $\mathcal{A}$ is forced to make the k' corresponding queries but will not be charged for this. Denote by $h_0', \ldots, h_{k'}'$ the state values corresponding to the evaluation of $\mathcal{H}(s', M')$.

The goal of the adversary is to find a tuple $(s, M) = (s', M')$ such that $\mathcal{H}(s, M) = \mathcal{H}(s', M')$ and such that the query history contains all block cipher evaluations required for the computation of $\mathcal{H}(s, M)$. We pose the following claim.

Claim. Suppose $\mathcal{A}$ finds $(s, M) = (s', M')$ such that $\mathcal{H}(s, M) = \mathcal{H}(s', M')$. Denote by $m_1, \ldots, m_k$, $t_1, \ldots, t_k$, and $h_0, \ldots, h_k$ the message blocks, counter values and intermediate state values corresponding to the computation of $\mathcal{H}(s, M)$. There must be $i \in \{1, \ldots, k\}$ and $i' \in \{1, \ldots, k'\}$ such that $f(h_{i-1}, s, m_i, t_i) = f(h_{i'-1}', s', m_{i'}', t_{i'}')$, where $(h_{i-1}, s, m_i) = (h_{i'-1}', s', m_{i'}')$ and $t_i, t_{i'}'$ satisfy

$$t_i = t_{i'}' \text{ or } \quad t_i = \langle i2n \rangle_{n/4} \text{ and } t_{i'}' = \langle i'2n \rangle_{n/4} \quad . \tag{3}$$

Proof (Proof of claim). As $\mathcal{H}(s, M) = \mathcal{H}(s', M')$, we have $h_k = h_{k'}'$. If $|M| = |M'|$, then $m_k = m_{k'}'$, $t_k = \langle k2n \rangle_{n/4}$ and $t_{k'}' = \langle k'2n \rangle_{n/4}$, and a collision of the prescribed form is found. Thus, suppose $|M| = |M'|$. This implies $k = k'$ and $t_i = t_i'$ for $i = 1, \ldots, k$. If $s = s'$, a collision for h_k is directly found. If $s = s'$, we necessarily have $M = M'$ and by the standard collision resistance preservation proof for the Merkle-Damgård mode of operation (see e.g. [1, 2, 9, 15]), there must by an index i such that $f(h_{i-1}, s, m_i, t_i) = f(h_{i-1}', s', m_i', t_i')$ but $(h_{i-1}, m_i) = (h_{i-1}', m_i')$. This completes the proof. ⊓

It consequently suffices to consider the probability of the adversary finding a collision with any of the k' compression function evaluations of $\mathcal{H}(s', M')$, such that the corresponding counter values satisfy (3). We call a collision of this form a "valid collision". Thus,

$$\mathbf{Adv}_{\mathcal{H}}^{\mathrm{esec}[\lambda]}(q) \leq \sum_{j=1}^{q} \mathbf{Pr}\left(j\text{-th query is valid collision}\right). \tag{4}$$

Let $j = 1, \ldots, q$. We consider the probability that the j-th query results in a collision with any of the target state values. We distinguish between *forward* and *inverse* queries.

Valid collision by a forward query. The adversary makes an encryption query of the form (m_j, v_j) to receive a ciphertext w_j such that $E(m_j, v_j) = w_j$. Parse $h_j \| s_j \| t_j^{(1)} \| t_j^{(2)} \| t_j^{(3)} \| t_j^{(4)} \leftarrow v_j \oplus (0^n \| C)$. If $t_j^{(1)} \| t_j^{(3)} = t_j^{(2)} \| t_j^{(4)}$ the block cipher query does not correspond to a compression function evaluation and a valid collision is obtained with probability 0. Hence, we assume $t_j^{(1)} \| t_j^{(3)} = t_j^{(2)} \| t_j^{(4)}$. In this case, the block cipher query corresponds to the compression function evaluation $f(h, s, m, t_j^{(1)} \| t_j^{(3)}) = w_j^l \oplus w_j^r \oplus h_j \oplus [s_j]_2$.

If $t_j^{(1)} \| t_j^{(3)} = \langle \alpha 2n \rangle_{n/4}$ for some $\alpha \in \{1, \ldots, k'-1\}$, this means the query corresponds to a compression function at the α-th position, and (3) may be satisfied only for $i' = \alpha$. If $t_j^{(1)} \| t_j^{(3)} = \alpha 2n$ for any $\alpha \in \{1, \ldots, k'-2\}$, (3) can be satisfied only for $i' \in \{k'-1, k'\}$. In any other case, there is no i' that makes (3) satisfied. In any case, there are at most 2 values that $w_j^l \oplus w_j^r \oplus h_j \oplus [s_j]_2$ may hit in order to render a valid collision. As in the proof of Thm. 2, the j-th query results in a valid collision with probability at most $\frac{2 \cdot 2^n}{2^{2n-q}}$.
Valid collision by a inverse query. The analysis follows the same lines. The j-th query results in a valid collision with probability at most $\frac{2 \cdot 2^{3n/4}}{2^{2n}-q}$.

A valid collision for the compression function f is generated by either a forward or inverse query, and the j-th query thus renders a valid collision with probability at most $\max\left\{\frac{2 \cdot 2^n}{2^{2n}-q}, \frac{2 \cdot 2^{3n/4}}{2^{2n}-q}\right\} = \frac{2 \cdot 2^n}{2^{2n}-q}$. Summing over all q queries, we obtain from (4):

$$\mathbf{Adv}_{\mathcal{H}}^{\mathrm{esec}[\lambda]}(q) \leq \sum_{j=1}^{q} \frac{2 \cdot 2^n}{2^{2n} - q} \leq \frac{2q2^n}{2^{2n} - q} \leq \frac{4q}{2^n},$$

where the last inequality holds as $q < 2^{2n-1}$. $\sqcap$

6 Indifferentiability of $\mathcal{H}$

We show that the BLAKE hash function is indifferentiable from a random oracle in the ideal cipher model. To this end, we construct a simulator such that any distinguisher requires at least approximately $2^{n/2}$ queries to differentiate $(\mathcal{H}, E, E^{-1})$ from $(RO, \mathcal{S}, \mathcal{S}^{-1})$.

Theorem 5. *Let $E \xleftarrow{\$} \mathrm{Bloc}(2n)$, let $\mathcal{H}$ be the BLAKE hash function, and let RO be a random oracle. Let $\mathcal{D}$ be any distinguisher that makes at most q_L left queries of maximal length $2n \cdot \ell$ bits (not including the salt), q_R queries to R and R^{-1}, and runs in time t. Then*

$$\mathbf{Adv}_{\mathcal{H}}^{\mathrm{pro}}(\mathcal{D}) \leq 3\frac{q(q+1)}{2^n},$$

where $q = \ell q_L + q_R$ and $\mathcal{S}$ is the simulator of Fig. 2, which makes less than q_R queries to RO and runs in time $O(q_R^2)$.

The remainder of this section is devoted to the proof of Thm. 5. In Sect. 6.1, we introduce some additional definitions required for the proof. The simulator used in the proof is introduced and formalized in Sect. 6.2. Then, Thm. 5 is proven in Sect. 6.3.

6.1 Definitions

To facilitate the analysis, we rewrite the BLAKE padding function pad', such that on input of a tuple $(s, M) \in \{0,1\}^{n/2} \times \{0,1\}^*$ it is defined as

$$\mathsf{pad}'(s, M) = (s\|m_1\|t_1) \,\|\, \cdots \,\|\, (s\|m_k\|t_k),$$

with $m_1\|\cdots\|m_k = \mathsf{pad}(M)$ and the t_i calculated appropriately based on the m_i. The strings $s\|m_i\|t_i$ are called augmented message blocks which we will denote by a_i. We analyze the BLAKE hash function with pad' padding and respectively its compression function f that accepts inputs of the form (h, a), with $h \in \{0,1\}^n$, and $a \in \{0,1\}^{n/2+2n+n/4}$ the augmented message.

Let $V := \{0,1\}^{2n}$ be the plain- and ciphertext space of both $\mathcal{S}(m, \cdot)$ and $E(m, \cdot)$. We define $\beta_{h,s} : V \to \{0,1\}^n$ as

$$\beta_{h,s}(w) = w^l \oplus w^r \oplus h \oplus [s]_2.$$

For $h' \in \{0,1\}^n$, we define $\beta_{h,s}^{-1}(h') = \{w \in \{0,1\}^{2n} \mid w^l \oplus w^r \oplus h \oplus [s]_2 = h'\}$. We call these sets the *fibers* of $\beta_{h,s}$. Note that each fiber is of size 2^n.

For the construction of the simulator, we will maintain an initially empty table T, in which all query-response tuples (m, v, w) of $\mathcal{S}$ and $\mathcal{S}^{-1}$ are stored. We write $T_m^+(v) = w$ and $T_m^-(w) = v$. Associated to T we define a graph G, which is initialized with the single node h_0. The compression function evaluations corresponding to the entries of T are maintained in G. The domain and range of $\mathcal{S}$ and $\mathcal{S}^{-1}$ are not intermediate state values and a query-response might not necessarily correspond to a path in G. In fact, a query-response tuple (m, v, w) corresponds to a compression function evaluation and can be converted into a path in G if and only if v can be parsed as $h\|s\|[t^l]_2\|[t^r]_2 \leftarrow v \oplus (0^n\|C)$. In this case, the tuple corresponds to the following path in G:

$$h \xrightarrow{s\|m\|t} \beta_{h,s}(w).$$

Note that if G contains a path $h_0 \xrightarrow{a_1} h_1 \cdots \xrightarrow{a_k} h_k$, this implies T contains all queries required for the evaluation of $f(\ldots f(f(h_0, a_1), a_2) \ldots, a_k) = h_k$, when f is instantiated with $\mathcal{S}$.

6.2 Simulator

Designing the simulator comes down to making sure that $(RO, \mathcal{S}, \mathcal{S}^{-1})$ matches $(\mathcal{H}, E, E^{-1})$ as closely as possible. Notice that for $(\mathcal{H}, E, E^{-1})$ an $\mathcal{H}$ query is a chain of E queries which can be converted to

$$h_0 \xrightarrow{a_1} h_1 \cdots \xrightarrow{a_k} h_k,$$

with $a_1\|\cdots\|a_k = \mathsf{pad}'(s, M)$ for some $s \in \{0,1\}^{n/2}$ and $M \in \{0,1\}^*$; it is this property that the simulator should mimic. What this essentially means is that the simulator needs to carefully handle queries that may extend the set of nodes reachable from h_0. For any other query, it suffices for the simulator to respond randomly.

For simplicity and to improve the readability of the simulator, we opt for a simulator that behaves like a random function. That is, when generating a random answer it will be sampled from V, therewith allowing collisions in T. Clearly, this will result in a higher success probability for the distinguisher, but because the elements of V are of size $2n$ bits (while the hash function has range $\{0,1\}^n$), this security loss will be negligible. This loss will be reflected in the bound obtained in Sect. 6.3.

Recall from Sect. 6.1 that a query-response tuple (m, v, w) adds an edge to the graph if and only if v can be parsed as $h\|s\|[t^l]_2\|[t^r]_2 \leftarrow v \oplus (0^n\|C)$ and for now we simply assume this to be true, adding the edge $h \xrightarrow{s\|m\|t} h' = \beta_{h,s}(w)$ to G. The main purpose of $\mathcal{S}$ is to maintain consistency for the paths leaving from h_0. Thus, we investigate how the simulator handles queries extending any of these paths.

In case of inverse queries, note that h depends on the output of the simulator, v. If the simulator generates v uniformly at random, the edge extends a path from h_0 only if h hits any node already reachable from h_0.

This case occurs with small probability, and we can safely have the simulator respond randomly on an inverse query. The resulting security loss is reflected in the obtained indifferentiability bound derived in Sect. 6.3.

In case of forward queries, h and $a = s\|m\|t$ are determined by the inputs by the distinguisher, and thus he may force the number of nodes reachable from h_0 to increase. Suppose a path $h_0 \xrightarrow{a_1} h_1 \cdots \xrightarrow{a_k} h_k = h$ is in G. We distinguish among the following cases:

- $a_1\|\cdots\|a_k\|a = \mathsf{pad}'(s, M)$ for some $s \in \{0,1\}^{n/2}$ and $M \in \{0,1\}^*$. The simulator should assure consistency with RO, hence its answer w should comply with $\beta_{h,s}(w) = RO(s, M)$;
- $a_1\|\cdots\|a_k\|a = \mathsf{pad}'(s, M)$ for any $s \in \{0,1\}^{n/2}$ and $M \in \{0,1\}^*$. There is no consistency required, and the simulator responds randomly.

Two peculiarities may occur in case of a forward query of this form. At first, it may be the case that a newly added edge extends to two different paths. This would however mean a compression function collision has occurred, an event that happens with small probability and results in a security loss in the final indifferentiability bound obtained in Sect. 6.3. Secondly, the value $h' = \beta_{h,s}(w)$ may hit a node in the graph, in which case the path will be increased with two edges. A similar reasoning as for inverse queries applies here: h' hits another node in the graph with small probability only, and the simulator does not need to handle this situation.

The formal description of the simulator is given in Fig. 2. It uses the following procedure.

```
procedure FINDPATHS(h, a)
    P ← ∅                                          ▷ will contain paths and corresponding messages
    for all paths h_0 --a_1--> h_1 ··· --a_k--> h_k in G do
        if h = h_k and ∃ M such that pad'(s, M) = a_1‖···‖a_k‖a then
            P ← (M, h_0 --a_1--> h_1 ··· --a_k--> h_k)
        end if
    end for
    return P
end procedure
```

Lemma 1. *If the simulator does not return a response retrieved from the table T, the response will be uniformly distributed over V.*

Proof. With a forward query there are two cases, one where the simulator queries RO and one where it does not. If the simulator does not query RO, then by definition it responds uniformly over V. If the simulator does query RO, then it receives an $h' = RO(s, M)$ uniformly distributed over $\{0,1\}^n$. As the fibers of $\beta_{h,s}$ form a partition of V, uniformly selecting an element from an arbitrary fiber of $\beta_{h,s}$ is the same as uniformly selecting an element from V.

Since the inverse queries of the simulator are by definition uniformly distributed over V, we attain our result. ⊓

6.3 Proof of Thm. 5

In this section, we will bound the advantage of any distinguisher in differentiating the simulated world (with the simulator of Fig. 2) from the real world. The proof of indifferentiability consists of a sequence of six games where we specify three algorithms (L_i, R_i, R_i^{-1}) (for $i = 1, \ldots, 6$) with which a distinguisher can interact. These games are given in Fig. 3. The first game corresponds to the simulated world and the sixth game corresponds to the real world $(\mathcal{H}^E, E, E^{-1})$. By G_i we denote the event $\mathcal{D}^{L_i, R_i, R_i^{-1}} = 1$. Clearly,

$$\mathbf{Adv}^{\mathrm{pro}}_{\mathcal{H}}(\mathcal{D}) = |\mathbf{Pr}(G_1) - \mathbf{Pr}(G_6)| \leq \sum_{i=1}^{5} |\mathbf{Pr}(G_i) - \mathbf{Pr}(G_{i+1})| . \tag{5}$$

In the remainder of this section, the distances between the adjacent games will be bounded, and the claim of Thm. 5 will be directly obtained from (5).

Simulator Forward Query

```
procedure S(m, v)
    if T_m^+(v) = ⊥ then
        h‖s‖t^(1)‖t^(2)‖t^(3)‖t^(4) ← v ⊕ (0^n‖C)
        w ←$ V
        if t^(1)‖t^(3) = t^(2)‖t^(4) then
            P ← findPaths(h, s‖m‖t^(1)‖t^(3))
            if P ≠ ∅ then
                (M, path) ←$ P
                w ←$ β_{h,s}^{-1}(RO(s, M))
            end if
            add h --s‖m‖t^(1)‖t^(3)--> β_{h,s}(w) to G
        end if
        T_m^+(v) ← w
    end if
    return T_m^+(v)
end procedure
```

Simulator Inverse Query

```
procedure S^{-1}(m, w)
    if T_m^-(w) = ⊥ then
        v ← T_m^-(w) ←$ V
        h‖s‖t^(1)‖t^(2)‖t^(3)‖t^(4) ← v ⊕ (0^n‖C)
        if t^(1)‖t^(3) = t^(2)‖t^(4) then
            add h --s‖m‖t^(1)‖t^(3)--> β_{h,s}(w) to G
        end if
    end if
    return T_m^-(w)
end procedure
```

Random Oracle

```
procedure RO(s, M)
    if F[s, M] = ⊥ then          ▷ F is an array
        F[s, M] ←$ {0,1}^n
    end if
    return F[s, M]
end procedure
```

Fig. 2: The definition of the simulator $\mathcal{S}$ used in the proof of Thm. 5, and the random oracle RO.

Games 1 and 2

The first game is $(RO, \mathcal{S}^{RO}, (\mathcal{S}^{RO})^{-1})$. The biggest change in the second game is that $\mathcal{H}^{\mathcal{S}}$ is called in L_2 in line 2. Note that the result of $\mathcal{H}^{\mathcal{S}}$ is not used and L_2 returns a value generated by RO, i.e. the responses of L_1 and L_2 are identical. Yet, calling $\mathcal{H}^{\mathcal{S}}$ still has side effects on game 2 as the simulator's table and graph are updated based on the $\mathcal{S}$ queries made by $\mathcal{H}^{\mathcal{S}}$. As a result the simulator in game 2 gains more knowledge than the simulator in game 1. In particular, we will *mark* each query-response from all calls of $\mathcal{S}$ in $\mathcal{H}^{\mathcal{S}}$ whenever a query has not been made before by $\mathcal{D}$; these marked query-responses in T represent the extra knowledge gained by the simulator in game 2. Queries made by $\mathcal{D}$ are made unmarkable in line 12 in R_2 and line 22 in R_2^{-1}, these queries provide the simulator with no extra information compared to the simulator of game 1.

Note that if we ignore the lines of code in R_2 and R_2^{-1} dealing with marked query-responses we are left with lines 11, 13, 21, and 23, where we see that R_2 and R_2^{-1} simply query $\mathcal{S}$ and $\mathcal{S}^{-1}$ and return the result. The rest of the code is there in order to undo the side effects of $\mathcal{H}^{\mathcal{S}}$. A first step in removing the side effects of $\mathcal{H}^{\mathcal{S}}$ is, when a marked query is made by $\mathcal{D}$, to remove the knowledge $\mathcal{S}$ has of that query (implemented in the if-statements) and then to re-query $\mathcal{S}$ again. This does not deal with all possible cases because we know that sometimes $\mathcal{S}$ queries RO in order to ensure consistency. In particular, in game 1 the distinguisher could know that a particular intermediate value should map to the result of some L_1 response while the simulator does not know this, resulting in a collision. Yet, this collision could be avoided if the simulator knows all of the intermediate values used through a call to $\mathcal{H}^{\mathcal{S}}$. To this end, R_2 employs the following procedure so that $\mathcal{S}$ "forgets" the intermediate values:

```
procedure deleteMarkedPaths(m, v)
    h‖s‖t^(1)‖t^(2)‖t^(3)‖t^(4) ← v ⊕ (0^n‖C)
    if t^(1)‖t^(3) ≠ t^(2)‖t^(4) then
        return
    end if
    P ← findPaths(h, s‖m‖t^(1)‖t^(3))
    for all (M, h_0 --a_1--> ··· --a_k--> h_k) ∈ P do
        for i ← 0, ..., k − 1 do
            if (m, v, w) associated with h_i --a_{i+1}--> h_{i+1} is marked then
```

Game 1

```
procedure L_1(s, M)
    return RO(s, M)
end procedure

procedure R_1(m, v)
    return S(m, v)
end procedure

procedure R_1^{-1}(m, w)
    return S^{-1}(m, w)
end procedure
```

Game 2

```
procedure L_2(s, M)
    mark all (m, v, w) used in H^S(s, M)
    return RO(s, M)
end procedure

procedure R_2(m, v)
    if (m, v, T_m^+(v)) is marked then
        delete (m, v, T_m^+(v)) from T
        delete corresponding path from G
    end if
    deleteMarkedPaths(m, v)
    w ← S(m, v)
    make (m, v, w) unmarkable
    return w
end procedure

procedure R_2^{-1}(m, w)
    if (m, T_m^-(w), w) is marked then
        bad ← true
        delete (m, T_m^-(w), w) from T
        delete corresponding path from G
    end if
    v ← S^{-1}(m, w)
    make (m, v, w) unmarkable
    return v
end procedure
```

Game 3

```
procedure L_3(s, M)
    return L_2(s, M)
end procedure

procedure R_3(m, v)
    return R_2(m, v)
end procedure

procedure R_3^{-1}(m, w)
    if (m, T_m^-(w), w) is marked then
        bad ← true
    end if
    return S^{-1}(m, w)
end procedure
```

Game 4

```
procedure L_4(s, M)
    H^{R_4}(s, M)
    return RO(s, M)
end procedure

procedure R_4(m, v)
    w ← S(m, v)
    if T_m^+(v) = ⊥ then
        bad ← isCollision(m, v, T_m^+(v))
    end if
    return w
end procedure

procedure R_4^{-1}(m, w)
    return S^{-1}(m, w)
end procedure
```

Game 5

```
procedure L_5(s, M)
    return H^{R_5}(s, M)
end procedure

procedure R_5(m, v)
    return R_4(m, v)
end procedure

procedure R_5^{-1}(m, w)
    return R_4^{-1}(m, v)
end procedure
```

Game 6

```
procedure L_6(s, M)
    return H^{R_6}(s, M)
end procedure

procedure R_6(m, v)
    return E(m, v)
end procedure

procedure R_6^{-1}(m, w)
    return E^{-1}(m, w)
end procedure
```

Fig. 3: Games $1, \ldots, 6$ used in the proof of Thm. 5.

```
            delete (m, v, w) from T
            delete h_i --a_{i+1}--> h_{i+1} from G
        end if
    end for
  end for
end procedure
```

When invoked through a forward query, DELETEMARKEDPATHS checks for all paths to which this particular query extends using the FINDPATHS procedure and deletes any marked query-responses used along these paths, thereby eliminating all marked intermediate values used by a $\mathcal{H}^{\mathcal{S}}$ query.

Now we take a look at R_2^{-1} and see exactly how the $\mathcal{H}^{\mathcal{S}}$ query is dealt with. If the query-response $(m, T_m^-(w), w)$ is not marked, then either (m, w) has never been queried before or the distinguisher has queried (m, w) before; in either case we get the exact same behavior as R_1^{-1}. If $(m, T_m^-(w), w)$ is marked, then this means that the distinguisher has not queried (m, w) and that $\mathcal{H}^{\mathcal{S}}$ has queried (m, w). Removing $(m, T_m^-(w), w)$ from T and G and then querying $\mathcal{S}^{-1}(m, w)$ will return some uniformly chosen response from V. This is the same as never having queried $\mathcal{S}^{-1}(m, w)$ and then querying it, meaning we get the same behavior out of $\mathcal{S}^{-1}$ in R_2^{-1} as in R_1^{-1}. Note that the newly generated $\mathcal{S}^{-1}(m, w)$ very likely differs from the value previously generated (when it was queried by $\mathcal{H}^{\mathcal{S}}$). However, as L_2 never discloses the data from $\mathcal{H}^{\mathcal{S}}$, this is not a problem.

Finally we just need to compare R_1 with R_2. Say that $(m, v, T_m^+(v))$ is unmarked, i.e. $\mathcal{H}^{\mathcal{S}}$ has never queried (m, v). When calling DELETEMARKEDPATHS, there are a few possibilities:

- FINDPATHS returns the empty set. The subsequent call to $\mathcal{S}$ will return some arbitrary element of V, as would exactly happen in R_1;
- FINDPATHS finds some valid path, but there are no marked query-responses along this path. This means that the distinguisher has queried the full path itself and $\mathcal{S}$ will respond similarly in both R_1 and R_2;
- FINDPATHS finds a valid path and there are marked query-responses along this path, but these are removed. Thus, the simulator call from R_2 has the same amount of information as the simulator call from R_1.

If $(m, v, T_m^+(v))$ is marked, then knowledge of that particular query-response is removed. In effect we are then dealing with an unmarked query-response $(m, v, \perp)$ and are reduced to the case above.

We have shown that each R_2 and R_2^{-1} query will execute the same code within $\mathcal{S}$ as each R_1 and R_1^{-1} query, respectively, and since they all return the response of the $\mathcal{S}$ query, we have

$$\mathbf{Pr}\,(G_1) = \mathbf{Pr}\,(G_2)\,.$$

Games 2 and 3

Note that L_2 and L_3, and R_2 and R_3 are exactly the same, so we need to compare the responses of R_2^{-1} and R_3^{-1}. It is clear that R_2^{-1} and R_3^{-1} are identical until **bad**. The **bad** event corresponds to $\mathcal{H}^{\mathcal{S}}$ first querying $\mathcal{S}$ resulting in the query-response (m, v, w) with path $h_1 \xrightarrow{a} h_2$, the distinguisher guessing this particular w correctly from the set $\beta_{h_1,s}^{-1}(h_2)$, and finally $\mathcal{D}$ calling $R_i^{-1}(m, w)$ without explicitly calling $R_i(m, v)$ (otherwise $R_i(m, v)$ would unmark (m, v, w)). Since every fiber of $\beta_{h_1,s}^{-1}$ has size 2^n, an upper bound for the probability of finding such a w is $q_R/2^n$, as q_R bounds the number of right oracle inverse queries by $\mathcal{D}$. Therefore, as **bad** can be triggered in both games,

$$|\mathbf{Pr}\,(G_2) - \mathbf{Pr}\,(G_3)| \leq 2\frac{q_R}{2^n}.$$

Games 3 and 4

The following procedure is used in game 4 to detect collisions:

```
procedure isCollision(m, v, w)
    h||s||t^(1)||t^(2)||t^(3)||t^(4) ← v
    if t^(1)||t^(3) = t^(2)||t^(4) then
        return false
    end if
    h' ← β_{h,s}(w)
```

return h' is a node of G ▷ returns *true/false*
end procedure

Since R_4 is identical to $\mathcal{S}$, L_3 and L_4 are identical. The only difference between games 3 and 4 can be found in R_3 and R_4, yet this is the same difference as between R_1 and R_2 and we may conclude that

$$\mathbf{Pr}\left(G_3\right) = \mathbf{Pr}\left(G_4\right).$$

Games 4 and 5

The difference between games 4 and 5 lies in the response given by the left oracles: game 4 uses RO while game 5 uses $\mathcal{H}^{R_5}$. We will show that as long as **bad** is not triggered, the responses of both left oracles are the same.

Lemma 2. *As long as **bad** is not set to true, $L_4(s, M) = L_5(s, M)$.*

Proof. We can write $\mathcal{H}^{R_i}(s, M)$, with i equal to 4 or 5, as

$$h_0 \xrightarrow{a_1} h_1 \cdots \xrightarrow{a_k} h_k,$$

with $a_1 \| \cdots \| a_k = \mathsf{pad}'(s, M)$. If none of the nodes h_j for $j > 0$ are in G, then $\mathcal{H}^{R_i}$ will query R_i in sequence starting from h_0 and ending up at $RO(s, M)$ since the simulator will learn the entire message M in sequence by the time $h_{k-1} \xrightarrow{a_k} h_k$ is queried and can respond with $RO(s, M)$.

On the other hand, if there is some node h_j in G, then it must be the case that the particular path $h_{j-1} \xrightarrow{a_j} h_j$ is in G, otherwise $\mathcal{H}^{R_i}(s, M)$ will trigger **bad**. Furthermore $h_{j-2} \xrightarrow{a_{j-1}} h_{j-1}$ must have been queried before the $h_{j-1} \xrightarrow{a_j} h_j$ query:

- if $h_{j-2} \xrightarrow{a_{j-1}} h_{j-1}$ is not in G then $\mathcal{H}^{R_i}(s, M)$ will place it in G resulting in a collision because h_{j-1} is already in G, and
- if $h_{j-2} \xrightarrow{a_{j-1}} h_{j-1}$ is in G then it must have occurred before the a_j query since the result of the a_{j-1} query would otherwise have ended up as a node in G.

This means that R_i receives each of the $h_{j-1} \xrightarrow{a_j} h_j$ queries in order from $j = 1$ to k and can respond consistently with $RO(s, M)$. ⊓

By the collision resistance of the BLAKE compression function (Sect. 4), the probability of a collision occurring is upper bounded by $q(q+1)/2^n$. Hence, as **bad** can be triggered in both games,

$$\mathbf{Pr}\left(G_4\right) - \mathbf{Pr}\left(G_5\right) \ \leq 2\frac{q(q+1)}{2^n}.$$

Games 5 and 6

The right oracles of game 6 form a permutation for each message, whereas the right oracles of game 5 do not. By Lem. 1, the right oracles of game 5 are uniformally distributed over V (R_5 and R_5^{-1} are essentially just the simulator), which means that the difference between game 5 and game 6 is the difference between a permutation and a random function, which we know is bounded as follows:

$$\mathbf{Pr}\left(G_5\right) - \mathbf{Pr}\left(G_6\right) \ \leq \frac{q_R(q_R - 1)}{2^{2n}}.$$

Acknowledgments. This work has been funded in part by the IAP Program P6/26 BCRYPT of the Belgian State (Belgian Science Policy), in part by the European Commission through the ICT program under contract ICT-2007-216676 ECRYPT II, and in part by the Research Council K.U.Leuven: GOA TENSE. The third author is supported by a Ph.D. Fellowship from the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen).

References

[1] Andreeva, E., Mennink, B., Preneel, B.: Security reductions of the second round SHA-3 candidates. In: Information Security Conference - ISC 2010. Lecture Notes in Computer Science, vol. 6531, pp. 39–53. Springer-Verlag, Berlin (2010)

[2] Andreeva, E., Neven, G., Preneel, B., Shrimpton, T.: Seven-property-preserving iterated hashing: ROX. In: Advances in Cryptology - ASIACRYPT 2007. Lecture Notes in Computer Science, vol. 4833, pp. 130–146. Springer-Verlag, Berlin (2007)

[3] Aumasson, J., Henzen, L., Meier, W., Phan, R.: SHA-3 proposal BLAKE (2010), submission to NIST's SHA-3 competition

[4] Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: ACM Conference on Computer and Communications Security. pp. 62–73. ACM, New York (1993)

[5] Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: The KECCAK sponge function family (2011), submission to NIST's SHA-3 competition

[6] Biham, E., Dunkelman, O.: A framework for iterative hash functions – HAIFA. Cryptology ePrint Archive, Report 2007/278 (2007), `http://eprint.iacr.org/2007/278`

[7] Black, J., Rogaway, P., Shrimpton, T.: Black-box analysis of the block-cipher-based hash-function constructions from PGV. In: Advances in Cryptology - CRYPTO 2002. Lecture Notes in Computer Science, vol. 2442, pp. 320–335. Springer-Verlag, Berlin (2002)

[8] Bouillaguet, C., Fouque, P.: Practical hash functions constructions resistant to generic second preimage attacks beyond the birthday bound (2010), submitted to Information Processing Letters

[9] Damgård, I.: A design principle for hash functions. In: Advances in Cryptology - CRYPTO '89. Lecture Notes in Computer Science, vol. 435, pp. 416–427. Springer-Verlag, Berlin (1990)

[10] Dean, R.: Formal Aspects of Mobile Code Security. Ph.D. thesis, Princeton University, Princeton (1999)

[11] Ferguson, N., Lucks, S., Schneier, B., Whiting, D., Bellare, M., Kohno, T., Callas, J., Walker, J.: The Skein Hash Function Family (2010), submission to NIST's SHA-3 competition

[12] Gauravaram, P., Knudsen, L., Matusiewicz, K., Mendel, F., Rechberger, C., Schläffer, M., Thomsen, S.: Grøstl – a SHA-3 candidate (2011), submission to NIST's SHA-3 competition

[13] Kelsey, J., Schneier, B.: Second preimages on n-bit hash functions for much less than 2^n work. In: Advances in Cryptology - EUROCRYPT 2005. Lecture Notes in Computer Science, vol. 3494, pp. 474–490. Springer-Verlag, Berlin (2005)

[14] Maurer, U., Renner, R., Holenstein, C.: Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In: Theory of Cryptography Conference 2004. Lecture Notes in Computer Science, vol. 2951, pp. 21–39. Springer-Verlag, Berlin (2004)

[15] Merkle, R.: One way hash functions and DES. In: Advances in Cryptology - CRYPTO '89. Lecture Notes in Computer Science, vol. 435, pp. 428–446. Springer-Verlag, Berlin (1990)

[16] National Institute for Standards and Technology: Announcing request for candidate algorithm nominations for a new cryptographic hash algorithm (SHA3) family (November 2007), `http://csrc.nist.gov/groups/ST/hash/documents/FR_Notice_Nov07.pdf`

[17] Ristenpart, T., Shacham, H., Shrimpton, T.: Careful with composition: Limitations of the indifferentiability framework. In: Advances in Cryptology - EUROCRYPT 2011. Lecture Notes in Computer Science, vol. 6632, pp. 487–506. Springer-Verlag, Berlin (2011)

[18] Rogaway, P., Shrimpton, T.: Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In: Fast Software Encryption 2004. Lecture Notes in Computer Science, vol. 3017, pp. 371–388. Springer-Verlag, Berlin (2004)

[19] Wang, X., Yin, Y., Yu, H.: Finding collisions in the full SHA-1. In: Advances in Cryptology - CRYPTO 2005. Lecture Notes in Computer Science, vol. 3621, pp. 17–36. Springer-Verlag, Berlin (2005)

[20] Wang, X., Yu, H.: How to break MD5 and other hash functions. In: Advances in Cryptology - EUROCRYPT 2005. Lecture Notes in Computer Science, vol. 3494, pp. 19–35. Springer-Verlag, Berlin (2005)

[21] Wu, H.: The Hash Function JH (2011), submission to NIST's SHA-3 competition