A Three-Property-Secure Hash Function

Elena Andreeva and Bart Preneel

SCD-COSIC, Dept. of Electrical Engineering, Katholieke Universiteit Leuven,
{Elena.Andreeva,Bart.Preneel}@esat.kuleuven.be

Abstract. This paper proposes a new hash construction based on the widely used Merkle-Damgård (MD) iteration [13,9]. It achieves the three basic properties required from a cryptographic hash function: collision (Coll), second preimage (Sec) and preimage (Pre) security. We show property preservation for the first two properties in the standard security model and the third Pre security property is proved in the random oracle model. Similar to earlier known hash constructions that achieve a form of Sec (eSec [16]) property preservation [4,17], we make use of fixed key material in the iteration. But while these hashes employ keys of size at least logarithmic in the message length (in blocks), we only need a small constant key size. Another advantage of our construction is that the underlying compression function is instantiated as a keyless primitive.

The Sec security of our hash scheme, however, relies heavily on the standard definitional assumption that the target messages are sufficiently random. An example of a practical application that requires Sec security and satisfies this definitional premise on the message inputs is the popular Cramer-Shoup encryption scheme [8]. Still, in practice we have other hashing applications where the target messages are not sampled from spaces with uniform distribution. And while our scheme is Sec preserving for uniform message distributions, we show that this is not always the case for other distributions.

1 Introduction

Hash functions in cryptography are used to compress inputs of arbitrary length to outputs of a fixed size. A typical way to build a hash function is to iteratively apply a fixed-input length compression function. Practical hash functions today are predominantly based on this principle and the most widespread application of an iterative construction is the Merkle-Damgård (MD) hash [13,9]. The main security feature of the MD hash is its collision (Coll) security preservation, which means that if the compression function is collision secure, then the iterated hash function is collision secure as well. But collision security is not the only security property required from hash functions. A good hash function should also be second preimage (Sec) and preimage (Pre) secure.

The recent attacks of Wang et al. [18,19,20], however, have revealed weaknesses in the expected ideal collision strength of the SHA-0 and SHA-1 hash functions. These and earlier MD5 collision attacks suggest that designing a collision secure hash function may turn out to be difficult. With the loss of the

R. Avanzi, L. Keliher, and F. Sica (Eds.): SAC 2008, LNCS 5381, pp. 228–244, 2009.

Coll security guarantee current hash functions fail also to provide apt security for the weaker security properties of Sec and Pre (see the attacks of [11] and the counterexamples of [2]).

The National Institute of Standards and Technology (NIST) of US has in turn addressed the problem by announcing a call for new hash functions [14]. The minimal security requirements stated in the call for proposals are Coll, Sec and Pre security with computational complexity of order $2^{n/2}$, $2^{n-\ell}$ and 2^n, respectively. Here the hash values are of n bits and the Sec security is expressed in terms of the message length in blocks (2^ℓ). We believe that proposals for new hash functions should provide guarantees for security preservation for not only Coll, but also Sec and Pre security. Preservation proofs are important because they allow one to rely on the hash function strength with respect to a concrete security property, independently of the weaknesses of the other properties.

The problem of designing a property-preserving iterations has been earlier investigated by [2,5,6,4,7,10,17]. Although these papers sometimes aim for properties different from the ones mentioned above, showing a property preserving hash function is one of their main goals.

In this paper we propose a new iterative hash function, that is based on the MD hash principle, and provably preserves the notions of Coll and Sec security in the standard security model and achieves Pre security when the compression function is instantiated as a random oracle. Our reduction for Coll is tight and we lose a factor of the message length (in blocks) in the Sec preservation. In the estimated Sec gap we are also able to mount the Sec attacks of [11,1]. Still, as we show, these attacks are only possible for target messages of a very specific structure. Finding a preimage message takes approximately 2^n evaluations of the compression function when it is modeled as a random oracle.

Our hash design benefits from a keyless compression function and makes use of keys in the iteration. We call this the *keyless compression function – keyed iteration* setting. Compared to the dedicated key setting of [6] (keyed compression function – keyed iteration), we achieve the three basic properties with a more practically understood and employed primitive, namely a keyless compression function. In the iterative portion of our design, we have reasons to believe that achieving security guarantees for Sec security is hard without the use of some form of randomization (provided by the keys in our case). A publicly known key selects a single function from a family of hash functions and once chosen at random it remains fixed for the hash algorithm. Note however, that any security claims for keyed hash functions hold only as long as the keys are generated honestly. If the keys are maliciously chosen, then they become exploitable constants and could potentially give rise to future attacks. A possible way to employ fixed keys in practice is to make the key selection process open and fixed in standard.

Achieving Coll and Sec preservation in the standard security and Pre security in the random oracle model partially attempts to answer the question from [2] if a multi-property preserving hash transform is realizable in the standard model. To achieve a seven-property-preserving hash function the authors of [2] benefit from the use of a random oracle for the mask (key) generation. Also, compared

to [4,17], which show the eSec property preservation, we use the key material sparsely. While the latter hash constructions use keys of length at least logarithmic in the message size (in blocks), we only need keys of constant length, $b + 2n$ bits, where b and n are the block and hash sizes, respectively.

Together with the basic hash function, we present some generalized versions of it. These vary according to the order in which the input values are processed by the compression function F. Still, the optimal input ordering for F in the iteration heavily depends on the specifications of the concrete compression function.

In the line of this work, another interesting problem has come to our notice. While our hash scheme offers a theoretically sound Sec preservation proof, in practical scenarios this result may lack the claimed strength. Why does this discrepancy occur? The standard Sec security definition assumes a uniform target message space distribution. We use this fact in our hash design to extract randomness from the message and mix it with the chaining portion of the iteration. A prominent example application that requires Sec security and where the hash inputs are chosen uniformly at random is the Cramer-Shoup cryptosystem [8][1]. However, in some applications, the target message space may not have the uniform distribution. By building a Sec secure compression function, we are able to demonstrate a Sec attack on our hash only for such biased distributions.

On the other hand, working with non-uniform target message distributions allows for better message visibility. Some messages are hashed with higher probability and thus are more predictable. This interpretation deviates from the Sec definition and is a shift towards the notion of target collision security, or eSec from [16], where the messages are fully predictable (chosen) by the adversary. This observation can be interpreted in two ways. One way to think about the problem is to work with variants of the Sec definitions that take into account the target message distribution. Another solution may be to provide appropriate message input randomization to guarantee the randomness of the message inputs. In the final part of our paper we provide a short discussion on the issue.

2 Security Definitions

NOTATION. Let ε be the empty string. $x \| y$ denotes the concatenation of strings x and y. If x is a string, then $x|_z^{\mathtt{msb}}$ and $x|_y^{\mathtt{lsb}}$ specify the most z and least y, respectively, significant bits of x. $|x|$ is the length in bits of the string x and $x|_i^j$ is the substring of x containing the i-th through j-th bit of x, inclusive.

If S is a set, then $x \xleftarrow{\$} S$ denotes the uniformly random selection of an element from S. We let $y \leftarrow \mathsf{A}(x)$ and $y \xleftarrow{\$} \mathsf{A}(x)$ be the assignment to y of the output of a deterministic and randomized algorithm A, respectively, when run on input x.

[1] "For this purpose, we will use a family of hash functions, such that given a randomly chosen tuple of group elements and randomly chosen hash function key, it is computationally infeasible to find a different tuple of group elements that hashes to the same value using the given hash key."

Definition of target collision resistance from [8] matching the standard Sec security one.

An *adversary* is an algorithm with polynomial running time, possibly with access to some oracles. To avoid trivial lookup attacks, it will be our convention to include in the time complexity of an adversary A its running time and its code size (relative to some fixed model of computation).

In our *keyless compression function-keyed iteration* setting we model the fixed-input-size function to be a keyless compression function. The iterative arbitrary-input-size hash function on the other hand is a family of functions indexed by a fixed random key.

SECURITY OF COMPRESSION FUNCTIONS. Let $\mathrm{F} : \{0,1\}^{b+n} \to \{0,1\}^n$ be a compression function that takes inputs of fixed size $(n+b)$ bits and maps them to outputs of size n. First we define the following advantage measures for Coll and Sec security for a fixed adversary A and message length $\lambda \in \mathbb{N}$:

$$\mathbf{Adv}^{\mathrm{Coll}}_{\mathrm{F}}(\mathsf{A}) = \Pr\left[\ M', M \xleftarrow{\$} \mathsf{A}(\varepsilon)\ :\ M \neq M' \text{ and } \mathrm{F}(M) = \mathrm{F}(M')\ \right]$$

$$\mathbf{Adv}^{\mathrm{Sec}[\lambda]}_{\mathrm{F}}(\mathsf{A}) = \Pr\left[\ M \xleftarrow{\$} \{0,1\}^\lambda\ ;\ M' \xleftarrow{\$} \mathsf{A}(M)\ :\ M \neq M' \text{ and } \mathrm{F}(M) = \mathrm{F}(M')\ \right]$$

$$\mathbf{Adv}^{\mathrm{Pre}[\lambda]}_{\mathrm{F}}(\mathsf{A}) = \Pr\left[\ M \xleftarrow{\$} \{0,1\}^\lambda\ ;\ Y \leftarrow \mathrm{F}(M)\ ;\ M' \xleftarrow{\$} \mathsf{A}(Y)\ :\ \mathrm{F}(M') = Y\ \right]$$

We say that F is (t,ϵ) *atk* secure for $\mathrm{atk} \in \{\mathrm{Sec}, \mathrm{Pre}\}$ if $\mathbf{Adv}^{\mathrm{atk}[\lambda]}_{\mathrm{F}}(\mathsf{A}) < \epsilon$ for all adversaries A running in time at most t and $\lambda = b+n$. Note that it is impossible to define security for the case of Coll in an analogous way. Indeed, if collisions on F exist, then an adversary A that simply prints out a collision that is hardcoded into it always has advantage 1. Rather than defining Coll security through the non-existence of an algorithm A, we follow Rogaway's human-ignorance approach [15] and use the above advantage function as a metric to relate the advantage of an adversary A against the hash function to that of an adversary B against the compression function.

SECURITY OF HASH FUNCTIONS. A hash function family is a function $\mathrm{H} : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ where the key space $\mathcal{K}$ and the target space $\mathcal{Y}$ are finite sets of bit strings. The message space $\mathcal{M}$ could be infinitely large; we assume that there exists at least one $\lambda \in \mathbb{N}$ such that $\{0,1\}^\lambda \subseteq \mathcal{M}$. The key K is an index that selects a instance from the function family. Following [16], we use the following advantage measures:

$$\mathbf{Adv}^{\mathrm{Coll}}_{\mathrm{H}}(\mathsf{A}) = \Pr\left[\ K \xleftarrow{\$} \mathcal{K}\ ;\ (M, M') \xleftarrow{\$} \mathsf{A}(K)\ :\ \begin{array}{c} M \neq M' \text{ and} \\ \mathrm{H}(K, M) = \mathrm{H}(K, M') \end{array}\ \right]$$

$$\mathbf{Adv}^{\mathrm{Sec}[\lambda]}_{\mathrm{H}}(\mathsf{A}) = \Pr\left[\ \begin{array}{c} K \xleftarrow{\$} \mathcal{K}\ ;\ M \xleftarrow{\$} \{0,1\}^\lambda \\ M' \xleftarrow{\$} \mathsf{A}(K, M) \end{array}\ :\ \begin{array}{c} M \neq M' \text{ and} \\ \mathrm{H}(K, M) = \mathrm{H}(K, M') \end{array}\ \right]$$

$$\mathbf{Adv}^{\mathrm{Pre}[\lambda]}_{\mathrm{H}}(\mathsf{A}) = \Pr\left[\ \begin{array}{c} K \xleftarrow{\$} \mathcal{K}\ ;\ M \xleftarrow{\$} \{0,1\}^\lambda \\ Y \leftarrow \mathrm{H}(K, M)\ ;\ M' \xleftarrow{\$} \mathsf{A}(K, Y) \end{array}\ :\ \mathrm{H}(K, M') = Y\ \right]$$

For $\mathrm{atk} = \mathrm{Coll}$, we say that H is (t,ϵ) atk secure if $\mathbf{Adv}^{\mathrm{atk}}_{\mathrm{H}}(\mathsf{A}) < \epsilon$ for all adversaries A running in time at most t. For $\mathrm{atk} \in \{\mathrm{Sec}, \mathrm{Pre}\}$, we say that H is

(t, ϵ) atk secure if $\mathbf{Adv}_{\mathrm{H}}^{\mathrm{atk}[\lambda]}(\mathsf{A}) < \epsilon$ for all adversaries A running in time at most t and for all $\lambda \in \mathbb{N}$ such that $\{0,1\}^{\lambda} \subseteq \mathcal{M}$.

Our security claims in the random oracle model consider $(q_{\mathrm{RO}}, \epsilon)$ atk-security, where q_{RO} is the total number of queries that the adversary A makes to the a random oracle. In the same model, we assume that the compression function $\mathrm{F} : \{0,1\}^{b+n} \rightarrow \{0,1\}^{n}$ behaves as a random oracle. That means F is chosen uniformly at random from the set of all functions with the respective domain and range space and is publicly computable function.

Security preservation. Our goal is to build an infinite-domain hash function family H out of a limited-domain compression function F so that the hash function "inherits" its Coll and Sec security from the natural analogues of these properties for F. For atk = Sec, we say that H *preserves* atk security if H is (t, ϵ) atk secure whenever F is (t', ϵ') atk secure, for some well-specified relation between $t, t', \epsilon, \epsilon'$. For the case of Coll, we have to be more careful because, as pointed out before, (t, ϵ)-Coll security cannot be defined for the keyless compression function F. Rather, we follow Rogaway [15] by saying that collision resistance is preserved if, for an explicitly given Coll adversary A against H, there exists a corresponding, explicitly specified Coll adversary B, as efficient as A, that finds collisions for F.

3 The Basic Construction

3.1 The $\mathcal{BCM}$ Hash Function

In this section we present our hash mode. We refer to it as the backwards chaining mode, or the $\mathcal{BCM}$ hash (see Fig. 1).

The Hash Function. The $\mathcal{BCM}_{\mathrm{F}}$ hash uses a fixed-input-length compression function $\mathrm{F} : \{0,1\}^{b+n} \rightarrow \{0,1\}^{n}$ where $b \geq n$ and takes as inputs a message M of arbitrary length and a key $K = K_1 \| K_2 \| K_3$ of fixed length $(b + 2n)$ bits, where $|K_2| = b$ and $|K_1| = |K_3| = n$. For security and practical reasons we set a bound on the minimal and maximal message length λ, or $n < \lambda < 2^c$ where typically $c = 64$ and $c < n$. The message is preprocessed with a standard MD

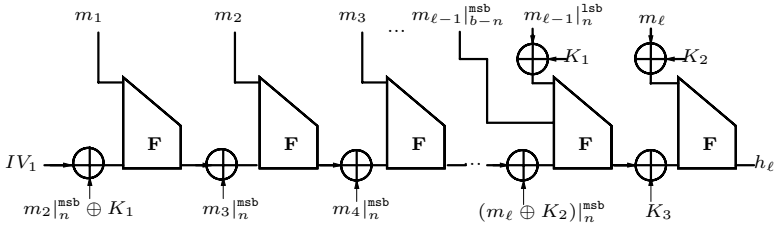

Fig. 1. The $\mathcal{BCM}$ Construction. The message M is MD strengthened. K_1, K_2 and K_3 are randomly chosen and fixed keys of length n, b and n bits, respectively.

strengthening [12]. That is, a single 1 bit is appended to the message M followed by as many zeros as needed and the binary encoding of $|M|$ in 64 bits. We denote the MD padding and strengthening function by pad and $m_1 \| \ldots \| m_\ell \leftarrow \mathsf{pad}(M)$, such that $|m_i| = b$ for $i = 1$ to ℓ.

The $\mathcal{BCM}_{\mathrm{F}}$ hash function can be described as follows. It XORs the key K_1 and the most significant n bits of block m_2 with the fixed initial chaining variable IV_1 (e.g. $IV_1 = 0^n$). The message block m_1 together with the resulting value from the XOR computation form the input to the first application of F. The current-in-line message block m_i and the chaining variable h_{i-1} XORed with the most significant n bits of the next-in-line message block m_{i+1} are the following inputs to the compression function F in the iteration for $i = 1$ to $\ell - 2$.

The one but last block is interpreted differently than the rest of the message blocks. Here the difference is that the least significant n bits of $m_{\ell-1}$ are XORed with the key K_1, while the chaining variable $h_{\ell-2}$ is XORed with $K_2|_n^{\mathtt{msb}}$ and $m_\ell|_n^{\mathtt{msb}}$. The order of processing the inputs is preserved also in the $(\ell - 1)$st block. The final input to the last compression function is provided by the last message block m_ℓ and the chaining variable $h_{\ell-1}$ XORed with keys K_2 and K_3, respectively.

We describe our construction in pseudocode below (Alg. 1) and give a graphical representation in Fig. 1.

Algorithm 1. $\mathcal{BCM}_{\mathrm{F}}(K, M)$:

$m_1 \| \ldots \| m_\ell \leftarrow \mathsf{pad}(M)$
$h_0 = IV_1$, $g_1 = h_0 \oplus K_1 \oplus m_2|_n^{\mathtt{msb}}$
$h_1 = \mathrm{F}(m_1, g_1)$
for $i = 2$ to $\ell - 2$ **do**
  $g_{i-1} = m_{i+1}|_n^{\mathtt{msb}} \oplus h_{i-1}$
  $h_i = \mathrm{F}(m_i, g_{i-1})$
end for
$g_{\ell-2} = (K_2 \oplus m_\ell)|_n^{\mathtt{msb}} \oplus h_{\ell-2}$
$h_{\ell-1} = \mathrm{F}(m_{\ell-1}|_{b-n}^{\mathtt{msb}} \| (m_{\ell-1}|_n^{\mathtt{lsb}} \oplus K_1), g_{\ell-2})$
$h_\ell = \mathrm{F}(m_\ell \oplus K_2, h_{\ell-1} \oplus K_3)$
return h_ℓ

The $\mathcal{BCM}$ hash of a single strengthened message block m_1 is computed as $h_1 = \mathrm{F}(m_1 \oplus K_2, IV_1 \oplus K_1 \oplus K_3)$. And when the message is two blocks long, then $h_1 = \mathrm{F}(m_1|_{b-n}^{\mathtt{msb}} \| m_1|_n^{\mathtt{lsb}} \oplus K_1, IV_1 \oplus K_1 \oplus (m_2 \oplus K_2)|_n^{\mathtt{msb}})$ and the final output hash is computed as $h_2 = \mathrm{F}(m_2 \oplus K_2, h_1 \oplus K_3)$.

Efficiency. The $\mathcal{BCM}_{\mathrm{F}}$ hash mode is a streaming hashing mode that compared to the known MD mode delays the processing with n bits in start-up time. Also as in the MD hash function a single message block is processed per call to the compression function F. Although we lack any concrete efficiency measurements, we expect a small loss in efficiency (compared to MD) due to the constant storage of extra n bits in memory.

Discussion on the Design Choices.We choose to XOR IV_1 with the key K_1 to provide additional randomization on the initialization value. The rest of the XOR choices, namely XORing the chaining variables with the most significant n bits of the incoming message blocks provide the randomization on the chaining values necessary for the Sec security preservation. Also, to achieve the Sec security we have to disallow any fixed inputs introduced by the message padding and strengthening. Hence, we XOR $m_{\ell-1}|_n^{\mathtt{lsb}}$ and m_ℓ with the keys K_1 and K_2, respectively, and we additionally use the key K_3 to randomize the final chaining hash value $h_{\ell-1}$. When F is modeled as a random oracle, the Pre property of $\mathcal{BCM}_{\mathrm{F}}$ is easily satisfiable as long as the message has a minimal length of n bits. An interesting observation is that none of the applied randomization techniques contributes for a Pre preservation in the standard security model.

3.2 Possible Variants

We discuss modifications on the basic $\mathcal{BCM}$ construction with respect to the order of input values to the compression function F. The variants include the different chaining iterations on F, such that F takes as inputs any ordering of: $(A_1 = m_1|_n^{\mathtt{msb}}, B_1 = m_1|_{b-n}^{\mathtt{lsb}}, C_1 = IV_1 \oplus K_1 \oplus m_2|_n^{\mathtt{msb}})$, $(A_i = m_i|_n^{\mathtt{msb}}, B_i = m_i|_{b-n}^{\mathtt{lsb}}, C_i = h_{i-1} \oplus m_{i+1}|_n^{\mathtt{msb}})$ for $i = 2$ to $\ell - 2$, $(A_{\ell-1} = m_{\ell-1}|_n^{\mathtt{msb}}, B_{\ell-1} = m_{\ell-1}|_{n+1}^{b-2n} \| m_{\ell-1}|_{b-2n+1}^{b-n} \oplus K_1, C_{\ell-1} = h_{\ell-2} \oplus (m_\ell \oplus K_2)|_n^{\mathtt{msb}})$ and finally $(A_\ell = (m_\ell \oplus K_2)|_n^{\mathtt{msb}}, B_\ell = (m_\ell \oplus K_2)|_{b-n}^{\mathtt{lsb}}, C_\ell = h_{\ell-1} \oplus K_3)$. The indices denote the position of the input values in the iteration, e.g. (A_1, B_1, C_1) forms the set of input values to the first application of F. There are at most six permuted input sets to F (per call to F). As long as the inputs in the final call to F are ordered identically for messages of any arbitrary length, then the security properties of the basic $\mathcal{BCM}$ carry through to any chaining iteration that switches the input wires to F in any chosen, but specified order.

Let $S_i^1 = \{A_i, B_i, C_i\}$ for $i = 1$ to ℓ be the sets containing the input values to F of the same index i. We then define the sets S_i^j for $j = 2$ to 6 and $i = 1$ to ℓ to be the rest of the possible orderings of the base set S_j^1, or these are $S_i^2 = \{A_i, C_i, B_i\}$, $S_i^3 = \{C_i, A_i, B_i\}$, $S_i^4 = \{C_i, B_i, A_i\}$, $S_i^5 = \{B_i, C_i, A_i\}$ and $S_i^6 = \{B_i, A_i, C_i\}$. Let $P_i^j : S_i^1 \rightarrow S_i^j$ where $j = 1$ to 6 and $i = 1$ to ℓ. With P_i^a we then denote any arbitrarily chosen mapping from S_i^1 to S_i^j for any $j = 1$ to 6 (i is a fixed input parameter), while P^f stands for the final mapping from S_ℓ^1 to S_ℓ^j for some randomly chosen and fixed j.

The $\mathcal{GBCM}$ hash of a 1-block message m_1 is $h_1 = \mathrm{F}(P_1^a(A_1, B_1, C_1))$ with $(A_1 = (m_1 \oplus K_2)|_n^{\mathtt{msb}}, B_1 = (m_1 \oplus K_2)|_{b-n}^{\mathtt{lsb}}, C_1 = IV_1 \oplus K_1 \oplus K_3)$.
The $\mathcal{GBCM}$ hash of a 2-block strengthened message is $h_2 = \mathrm{F}(P^f(A_2, B_2, C_2))$ for $(A_2 = (m_2 \oplus K_2)|_n^{\mathtt{msb}}, B_2 = (m_2 \oplus K_2)|_{b-n}^{\mathtt{lsb}}, C_2 = h_1 \oplus K_3)$ where $h_1 = \mathrm{F}(P_1^a(A_1, B_1, C_1))$ and $(A_1 = m_1|_n^{\mathtt{msb}}, B_1 = m_1|_{n+1}^{b-n} \| m_1|_n^{\mathtt{lsb}} \oplus K_1, C_1 = IV_1 \oplus K_1 \oplus (m_2 \oplus K_2)|_n^{\mathtt{msb}})$.

We then summarize the variants of $\mathcal{BCM}$ by exhibiting a generalized $\mathcal{GBCM}$ construction and describe it in pseudocode in Algorithm 2.

Algorithm 2. $\mathcal{GBCM}_{\mathrm{F}}(K, M)$:

$m_1 \| \ldots \| m_\ell \leftarrow \mathsf{pad}(M)$
$h_0 = IV_1$,
for $i = 1$ to $\ell - 1$ **do**
  $h_i = \mathrm{F}(P_i^a(A_i, B_i, C_i))$
end for
$h_\ell = \mathrm{F}(P^f(A_\ell, B_\ell, C_\ell))$
return h_ℓ

4 Property Preservation of the $\mathcal{BCM}$ ($\mathcal{GBCM}$) Construction

In this section we provide the full proofs for Coll and Sec security preservation in the standard security model and we show Pre security of the $\mathcal{BCM}_{\mathrm{F}}$ construction when the compression function is instantiated as a random oracle. We provide the proofs of $\mathcal{GBCM}$ in the Appendix.

Theorem 1. *If there exists an explicitly given adversary* A *that* (t, ϵ)*-breaks the* Coll *security of* $\mathcal{BCM}_{\mathrm{F}}$ *(*$\mathcal{GBCM}_{\mathrm{F}}$*), then there exists an explicitly given adversary* B *that* (t', ϵ')*-breaks the* Coll *security of* F *for* $\epsilon' \geq \epsilon$ *and* $t' \leq t + 2\ell \cdot \tau_{\mathrm{F}}$*. Here,* τ_{F} *is the time required for the evaluation of* F *and* $\ell = \lceil (\lambda + 65)/b \rceil$ *where* λ *is the maximum message length of the two messages output by* A*.*

Proof. Given a Coll adversary A against the iterated hash $\mathcal{BCM}_{\mathrm{F}}$, we construct a Coll adversary B against the compression function F. B generates at random a key $K \xleftarrow{\$} \{0,1\}^{b+2n}$ with $K = K_1 \| K_2 \| K_3$ where $|K_1| = |K_3| = n$ and $|K_2| = b$. B runs A on input K. Finally, A outputs a colliding pair of messages M and M', such that $\mathcal{BCM}_{\mathrm{F}}(K, M) = \mathcal{BCM}_{\mathrm{F}}(K, M')$. We investigate the following two cases:

1. If $|M| \neq |M'|$, then the inputs to the last compression function differ (due to the present message length encoding in m_ℓ) and therefore a collision on the final F occurs, or $m_\ell \oplus K_2 \neq m'_{\ell'} \oplus K_2$ where $\mathrm{F}(m_\ell \oplus K_2, h_{\ell-1} \oplus K_3) = \mathrm{F}(m'_{\ell'} \oplus K_2, h'_{\ell'-1} \oplus K_3)$. B then outputs $(m_\ell \oplus K_2, h_{\ell-1} \oplus K_3)$ and $(m'_{\ell'} \oplus K_2, h'_{\ell'-1} \oplus K_3)$ as a valid colliding pair.
2. Else if $|M| = |M'|$, then $\ell = \ell'$. If $m_\ell \oplus K_2 \| h_{\ell-1} \oplus K_3 \neq m'_\ell \oplus K_2 \| h'_{\ell-1} \oplus K_3$, then a collision occurs again in the last application of F. Else B proceeds in the following way.
   B parses the inputs to the $(\ell-1)$st application of F as $(m_{\ell-1}|_{b-n}^{\mathtt{msb}} \| (m_{\ell-1}|_n^{\mathtt{lsb}} \oplus K_1), g_{\ell-2})$ and $(m'_{\ell-1}|_{b-n}^{\mathtt{msb}} \| (m'_{\ell-1}|_n^{\mathtt{lsb}} \oplus K_1), g'_{\ell-2})$. If these inputs differ, then they constitute a valid collision pair for B, else $g_{\ell-2} = g'_{\ell-2}$ and hence $h_{\ell-2} = h_{\ell-2}$ because of the previous equality for $m_\ell \oplus K_2|_n^{\mathtt{msb}} = m'_\ell \oplus K_2|_n^{\mathtt{msb}}$. Following the iteration principle B parses the previous inputs as $m_{\ell-2} \| g_{\ell-3}$ and $m'_{\ell-2} \| g'_{\ell-3}$ and proceeds in the same manner backwards.

The inequality of the message inputs M and M' guarantees the existence of an index $i > 0$, such that $m_i \| g_{i-1} \neq m'_i \| g'_{i-1}$ where $\mathrm{F}(m_i \| g_{i-1}) = \mathrm{F}(m'_i \| g'_{i-1})$. B outputs then the colliding pair $(m_i \| g_{i-1}, m'_i \| g'_{i-1})$ for the $\max(i)$ satisfying the former statement.

Whenever A succeeds, then B also succeeds with the same advantage. The time complexity of B is at most the time complexity of A plus two evaluations of $\mathcal{BCM}_{\mathrm{F}}$ over messages M and M' taking time $2\ell \cdot \tau_{\mathrm{F}}$. □

Theorem 2. *For* atk = Sec*, if the compression function* F *is* (t', ϵ') atk *secure, then the iterated function* $\mathcal{BCM}_{\mathrm{F}}$ *(*$\mathcal{GBCM}_{\mathrm{F}}$*) is* (t, ϵ) atk *secure for* $\epsilon \leq \ell \cdot \epsilon'$ *and* $t \geq t' - 2\ell \cdot \tau_{\mathrm{F}}$*. Here,* τ_{F} *is the time required for the evaluation of* F *and* $\ell = \lceil (\lambda + 65)/b \rceil$ *where* λ *is the maximum message length of the two messages output by* A*.*

Proof. Given a Sec[λ] adversary A against $\mathcal{BCM}_{\mathrm{F}}$, we construct a Sec adversary B against the compression function F. B receives a random challenge message $m \| h$. For a randomly chosen index i the goal of the adversary B is to construct a challenge message M and a key K, which have B's challenge message $m \| h$ embedded, such that when A outputs its second preimage message M', then a collision for M and M' can be found at the ith block of M. To simulate A's view correctly, however, B generates M and K, such that they are uniformly distributed. The proof goes as follows.

B chooses a random index $i \xleftarrow{\$} \{1, \ldots, \ell = \lceil (\lambda + 65)/b \rceil\}$ and a random message M of length λ. B has now to successfully embed his challenge $m \| h$ at position i in the target strengthened message $m_1 \| \ldots \| m_\ell \leftarrow \mathsf{pad}(M)$ and in the chaining iterative portion of $\mathcal{BCM}_{\mathrm{F}}(K, M)$. Let $\hat{M} = m_1 \| \ldots \| m_\ell$ be the strengthened message M. Depending on the outcomes for i, B takes its decisions as follows:

1. If $i = 1$, then B sets $m_1 \leftarrow m$ and $K_1 \leftarrow IV_1 \oplus h \oplus m_2|_n^{\mathtt{msb}}$. Except block m_1, the rest of $\hat{M}$ is unaltered. B chooses $K_2 \| K_3 \xleftarrow{\$} \{0,1\}^{b+n}$. Two special cases arise in the case when $\lambda < b - 65$ or $\lambda < 2b - 65$. In the former case, B proceeds as described below for $i = \ell$, and in the latter case as for $i = \ell - 1$.
2. If $i \in \{2, \ldots, \ell - 2\}$, then B continues as follows. B sets $m_i \leftarrow m$ and computes the intermediate chaining value h_{i-1} with $K_1 \xleftarrow{\$} \{0,1\}^n$. B sets $m_{i+1}|_n^{\mathtt{msb}} \leftarrow h_{i-1} \oplus h$. Then B chooses the keys $K_2 \| K_3$ at random. Except modifying blocks m_i and $m_{i+1}|_n^{\mathtt{msb}}$, B leaves the rest of $\hat{M}$ unaltered.
3. If $i = \ell - 1$, then B sets $m_{\ell-1}|_{b-n}^{\mathtt{msb}} \leftarrow m|_{b-n}^{\mathtt{msb}}$ and $K_1 \leftarrow (m_{\ell-1} \oplus m)|_n^{\mathtt{lsb}}$, and computes the intermediate chaining value $h_{\ell-2}$. $h_{\ell-2}$ is set to $IV_1 \oplus K_1$ when $b - 65 \leq \lambda < 2b - 65$. B sets $K_2|_n^{\mathtt{msb}} \leftarrow h_{\ell-2} \oplus h \oplus m_\ell|_n^{\mathtt{msb}}$. Then B chooses at random $K_2|_{b-n}^{\mathtt{lsb}} \| K_3 \xleftarrow{\$} \{0,1\}^b$. With the exception of $m_{\ell-1}|_{b-n}^{\mathtt{msb}}$, the rest of $\hat{M}$ remains unaltered.
4. If $i = \ell$, then B chooses at random $K_1 \xleftarrow{\$} \{0,1\}^n$ and computes the intermediate hash value $h_{\ell-1}$. Note that if $\lambda < b - 65$, then the chaining value is computed as $IV_1 \oplus K_1$. B then sets $K_3 \leftarrow h \oplus h_{\ell-1}$ and $K_2 \leftarrow m \oplus m_\ell$. $\hat{M}$ remains unchanged.

After $m\|h$ is successfully embedded, B proceeds by running A on inputs message M where $M = \mathsf{pad}^{-1}(\hat{M})$(the non-strengthened version of $\hat{M}$ with the applied modifications) and key $K = K_1\|K_2\|K_3$. Note that both M and K are uniformly distributed. Initially B chooses uniformly at random M and then modifies some of its blocks as prescribed in the former cases. However, the modified blocks of M are assigned only independent random values. Hence, the resulting final challenge message M is also uniformly distributed. The key K is constructed in a similar way.

On inputs M and K A returns a second preimage message M', such that $\mathcal{BCM}_{\mathrm{F}}(K, M) = \mathcal{BCM}_{\mathrm{F}}(K, M')$. For the rest of the proof B acts identically as in the Coll proof. With probability $1/\ell$ B finds the colliding pair at the correct position i (at which B embedded $m\|h$) and outputs the colliding inputs for F as its valid second preimage. If A succeeds with advantage ϵ, then B also succeeds with advantage ϵ/ℓ. The time complexity of B is at most the time complexity of A plus two evaluations of $\mathcal{BCM}_{\mathrm{F}}$. This completes the proof. □

Theorem 3. *If the compression function* F *is instantiated as a random oracle, then the iterated function* $\mathcal{BCM}_{\mathrm{F}}$ *(*$\mathcal{GBCM}_{\mathrm{F}}$*) is* $(q_{\mathrm{RO}}, \epsilon)$ $\mathrm{Pre}[\lambda]$ *secure where* $\epsilon \leq q_{\mathrm{RO}}/2^n$ *and* q_{RO} *is the number of queries to the random oracle.*

Proof. Let A be a $\mathrm{Pre}[\lambda]$ adversary on the iterated hash function $\mathcal{BCM}_{\mathrm{F}}$. Given a challenge hash value Y and key $K = K_1\|K_2\|K_3$, the goal of A is to invert Y, which is computed as $Y = \mathcal{BCM}_{\mathrm{F}}(K, M)$ for a randomly chosen message $M \xleftarrow{\$} \{0,1\}^{\lambda}$ and a key $K \xleftarrow{\$} \mathcal{K}$.

We investigate the following two cases: 1. $n < \lambda \leq b - 65$ and 2. $\lambda > b - 65$. In the first case $Y = \mathrm{F}(m_1 \oplus K_2, IV_1 \oplus K_1 \oplus K_3)$. The adversary A knows the message length, respectively the applied strengthening bits, the fixed IV_1 value and the random key values $K_1\|K_2\|K_3$. However, A has no information of at least n bits of the message input m_1. Thus, the only way to find a valid preimage message for $\mathcal{BCM}_{\mathrm{F}}$ is to exhaustively query the random oracle F on chosen inputs for the missing part of m_1. The probability to find the correct preimage message per single query is $1/2^n$ and after q_{RO} queries to the random oracle A succeeds to invert Y with probability $q_{\mathrm{RO}}/2^n$.

In the second case $Y = \mathrm{F}(m_\ell \oplus K_2, h_{\ell-1} \oplus K_3)$. Here A knows the keys K_2 and K_3, the message length λ and the respective strengthening bits used in the last message block m_ℓ. Still, B does not know the intermediate chaining value $h_{\ell-1}$. Again as in the former case, A needs to invert Y and its success ϵ is bound by $q_{\mathrm{RO}}/2^n$. □

A Pre Counterexample in the Standard Model. Surprisingly, the presented $\mathcal{BCM}_{\mathrm{F}}$ does not provide Pre property preservation when the compression function is a Pre secure hash and not modeled as a random oracle. Here we provide a counterexample compression function, which is Pre secure as long as the underlying compressing function is also Pre secure.

Let F be defined as $\mathrm{F}(m\|h) = \mathrm{CE}_1(m)$. If $\mathrm{CE}_1 : \{0,1\}^b \to \{0,1\}^n$ is (t', ϵ') Pre secure compressing function, then F is also (t, ϵ) Pre secure function with $\epsilon \leq \epsilon'$ and $t \geq t'$.

A Pre[λ] adversary A on the iterated hash function $\mathcal{BCM}_{\mathrm{F}}$ succeeds in constant time with probability one in breaking the Pre[λ] security. A is given a challenge hash value $Y = \mathcal{BCM}_{\mathrm{F}}(M, K)$ for a random $M \xleftarrow{\$} \{0,1\}^{\lambda}$ and a random key $K \leftarrow \mathcal{K}$ with $K = K_1 \| K_2 \| K_3$. A succeeds by outputting any message M' of length λ, because all these messages result in $Y = \mathrm{F}(m_\ell \oplus K_2, h_{\ell-1} \oplus K_3) = \mathrm{CE}_1(m_\ell \oplus K_2)$. Here we made an assumption that λ is such that m_ℓ consists only of padding and strengthening bits.

4.1 Second Preimage Attacks Beyond $2^{n-\ell}$

From [11] and [1] we know that earlier Merkle-Damgård based constructions are prone to Sec attacks in a bit more that $2^{n-\ell}$ compression function calls when the target messages are of size 2^ℓ blocks. The latter attacks apply to our scheme given that the target messages are of a specific format. Let the challenge messages be parsed as a sequence of b-bit blocks. When these contain fixed and predictable message chunks in their n most significant bits (e.g. $m_i|_n^{\mathtt{msb}} = 0^n$), the mentioned attacks can be mounted on our hash construction. But even then the attacks are in no contradiction with our claimed Sec security result. In our Sec security proof we loose a factor of ℓ (number of message blocks), while the attacks are valid in the estimated security gap (between the exhibited $2^{n-\ell}$ and the ideal 2^n Sec security).

To build either an expandable message or a diamond structure used in the attacks, an adversary searches for collisions on F. These are possible by going over different values only in the least significant $(b-n)$ bits of the message blocks chosen by the adversary. The adversary then commits to an intermediate hash value h_i. Next, in both the expandable message attack [11] and the diamond structure [1] second preimage attacks, the adversary has to connect from h_i to a chaining value in the target message M. Here the requirement for a specific message format comes into play. If the message blocks differ in their most significant n bits, the adversary's probability to connect correctly is small. That is because he has to have predicted in advance $m_j|_n^{\mathtt{msb}}$, given that he successfully connects to a chaining value h_j. The best adversarial strategy here is to exploit message blocks repetitions in their most significant n bits. Then if all these are equal, the attacks become feasible in approximately $2^{n-\ell}$ steps (compression function calls).

One way to fully block this type of attacks is to XOR the chaining values with the output of a function f that takes as input the complete forth-coming message block of b bits. In the iteration we would replace the $m_i|_n^{\mathtt{msb}}$ with $f(m_i)$ for all $i = 1$ to ℓ. To achieve the property preservation the function f has to be instantiated as a random oracle, which turns the suggested scheme into a less efficient variant (linear number of calls to RO) of the ROX [2] hash.

5 Security Discussion or Where Theory Meets Practice

Our scheme preserves the Coll security of the compression function F due to the MD strengthening and achieves the $2^{n/2}$ security level if F is an ideal function.

The Sec security property is preserved through the randomization of the final inputs of F with keys K_2 and K_3, and the intermediate hash values with parts of the message blocks.

Notice that the standard Sec definition we use for the latter result assumes the uniform distribution U on the message space $\mathcal{M}$. This allows us to extract randomness from the challenge messages, rather than adding extra key material (e.g. log number of keys in Shoup's hash). Where is the caveat here? When the messages are not sufficiently random, the $\mathcal{BCM}$ hash scheme does not reach the claimed Sec security level. More precisely, we do not need the randomness from the whole message source but we extract it only from the n most significant bits of every b-bit chunk of message. Then messages without sufficient entropy in those most significant bits can introduce Sec weaknesses in our scheme, as we show. Non-uniform distributions that allow for such attacks are some concrete distributions of messages with low entropy. Next we exhibit such a counterexample.

The Low Entropy Messages Counterexample. In our counterexample we construct a contrived Sec secure compression function and specify the format of the messages that occur with the highest probability according the challenge messages distribution.

Let the challenge message space $\mathcal{M}^{D_l}$ be assigned the distribution D_l. Here we define $\mathcal{M}^{D_l} = \{0,1\}^{\lambda}$ where $\lambda > 2b-65$. According to D_l for any $m_i|_n^{\mathtt{lsb}}$ with $i = 2$ to ℓ ($\ell = \lceil(\lambda)/b\rceil$) the messages $m_1\|0^n\|m_2|_{b-n}^{\mathtt{lsb}}\|\ldots\|0^n\|m_\ell|_{b-n}^{\mathtt{lsb}}$ appear with high probability $(1-\epsilon')$ while all the rest of the messages occur with negligible probability ϵ'. The most frequent challenge messages contain n bits of 0s in the most significant bits of their b-bit blocks. The counterexample compression function we use is similar to the one from Theorem 3.2 [2].

Theorem 4. *If there exists a* (t,ϵ) Sec *secure function* $\mathrm{G} : \{0,1\}^{b+n} \rightarrow \{0,1\}^{n-1}$, *then there exists a* $(t, \epsilon - 1/2^{n-1})$ Sec *secure compression function* $\mathrm{CE}_2 : \{0,1\}^{b+n} \rightarrow \{0,1\}^n$ *and an adversary* A *running in constant time with* $\mathrm{Sec}[\lambda, D_l]$*-advantage* $(1-\epsilon')$ *in breaking* $\mathcal{BCM}_{\mathrm{CE}_2}$ *for any challenge message* $M \in \mathcal{M}^{D_l}$ *chosen according to the distribution* D_l.

Note that we additionally parameterize the adversarial advantage by the message space distribution D_l and that ϵ' is the probability for messages different from the specified format to be chosen from $\mathcal{M}^{D_l}$.

Proof. Our CE_2 is given by

$$\begin{aligned}\mathrm{CE}_2(m\|h) &= 0^n \qquad\qquad \text{if } h = 0^n \text{ or } m|_n^{\mathtt{msb}} = 0^n \\ &= \mathrm{G}(m\|h) \parallel 1 \text{ otherwise .}\end{aligned}$$

If G is (t,ϵ) Sec secure, then CE_2 is $(t, \epsilon - 1/2^{n-1})$ Sec secure; we refer to the full version [3] for the proof.

According to the distribution D_l the messages of the specified format are chosen with high probability $(1-\epsilon')$ as target messages. Then for any random key $K \xleftarrow{\$} \{0,1\}^{b+2n}$ and target message $M = m_1\|0^n\|m_2|_{b-n}^{\mathtt{lsb}}\|\ldots\|0^n\|m_\ell|_{b-n}^{\mathtt{lsb}}$, a $\mathrm{Sec}[\lambda, D_l]$ adversary A finds a second preimage message M', such that $|M| = |M'|$

and M' is of the same format as M. Let M' differ from M only in the least significant $(b-n)$ bits of their second blocks. Then the chaining values in the computation of M and M' are equal immediately after the second application of CE_2. For any $\lambda > 2b - 65$ the chaining value 0^n is propagated further in the chain. Finally $\mathcal{BCM}_{\mathrm{CE}_2}(K, M) = \mathcal{BCM}_{\mathrm{CE}_2}(K, M')$. □

We admit that this is a particularly contrived counterexample for a low entropy message distribution. The minimal requirement on the message structure with this type of counterexample compression functions is a repetition in the most significant n bits of two adjacent b-bit message blocks. Such counterexamples are especially problematic for low entropy message distributions and are also valid for high entropy message distributions. However, in the latter case, the probability for such challenge messages to be chosen is not high on average and we cannot exhibit an efficient Sec adversary.

6 Concluding Discussion

In our opinion the Sec security preservation is one of the hardest security notions to satisfy in an iterative hash mode. At the cost of a logarithmic number of keys to randomize the chaining values and an additional constant b-bit key to randomize the message blocks, Shoup's hash [17] could be modified to also achieve Sec preservation in the standard model. Notice, however, that once the keys are fixed, we can always identify non-uniform distributions for which contrived counterexamples are possible (even if we increase the fixed keys for the randomization of the message blocks from a constant to linear in the message length). One way to avoid this problem is to introduce randomization per message, known also as salting. It is therefore an interesting question to identify the conditions that such a message randomization transform needs to satisfy in order to provide Sec preservation for any target message distribution.

On the other hand, the question of correctly formalizing and satisfying Sec security properties that take into account biased challenge message distributions may be practically relevant. Practical message distributions that deviate from uniform allow for predictability of certain target messages and in our view are a shift from the Sec to the known target collision resistance (TCR/UOWH/eSec) property. Another interesting problem may then be to find ways to achieve Sec security for any message distribution with an efficient hash construction that uses a minimal amount of key material. In our view, one possible way to go around this problem is to correctly identify new assumptions on the compression function.

Acknowledgements

We would like to thank Gregory Neven, Sebastiaan Indesteege and Markulf Kohlweiss for the helpful discussions. Also, we would like to thank the anonymous referees for their useful feedback. This work was supported in part by

the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT, and in part by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy). The first author is supported by a Ph.D. Fellowship from the Flemish Research Foundation (FWO - Vlaanderen).

References

1. Andreeva, E., Bouillaguet, C., Fouque, P.-A., Hoch, J.J., Kelsey, J., Shamir, A., Zimmer, S.: Second preimage attacks on dithered hash functions. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 270–288. Springer, Heidelberg (2008)
2. Andreeva, E., Neven, G., Preneel, B., Shrimpton, T.: Seven-property-preserving iterated hashing: ROX. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 130–146. Springer, Heidelberg (2007)
3. Andreeva, E., Neven, G., Preneel, B., Shrimpton, T.: Seven-property-preserving iterated hashing: ROX. Cryptology ePrint Archive, Report 2007/176 (2007)
4. Bellare, M., Rogaway, P.: Collision-resistant hashing: Towards making uOWHFs practical. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 470–484. Springer, Heidelberg (1997)
5. Bellare, M., Ristenpart, T.: Multi-property-preserving hash domain extension and the EMD transform. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 299–314. Springer, Heidelberg (2006)
6. Bellare, M., Ristenpart, T.: Hash functions in the dedicated-key setting: Design choices and MPP transforms. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 399–410. Springer, Heidelberg (2007)
7. Coron, J.-S., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-damgård revisited: How to construct a hash function. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 430–448. Springer, Heidelberg (2005)
8. Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. SIAM Journal on Computing 33(1), 167–226 (2003)
9. Damgård, I.B.: A design principle for hash functions. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 416–427. Springer, Heidelberg (1990)
10. Halevi, S., Krawczyk, H.: Strengthening digital signatures via randomized hashing. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 41–59. Springer, Heidelberg (2006)
11. Kelsey, J., Schneier, B.: Second preimages on n-bit hash functions for much less than 2^n work. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 474–490. Springer, Heidelberg (2005)
12. Lai, X., Massey, J.L.: Hash functions based on block ciphers. In: Rueppel, R.A. (ed.) EUROCRYPT 1992. LNCS, vol. 658, pp. 55–70. Springer, Heidelberg (1993)
13. Merkle, R.C.: One way hash functions and DES. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 428–446. Springer, Heidelberg (1990)
14. NIST. Announcing request for candidate algorithm nominations for a new cryptographic hash algorithm (sha–3) family (2007), `http://csrc.nist.gov/groups/ST/hash`
15. Rogaway, P.: Formalizing human ignorance. In: Nguyên, P.Q. (ed.) VIETCRYPT 2006. LNCS, vol. 4341, pp. 211–228. Springer, Heidelberg (2006)

16. Rogaway, P., Shrimpton, T.: Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 371–388. Springer, Heidelberg (2004)
17. Shoup, V.: A composition theorem for universal one-way hash functions. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 445–452. Springer, Heidelberg (2000)
18. Wang, X., Yu, H.: How to break MD5 and other hash functions. In: Cramer, R. (ed.) EUROCRYPT 2005, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)
19. Wang, X., Yin, Y.L., Yu, H.: Finding collisions in the full SHA-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005)
20. Wang, X., Yu, H., Yin, Y.L.: Efficient collision search attacks on SHA-0. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 1–16. Springer, Heidelberg (2005)

A $\mathcal{GBCM}$ Proofs

A.1 Coll Proof of Theorem 1 for $\mathcal{GBCM}$

Given a Coll adversary A against $\mathcal{GBCM}_{\mathrm{F}}$, we construct a Coll adversary B against the compression function F. B generates at random a key $K \xleftarrow{\$} \{0,1\}^{b+2n}$ with $K = K_1 \| K_2 \| K_3$ where $|K_1| = |K_3| = n$ and $|K_2| = b$. B runs A on input K. Finally, A outputs a colliding pair of messages M and M', such that $\mathcal{BCM}_{\mathrm{F}}(K, M) = \mathcal{BCM}_{\mathrm{F}}(K, M')$. We investigate the following two cases:

1. If $|M| \neq |M'|$, then the inputs to the last compression function differ (due to the present message length encoding in m_ℓ) and therefore a collision on the final F occurs, or $P^f(A_\ell, B_\ell, C_\ell) \neq P^f(A'_\ell, B'_\ell, C'_\ell)$. Note that the transformation P^f is fixed and identical for messages of arbitrary length. Therefore, a difference in blocks B_ℓ and $B_{\ell'}$ (induced by the applied strengthening) results in difference in the outputs of P^f on the same input wires for F. B outputs $P^f(A_\ell, B_\ell, C_\ell), P^f(A'_\ell, B'_\ell, C'_\ell)$ as a valid colliding pair.
2. Else if $|M| = |M'|$, then $\ell = \ell'$ and the processing of M and M' is symmetric with respect to the inputs of F (the same arbitrary P_j^a for $j = 1$ to $\ell - 1$ is applied at all positions j for both M and M'). Here B proceeds by searching backwards (block-by-block) for distinct F inputs $P_j^a(A_j, B_j, C_j)$ and $P_j^a(A'_j, B'_j, C'_j)$, which result in equal output hash values h_j an h'_j under F. Since $M \neq M'$, then there exists an index $j > 0$, such that $P_j^a(A_j, B_j, C_j) \neq P_j^a(A'_j, B'_j, C'_j)$. Then for the $\max(j)$ that satisfies the inequality, B outputs the corresponding colliding pair $(P_j^a(A_j, B_j, C_j), P_j^a(A'_j, B'_j, C'_j))$. □

A.2 Sec Proof of Theorem 2 for $\mathcal{GBCM}$

Given a Sec[λ] adversary A against $\mathcal{GBCM}_{\mathrm{F}}$, we construct a Sec adversary B against the compression function F. B receives a random challenge message $m \| h$. Then B chooses a random index $i \xleftarrow{\$} \{1, \ldots, \ell = \lceil (\lambda + 65)/b \rceil\}$ and a random

message $M \xleftarrow{\$} \{0,1\}^\lambda$. B has to successfully embed his challenge $m\|h$ in the target strengthened message $m_1\| \ldots \|m_\ell \leftarrow \mathsf{pad}(M)$ and in the chaining iterative portion of $\mathcal{GBCM}_{\mathrm{F}}(K, M)$. Let $\hat{M} = m_1\| \ldots \|m_\ell$ be the strengthened message.

Let $(X_i, Y_i, Z_i) \leftarrow P_i^a(A_i, B_i, C_i)$ for $i = 1$ to $\ell - 1$ and $(X_\ell, Y_\ell, Z_\ell) \leftarrow P^f(A_\ell, B_\ell, C_\ell)$. Then B identifies the type of mapping applied in the respective ith position in the iteration for $i = 1$ to $\ell - 1$

1. if $P_i^a = P_i^1$, then $X_i\|Y_i = m$ and $Z_i = h$.
2. if $P_i^a = P_i^2$, then $X_i\|Z_i = m$ and $Y_i = h$.
3. if $P_i^a = P_i^3$, then $Y_i\|Z_i = m$ and $X_i = h$.
4. if $P_i^a = P_i^4$, then $Z_i\|Y_i = m$ and $X_i = h$.
5. if $P_i^a = P_i^5$, then $Z_i\|X_i = m$ and $Y_i = h$.
6. if $P_i^a = P_i^6$, then $Y_i\|X_i = m$ and $Z_i = h$.

Now if $i = 1$, then a value that equals to h translates to B setting $K_1 \leftarrow IV_1 \oplus h \oplus m_2|_n^{\mathtt{msb}}$. If $\lambda < b - 65$, then B proceeds as in case $i = \ell$.

If $i \in \{2 \ldots \ell - 2\}$, then equality to h translates to B setting $m_{i+1}|_n^{\mathtt{msb}} \leftarrow h_{i-1} \oplus h$ for a randomly chosen $K_1 \xleftarrow{\$} \{0,1\}^n$ (h_{i-1} is the $(i-1)$st intermediate chaining value computed by B). If $\lambda < 2b - 65$, then B proceeds as in case $i = \ell - 1$.

In both these cases B modifies either block m_1, or blocks m_i and $m_{i+1}|_n^{\mathtt{msb}}$ from the message $\hat{M}$. B also chooses at random $K_2\|K_3 \xleftarrow{\$} \{0,1\}^{b+n}$.

If $i = \ell - 1$, then equality of a value to m is equivalent to B setting the most significant $b - n$ bits of it to $m|_{b-n}^{\mathtt{msb}}$ and $K_1 \leftarrow (m \oplus m_{\ell-1})|_n^{\mathtt{lsb}}$. Equality to h here means that B sets $K_2|_n^{\mathtt{msb}} \leftarrow h \oplus h_{\ell-2} \oplus m_\ell|_n^{\mathtt{msb}}$ ($h_{\ell-2}$ is the $(\ell-2)$nd intermediate chaining value computed by B when $\lambda \geq 2b - 65$ and $IV_1 \oplus K_1$ when $b - 65 \leq \lambda < 2b - 65$). A chooses $K_2|_{b-n}^{\mathtt{lsb}}\|K_3 \xleftarrow{\$} \{0,1\}^n$. Only the first $b - n$ bits of block $m_{\ell-1}$ are modified from the originally generated message $\hat{M}$.

If $i = \ell$, then

1. if $P^f = P^1$, then $X_\ell\|Y_\ell = m$ and $Z_\ell = h$.
2. if $P^f = P^2$, then $X_\ell\|Z_\ell = m$ and $Y_\ell = h$.
3. if $P^f = P^3$, then $Y_\ell\|Z_\ell = m$ and $X_\ell = h$.
4. if $P^f = P^4$, then $Z_\ell\|Y_\ell = m$ and $X_\ell = h$.
5. if $P^f = P^5$, then $Z_\ell\|X_\ell = m$ and $Y_\ell = h$.
6. if $P^f = P^6$, then $Y_\ell\|X_\ell = m$ and $Z_\ell = h$.

B chooses at random $K_1 \xleftarrow{\$} \{0,1\}^n$ and computes the intermediate hash value $h_{\ell-1}$. An equality to m means that B then sets $K_2 \leftarrow m \oplus m_\ell$ and equality to h that $K_3 \leftarrow h \oplus h_{\ell-1}$ ($h_{\ell-1}$ is the $(\ell-1)$st intermediate chaining value computed by B when $\lambda \geq b - 65$ and $IV_1 \oplus K_1$ when $\lambda < b - 65$). Except m_ℓ, the rest of $\hat{M}$ remains unchanged.

Now $m\|h$ is successfully embedded and B proceeds by running A on inputs message M where $M = \mathsf{pad}^{-1}(\hat{M})$(the non-strengthened version of $\hat{M}$) and key $K = K_1\|K_2\|K_3$. As in the case of $\mathcal{BCM}$ hash, the message M and key K are uniformly distributed. A returns a second preimage message M', such that $\mathcal{BCM}_{\mathrm{F}}(K, M) = \mathcal{BCM}_{\mathrm{F}}(K, M')$. For the rest of the proof B acts identically as in the Coll proof. With probability $1/\ell$ B finds the colliding pair at the correct

position i (at which B embedded $m\|h$) and outputs the colliding inputs for F as its valid second preimage. This completes the proof. □

A.3 Pre Proof of Theorem 3 for $\mathcal{GBCM}$

Proof. Let A be a Pre[λ] adversary on the iterated hash function $\mathcal{GBCM}_{\mathrm{F}}$. Given a challenge hash value Y and key $K = K_1\|K_2\|K_3$, the goal of A is to invert Y, which is computed as $Y = \mathcal{GBCM}_{\mathrm{F}}(K, M)$ for a randomly chosen message $M \xleftarrow{\$} \{0,1\}^{\lambda}$ and a random key $K \xleftarrow{\$} \mathcal{K}$.

We investigate the following two cases: 1. $n < \lambda \leq b - 65$ and 2. $\lambda > b - 65$. In the first case $Y = \mathrm{F}(P^f(A_1, B_1, C_1))$. A knows λ, respectively the applied strengthening bits, IV_1 and $K_1\|K_2\|K_3$. Thus, A knows at most the input B_1 and C_1 and can derive at most $b - n$ bits from the output of P^f . However, A has no information on at least n bits of the message input $A_1 = m_1$. Thus, the only way to find a valid preimage message for $\mathcal{BCM}_{\mathrm{F}}$ is to exhaustively query the random oracle F on chosen inputs for the missing part of m_1. The probability to find the correct preimage message per single query is $1/2^n$ and therefore after q_{RO} queries to the random oracle A succeeds to invert Y with probability $q_{\mathrm{RO}}/2^n$.

In the second case $Y = \mathrm{F}(P^f(A_\ell, B_\ell, C_\ell))$. Here A knows the keys K_2 and K_3, the message length λ and the respective strengthening bits used in the last message block m_ℓ. A can compute at most $b-n$ bits of $P^f(A_\ell, B_\ell, C_\ell)$. But again B does not know the intermediate chaining value $h_{\ell-1}$ and also $C_\ell = h_{\ell-1} \oplus K_3$, then again as in case one, A can at best try to invert Y. A'a success ϵ is bound by $q_{\mathrm{RO}}/2^n$. □