An extended abstract of this paper appears in Kaoru Kurosawa, editor, *Advances in Cryptology – ASIACRYPT 2007*, volume ???? of *Lecture Notes in Computer Science*, Springer-Verlag, 2007 [ANPS07a]. This is the full version.

Seven-Property-Preserving Iterated Hashing: ROX

Elena Andreeva[1], Gregory Neven[1,2], Bart Preneel[1], Thomas Shrimpton[3,4]

[1] SCD-COSIC, Dept. of Electrical Engineering, Katholieke Universiteit Leuven
Kasteelpark Arenberg 10, B-3001 Heverlee, Belgium
{Elena.Andreeva,Gregory.Neven,Bart.Preneel}@esat.kuleuven.be
[2] Département d'Informatique, Ecole Normale Supérieure
45 rue d'Ulm, 75005 Paris, France
[3] Dept. of Computer Science, Portland State University
1900 SW 4th Avenue, Portland, OR 97201, USA
teshrim@cs.pdx.edu
[4] Faculty of Informatics, University of Lugano
Via Giuseppe Buffi 13, CH-6904 Lugano, Switzerland

Abstract

Nearly all modern hash functions are constructed by iterating a compression function. At FSE'04, Rogaway and Shrimpton [RS04] formalized seven security notions for hash functions: collision resistance (Coll) and three variants of second-preimage resistance (Sec, aSec, eSec) and preimage resistance (Pre, aPre, ePre). The main contribution of this paper is in determining, by proof or counterexample, which of these seven notions is preserved by each of eleven existing iterations. Our study points out that none of them preserves more than three notions from [RS04]. In particular, only a single iteration preserves Pre, and none preserves Sec, aSec, or aPre. The latter two notions are particularly relevant for practice, because they do not rely on the problematic assumption that practical compression functions be chosen uniformly from a family. In view of this poor state of affairs, even the mere existence of seven-property-preserving iterations seems uncertain. As a second contribution, we propose the new Random-Oracle XOR (ROX) iteration that is the first to provably preserve all seven notions, but that, quite controversially, uses a random oracle in the iteration. The compression function itself is *not* modeled as a random oracle though. Rather, ROX uses an auxiliary small-input random oracle (typically 170 bits) that is called only a logarithmic number of times.

Keywords: Cryptographic hash functions, Merkle-Damgård, collision resistance, preimage resistance, second-preimage resistance, provable security.

Contents

1 Introduction

Cryptographic hash functions, publicly computable maps from inputs of arbitrary length to (short) fixed-length strings, have become a ubiquitous building block in cryptography. Almost all cryptographic hash functions are iterative: given a compression function F that takes $(n + b)$ bits of input and produces n bits of output, they process an arbitrary length input by dividing it into b-bit blocks and iterating F appropriately. The widely used Strengthened Merkle-Damgård (SMD) construction [Mer90a, Dam90] is known to yield a collision-resistant iterated hash function if the underlying compression function is collision resistant; in other words, SMD *preserves* collision resistance of the compression function.

Unfortunately, designing collision resistant compression functions seems hard: witness the recent collision attacks on several popular hash functions by Wang et al. [WY05, WYY05]. One way out is to aim for a weaker security notion for the compression function, but not so weak as to make the resulting hash function useless in practice. A natural question to ask is whether these weaker properties are also preserved by SMD. For example, does it preserve second-preimage resistance? One hopes so, since then those applications requiring only second-preimage resistance might remain secure until efficient second-preimage attacks on the compression function are found. Intuitively, the answer seems to be positive: after all, SMD preserves collision resistance, and collision resistance can be shown to imply second-preimage resistance. This intuition is tempting, but wrong. It is true that when using a collision-resistant compression function, SMD yields a second-preimage resistant hash function (although the effective security level will be bounded by that of the collision resistance). Butthis says *nothing* about what happens if you *start* with a compression function that is only second-preimage resistant. Lai and Massey [LM92] claimed that finding second preimages for an iterated hash is equally as hard as finding second preimages for the compression function, but this was found to be incorrect by Dean [Dea99] and Kelsey and Schneier [KS05], who show that (for the case of SMD) efficient collision-finding attacks immediately give rise to second-preimage attacks that beat the anticipated security bound.

Contributions. We took as a starting point a paper by Rogaway and Shrimpton [RS04] that provides a unifying framework of seven security notions for hash functions and the relations among them. Our work explores in detail which of the seven properties of [RS04] are preserved by several published hash constructions. Of the eleven schemes we consider (see Table 1), we found that in fact none preserved all seven. This raises the question whether it is possible at all to preserve all seven properties. We answer this question in the affirmative, in the random oracle model [BR93], by presenting a construction that builds on previous work by Bellare, Rogaway, Shoup and Mironov [BR97, Sho00, Mir01]. Our construction iterates a *real-world* compression function but, in the iteration, makes a logarithmic (in the message length) number of calls to an auxiliary small-input random oracle; we will say more in a moment to justify this choice. The existence of seven-property-preserving iterations in the standard model is left as an open problem.

Relevance of the seven properties. Apart from collision-resistance, Rogaway and Shrimpton consider three variants of second-preimage resistance (Sec) and preimage resistance (Pre). The standard variants of Sec and Pre are restricted to randomly chosen preimages, and have important applications like the Cramer-Shoup cryptosystem [CS03] for Sec and Unix-like password storage [LR89, WG00] for Pre. The stronger *everywhere* variants (eSec, ePre) consider adversarially chosen preimages. The notion of ePre is implied by the "strong" hash requirement used in the proof of a signature scheme with partial message recovery [BJ01]. The notion of eSec is equivalent to the universal one-way hash functions of Naor and Yung [NY89] and to the target collision resistance of Bellare and Rogaway [BR97]. Bellare and Rogaway show that eSec is sufficient to extend the message space of signature schemes that are defined for small messages only.

Following the standard convention established by Damgård [Dam90], and Bellare and Rogaway [BR97], these notions were formalized for hash function families, indexed by a (publicly known) key K. Current practical hash functions however do not have explicit keys. In fact, it is not even clear what the family is that they belong to, so it is rather contrived to regard SHA-256 as a randomly drawn member of such a family. Instead, the always-notions aSec and aPre capture the intuition that a hash function ought to be (second-)preimage resistant for *all* members of the family, so that it doesn't matter which one is actually used. Alternatively, one could see the aSec and aPre notions as the the natural extensions to (second-)preimage resistance of Rogaway's human-ignorance approach to collision-resistant hashing with unkeyed compression functions [Rog06]. (See [ANPS07b] for a subsequent work on property preservation for iterations of unkeyed compression functions.) In this sense, the aSec and aPre notions strengthen the standard notions of second-preimage resistance and preimage resistance, respectively, in the way needed to say that a fixed function such as SHA-256 is Sec and Pre secure. They therefore inherit the practical applications of Sec and Pre security, and are thus the right notions to consider when instantiating Cramer-Shoup encryption or Unix-like password storage with a fixed function like SHA-256. The formal definitions of all seven notions are recalled in Section 2.

Existing constructions. Let us now take a closer look at a number of existing constructions to see which of the seven notions of [RS04] they preserve. Our findings are summarized in Table 1, which we see as the main research contribution of our paper. Except for the few entries in the table with question marks, we come up with either proofs or counterexamples in support of our claims. We found for example that the ubiquitous SMD construction preserves Coll and ePre security, but surprisingly fails to preserve any of the other notions. In fact, all of the examined schemes preserve ePre; intuitively, if all of the range of the (randomly keyed) compression function is hard to invert, then iterating produces a function whose range is similarly hard to invert. Apart from ePre, most schemes preserve only collision resistance. These schemes include SMD and the more recent EMD, HAIFA, and Randomized hash. Of the eleven schemes in the table, none preserves all seven notions. In fact, the best-performing constructions in terms of property preservation are the XOR Linear hash and Shoup's hash, which still preserve only three of the seven notions (Coll, eSec, and ePre). The XOR Tree hash is the only iteration to preserve Pre, and none of the schemes preserve Sec, aSec or aPre. Remember that the latter two are particularly relevant for the security of practical hash functions because they do not rely on the compression functions being chosen at random from a family.

Preserving all properties: the ROX Construction. This rather poor state of affairs may leave one wondering whether preserving all seven notions is possible at all. We answer this question in the affirmative, but, quite controversially, were only able to do so in the random oracle model. We explicitly do *not* model the compression function itself as a random oracle however. While we view the main interest of our construction to be a feasibility result for seven-property-preserving hashing, we do have reasons to believe that our construction makes very "reasonable" use of the random oracle. Allow us to explain.

Our Random-Oracle-XOR (ROX) construction draws largely on the XOR-linear hash [BR97] and Shoup's hash [Sho00]. Both are extensions of the original MD construction that XOR a mask into the chaining value prior to application of the compression function. Shoup's construction requires only a logarithmic number of masks (in the length of the message), which is an improvement over the linear number required in Bellare and Rogaway's XOR-linear hash. Moreover, Mironov [Mir01] proves that Shoup's mask scheduling is optimal. We, too, require a logarithmic number of masks, and we apply the masks according to Shoup's algorithm. Our masks, however, are generated by applying a random oracle to a sequence of strings $(K, \mathfrak{m}, \langle i \rangle)$ that consist of the compression function key K of length k, the first k bits of the message, and an encoding of a counter i. For a typical security level of $k = 80$, this means we only need a random oracle with 170-bit inputs. To hash an ℓ-block message, we query the random oracle on a number of domain points that is logarithmic in; for example, if $|M| = 2^{64}$ bits and blocks are 512 bits long, we need at most 56 random oracle calls. We also use a random oracle to create a padding string; this adds a small, constant number (e.g., two) of calls to a 140-bit random

Table 1: **Overview of constructions and the properties they preserve.** Each row in the table represents a hash function construction, each column a security notion of [RS04]. The symbol "Y" means that the notion is provably preserved by the construction; "N" means that it is not preserved, in the sense that we come up with a counterexample; "?" means that neither proof nor counterexample are known. Underlined entries were known, all other results are new.

Scheme	Coll	Sec	aSec	eSec	Pre	aPre	ePre
Strengthened MD [Mer90b, Dam90]	$\underline{\text{Y}}$	N	N	$\underline{\text{N}}$	N	N	Y
Linear [BR97]	N	N	N	$\underline{\text{N}}$	N	N	Y
XOR-Linear [BR97]	Y	N	N	$\underline{\text{Y}}$	N	N	Y
Shoup's [Sho00]	Y	N	N	$\underline{\text{Y}}$	N	N	Y
Prefix-free MD [CDMP05]	$\underline{\text{N}}$	N	N	N	N	N	Y
Randomized [HK06]	Y	N	N	N	N	N	Y
HAIFA [BD06]	$\underline{\text{Y}}$	N	N	N	N	N	Y
Enveloped MD [BR06]	$\underline{\text{Y}}$	N	N	N	N	N	Y
Strengthened Merkle Tree [Mer80]	$\underline{\text{Y}}$	N	N	N	N	N	Y
Tree Hash [BR97]	N	N	N	N	N	N	Y
XOR Tree [BR97]	?	?	N	?	Y	N	Y
ROX	Y	Y	Y	Y	Y	Y	Y

oracle.This limited use of the random oracle has the important practical ramification that the function instantiating it need not be as efficient as the compression function, and can therefore be made with large security margins. We'll come back to candidate instantiations in Section 4.

The idea of generating the masks through a random oracle is not new; in fact, it was explicitly suggested at two separate occasions by Mironov [Mir01, Mir06]. The idea was discarded in [Mir01] for trivializing the problem, but was revisited in [Mir06] as a viable way to obtain shorter keys for eSec-secure hashing. Indeed, if one assumes the existence of random oracles with very large domains, then one can simply use the random oracle to do the hashing. The ROX construction, on the other hand, still uses a real compression function in the chaining, and uses a small-domain random oracle to preserve all seven notions of [RS04] using a very short key, including the important aSec and aPre notions.[1] Moreover, we do so without changing the syntax of the compression function [BD06] or doubling its output size [Luc05], both of which can come at a considerable performance penalty.

What about other properties? The seven security notions formalized by [RS04] are certainly not the only ones that are of interest. Kelsey and Kohno [KK06] suggest chosen-target forced-prefix security, which can be seen as a special form of multi-collision resistance, as the right goal to stop Nostradamus attacks. Bellare and Ristenpart [BR06], following previous work by Coron et al. [CDMP05] and Bellare et al. [BCK96], formalize pseudorandom oracle preservation (PRO-Pr) and pseudorandom function preservation (PRF-Pr) as goals. Their EMD construction is shown to be PRO-Pr, PRF-Pr and to preserve collision resistance. More recently, and independently of this work, Bellare and Ristenpart [BR07] study the Coll, eSec, PRO, PRF, and MAC (unforgeability) preservation of various iterations, including the SMD, Prefix-free MD, Shoup, and EMD iterations that we study. Their work does not cover the five other notions of [RS04], while our work does not cover the PRO, PRF, and

[1] While ROX itself is an explicitly keyed construction, its preservation of aSec/aPre implies that the instantiating compression function need not be. Indeed, when instantiated with a fixed aSec/aPre-secure compression function like SHA-256, then the resulting iterated hash is aSec/aPre-secure and therefore also Sec/Pre-secure. ROX thereby provides a secure way of iterating unkeyed (second-)preimage resistant compression functions.

MAC properties. We leave the study of the preservation of these properties by our $\mathcal{ROX}$ construction to future work.

2 Security Definitions

In this section, we explain the security notions for hash functions of [RS04]. Let us begin by establishing some notation. Let $\mathbb{N} = \{0, 1, \ldots\}$ be the set of natural numbers and $\{0,1\}^*$ be the set of all bit strings. If $k \in \mathbb{N}$, then $\{0,1\}^k$ denotes the set of all k-bit strings and $\{0,1\}^{k\times *}$ denotes the set of all bit strings of length an integer multiple of k. The empty string is denoted ε. If b is a bit then $\overline{b}$ denotes its complement. If x is a string and $i \in \mathbb{N}$, then $x^{(i)}$ is the i-th bit of x and x^i is the concatenation of i copies of x. If x, y are strings, then $x \| y$ is the concatenation of x and y. If $k, l \in \mathbb{N}$ then $\langle k \rangle_l$ is the encoding of k as an l-bit string. We occasionally write $\langle k \rangle$ when the length is clear from the context. If S is a set, then $x \stackrel{\$}{\leftarrow} S$ denotes the uniformly random selection of an element from S. We let $y \leftarrow \mathsf{A}(x)$ and $y \stackrel{\$}{\leftarrow} \mathsf{A}(x)$ be the assignment to y of the output of a deterministic and randomized algorithm A, respectively, when run on input x.

An *adversary* is an algorithm, possibly with access to 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).

Security Notions for Keyed Hash Functions. Formally, 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 only assume that there exists at least one $\lambda \in \mathbb{N}$ such that $\{0,1\}^\lambda \subseteq \mathcal{M}$. We treat (fixed input length) compression functions and (variable input length) hash functions just the same, the former being simply a special case of the latter.

For an adversary A attacking a hash function H we define the following advantage measures:

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

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

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

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

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

$$\mathbf{Adv}^{\mathrm{ePre}}_{\mathrm{H}}(\mathsf{A}) = \Pr\left[(Y, St) \stackrel{\$}{\leftarrow} \mathsf{A}\,;\, K \stackrel{\$}{\leftarrow} \mathcal{K}\,;\, M' \stackrel{\$}{\leftarrow} \mathsf{A}(K, St) \;:\; \mathrm{H}(K, M') = Y \right]$$

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

These are the seven security notions from Rogaway-Shrimpton [RS04]: the standard three of *collision-resistance* (Coll), *preimage-resistance* (Pre) and *second-preimage-resistance* (Sec); and the *always*- and *everywhere*- variants of (second)-preimage-resistance (aPre, aSec, ePre and eSec).

For atk $\in$ {Coll, eSec, ePre}, we define $\mathbf{Adv}_{\mathrm{H}}^{\mathrm{atk}}(t)$ to be the maximum advantage of any adversary with time complexity at most t. We say that the hash function family H is (t, ϵ) atk-secure if $\mathbf{Adv}_{\mathrm{H}}^{\mathrm{atk}}(t) < \epsilon$. For technical reasons (having to do with uniform sampling of a set) some of our definitions are parameterized by λ. For atk $\in$ {Sec, aSec, Pre, aPre}, we say that H is (t, ϵ) atk[λ]-secure if $\mathbf{Adv}_{\mathrm{H}}^{\mathrm{atk}[\lambda]}(t) < \epsilon$, and we say that H is (t, ϵ) atk-secure if $\mathbf{Adv}_{\mathrm{H}}^{\mathrm{atk}[\lambda]}(t) < \epsilon$ for all $\lambda \in \mathbb{N}$ such that $\{0,1\}^\lambda \subseteq \mathcal{M}$. Note that fixed-length compression functions have $\mathcal{M} = \{0,1\}^\lambda$, so that atk[$\lambda$]-security and atk-security are actually equivalent. When giving results in the random oracle model, we also account for the total number of queries q_{RO} that the adversary makes to its random oracles. In this case, we will write $(t, q_{\mathrm{RO}}, \epsilon)$-secure with the obvious meaning.

Note that the security notions above do not insist that the colliding message M' be of length λ. It is our conscious choice to focus on arbitrary-length security here, meaning that adversaries may find collisions between messages of varying lengths. In practice, the whole purpose of hash iterations is to extend the domain of a compression function to arbitrary lengths, so it makes perfect sense to require that the hash function withstands attacks using messages of different lengths.

3 Properties Preserved by Existing Constructions

In this section we take a closer look at eleven hash iterations that previously appeared in the literature, and check which of the seven security properties from [RS04] they preserve. The algorithms are described in Figure 1, the results of our analysis are summarized in Table 1.

As mentioned in the previous section, we focus on arbitrary-length security in this paper. Allowing for arbitrary-length message attacks invariably seems to require some sort of message padding (unstrengthened MD does not preserve collision resistance), but care must be taken when deciding on the padding method: one method does not fit all. This was already observed by Bellare and Rogaway [BR97], who proposed an alternative form of strengthening where a final block containing the message length is appended and processed with a different key than the rest of the iteration. This works fine in theory, but since current compression functions are not keyed, it is not clear how this construction should be instantiated in practice. In absence of a practical generic solution, we chose to add standard one-zeroes padding and length strengthening to all chaining iterations that were originally proposed without strengthening. For tree iterations we use one-zeroes padding for the message input at the leaves, and at the root make one extra call to the compression function on input the accumulated hash value concatenated with the message length. (Standard length strengthening at the leaves fails to preserve even collision resistance here.) These strengthening methods sometimes help but never harm for property preservation.

3.1 Chaining Iterations

Strengthened Merkle-Damgård. The *Strengthened Merkle-Damgård* ($\mathcal{SMD}$) construction is known to preserve collision resistance [Dam90] and to not preserve eSec security [BR97]. In the following two theorems we prove that it also preserves ePre security, but does not preserve Sec, aSec, Pre, and aPre security. τ_{F} is the time required for an evaluation of F and $\ell = \lceil (\lambda + 2n)/b \rceil$ where $\lambda = |M|$.

Theorem 3.1 If F is (t', ϵ') ePre-secure, then $\mathcal{SMD}_{\mathrm{F}}$ is (t, ϵ) ePre-secure for $\epsilon = \epsilon'$ and $t = t' - \ell \cdot \tau_{\mathrm{F}}$.

Proof: Given an ePre-adversary A against $\mathcal{SMD}_{\mathrm{F}}$, consider the following ePre-adversary B against F. B runs A to obtain the target value Y and outputs the same string Y. When it gets a random key K it runs A on the same key to obtain a preimage message M'. Let $m'_1 \| \ldots \| m'_\ell \leftarrow$ ls-pad(M') and let $h'_{\ell-1}$

```
Algorithm $\mathcal{SMD}_{\mathrm{F}}(K, M)$:
  $m_1\|\ldots\|m_\ell \leftarrow$ ls-pad$(M)$ ; $h_0 \leftarrow IV$
  For $i = 1 \ldots \ell$ do $h_i \leftarrow \mathrm{F}(K, m_i\|h_{i-1})$
  Return $h_\ell$
```

```
Algorithm $\mathcal{LH}_{\mathrm{F}}(K_1\|\ldots\|K_\ell\ ,\ M)$:
  $m_1\|\ldots\|m_\ell \leftarrow$ ls-pad$(M)$ ; $h_0 \leftarrow IV$
  For $i = 1, \ldots, \ell$ do $h_i \leftarrow \mathrm{F}(K_i, m_i\|h_{i-1})$
  Return $h_\ell$
```

```
Algorithm $\mathcal{XLH}_{\mathrm{F}}(K\|K_1\|\ldots\|K_\ell\ ,\ M)$:
  $m_1\|\ldots\|m_\ell \leftarrow$ ls-pad$(M)$ ; $h_0 \leftarrow IV$
  For $i = 1, \ldots, \ell$ do
    $h_i \leftarrow \mathrm{F}(K, m_i\|(h_{i-1} \oplus K_{i-1}))$
  Return $h_\ell$
```

```
Algorithm $\mathcal{SH}_{\mathrm{F}}(K\|K_1\|\ldots\|K_{\lceil \log \ell \rceil}, M)$:
  $m_1\|\ldots\|m_\ell \leftarrow$ ls-pad$(M)$ ; $h_0 \leftarrow IV$
  For $i = 1, \ldots, \ell$ do
    $h_i \leftarrow \mathrm{F}(K, m_i\|(h_{i-1} \oplus K_{\nu(i)}))$
  Return $h_\ell$
```

```
Algorithm $\mathcal{PfMD}_{\mathrm{F}}(K, M)$:
  $m_1\|\ldots\|m_\ell \leftarrow$ pf-pad$(M)$ ; $h_0 \leftarrow IV$
  For $i = 1, \ldots, \ell$ do $h_i \leftarrow \mathrm{F}(K, m_i\|h_{i-1})$
  Return $h_\ell$
```

```
Algorithm $\mathcal{EMD}_{\mathrm{F}}(K, M)$:
  $m_1\|\ldots\|m_\ell \leftarrow$ emd-pad$(M)$ ; $h_0 \leftarrow IV_1$
  For $i = 1 \ldots \ell - 1$ do $h_i \leftarrow \mathrm{F}(K, m_i\|h_{i-1})$
  Return $h_\ell \leftarrow \mathrm{F}(K, h_{\ell-1}\|m_\ell\|IV_2)$
```

```
Algorithm $\mathcal{HAIFA}_{\mathrm{F}}(K, M)$:
  $m_1\|\ldots\|m_\ell \leftarrow$ oz-pad$(M, i \cdot b)$ ; $h_0 \leftarrow IV$
  $ctr \leftarrow 0$ ; $S \xleftarrow{\$} \{0,1\}^s$   // $S$ is a salt
  For $i = 1 \ldots \ell - 1$ do
    $ctr \leftarrow ctr + b$ ; $h_i \leftarrow \mathrm{F}(K, m_i\|\langle ctr \rangle_l\|S\|h_{i-1})$
  $h_\ell \leftarrow \mathrm{F}(K, m_\ell\|\langle |M| \rangle\|S\|h_{\ell-1})$
  Return $S, h_\ell$
```

```
Algorithm $\mathcal{RH}_{\mathrm{F}}(K\|R, M)$:
  $m_1\|\ldots\|m_\ell \leftarrow$ sf-pad$(M)$
  $h_0 \leftarrow \mathrm{F}(K, R\|IV)$
  For $i = 1 \ldots \ell$ do
    $h_i \leftarrow \mathrm{F}(K, (m_i \oplus R)\|h_{i-1})$
  Return $h_\ell$
```

```
Algorithm $\mathcal{SMT}_{\mathrm{F}}(K, M)$:
  $m_1\|\ldots\|m_\ell \leftarrow$ tpad$(M)$
  For $j = 1, \ldots, a^{d-1}$ do
    $h_{1,j} \leftarrow \mathrm{F}(K, m_{(j-1)a+1}\|\ldots\|m_{ja})$
  For $i = 2, \ldots, d$ and $j = 1, \ldots, a^{d-i}$ do
    $h_{i,j} \leftarrow \mathrm{F}(K, h_{i-1,(j-1)a+1}\|\ldots\|h_{i-1,ja})$
  $h_{d+1,1} \leftarrow \mathrm{F}(K, h_{d,1}\|\langle |M| \rangle_{n(a-1)})$
  Return $h_{d+1,1}$
```

```
Algorithm $\mathcal{TH}_{\mathrm{F}}(K_1\|\ldots\|K_{d+1}\ ,\ M)$:
  $m_1\|\ldots\|m_\ell \leftarrow$ tpad$(M)$
  For $j = 1, \ldots, a^{d-1}$ do
    $h_{1,j} \leftarrow \mathrm{F}(K_1, m_{(j-1)a+1}\|\ldots\|m_{ja})$
  For $i = 2, \ldots, d$ and $j = 1, \ldots, a^{d-i}$ do
    $h_{i,j} \leftarrow \mathrm{F}(K_i, h_{i-1,(j-1)a+1}\|\ldots\|h_{i-1,ja})$
  $h_{d+1,1} \leftarrow \mathrm{F}(K_{d+1}, h_{d,1}\|\langle |M| \rangle_{n(a-1)})$
  Return $h_{d+1,1}$
```

```
Algorithm $\mathcal{XTH}_{\mathrm{F}}(K\|K_1\|\ldots\|K_{d+1}\ ,\ M)$:
  $m_1\|\ldots\|m_\ell \leftarrow$ tpad$(M)$
  For $j = 1, \ldots, a^{d-1}$ do
    $h_{1,j} \leftarrow \mathrm{F}(K, (m_{(j-1)a+1}\|\ldots\|m_{ja}) \oplus K_1)$
  For $i = 2, \ldots, d$ and $j = 1, \ldots, a^{d-i}$ do
    $h_{i,j} \leftarrow \mathrm{F}(K, (h_{i-1,(j-1)a+1}\|\ldots\|h_{i-1,ja}) \oplus K_i)$
  $h_{d+1,1} \leftarrow \mathrm{F}(K, (h_{d,1}\|\langle |M| \rangle_{n(a-1)}) \oplus K_{d+1})$
  Return $h_{d+1,1}$
```

```
Padding algorithms:
  oz-pad$(M, x) = M\|100^{x-|M|-2}$
  ls-pad$(M) =$ oz-pad$(M, x)\|\langle |M| \rangle_b$
    where $x = \lceil (|M| + 2)/b \rceil \cdot b$
  emd-pad$(M) =$ oz-pad$(M, x)\|\langle |M| \rangle_{64}$
    where $x = \lceil (|M| + 66)/b \rceil \cdot b - 64$
  tpad$(M) =$ oz-pad$(M, x)$
    where $x = a^{\lceil \log_a |M| \rceil} \cdot n$
```

Figure 1: **Some existing iterative hash constructions.** Chaining iterations $\mathcal{SMD}$, $\mathcal{LH}$, $\mathcal{XLH}$, $\mathcal{SH}$, $\mathcal{PfMD}$, $\mathcal{RH}$, and $\mathcal{EMD}$ use a compression function $\mathrm{F} : \{0,1\}^k \times \{0,1\}^{b+n} \to \{0,1\}^n$; $\mathcal{HAIFA}$ uses a compression function $\mathrm{F} : \{0,1\}^k \times \{0,1\}^{b+l+s+n} \to \{0,1\}^n$. Tree iterations $\mathcal{SMT}$, $\mathcal{TH}$, and $\mathcal{XTH}$ use a compression function $\mathrm{F} : \{0,1\}^k \times \{0,1\}^{an} \to \{0,1\}^n$. Strings $IV, IV_1, IV_2 \in \{0,1\}^n$ are fixed initialization vectors. Padding algorithms are given on the bottom right; pf-pad(M) and sf-pad(M) are any prefix-free padding and suffix-free padding algorithms, respectively. The function $\nu(i)$ is the largest integer j such that $2^j | i$.

be the one-but-last chaining value computed in an execution of $\mathcal{SMD}_{\mathrm{F}}(K, M')$. Algorithm B outputs $m'_\ell\|h'_{\ell-1}$ as its own preimage. ▌

While at first sight the above proof may seem to go through for Pre and aPre security as well, this is not the case. The target point Y in a Pre attack on F is distributed as $\mathrm{F}(K, m\|h)$ for a random $m\|h \xleftarrow{\$} \{0,1\}^{b+n}$. But the target point for the iterated structure $\mathcal{SMD}_{\mathrm{F}}$ is generated as $\mathcal{SMD}_{\mathrm{F}}(K, M)$ for a random $M \xleftarrow{\$} \{0,1\}^\lambda$. These two distributions can actually be very different, as is illustrated by the following counterexample.

Theorem 3.2 For atk $\in \{\text{Sec}, \text{aSec}, \text{Pre}, \text{aPre}\}$, if there exists a (t, ϵ) atk-secure compression function $\mathrm{G} : \mathcal{K} \times \{0,1\}^{b+n} \to \{0,1\}^{n-1}$, then there exists a $(t, \epsilon - 1/2^n)$ atk-secure compression function $\mathrm{CE}_1 : \mathcal{K} \times \{0,1\}^{b+n} \to \{0,1\}^n$ and an adversary A running in one time step with atk[λ]-advantage one in breaking $\mathcal{SMD}_{\mathrm{CE}_1}$.

Proof: For any compression function G, consider CE_1 given by

$$\begin{array}{rcll} \mathrm{CE}_1(K, m\|h) & = & IV & \text{if } h = IV \\ & = & \mathrm{G}(K, m\|h) \;\|\; \overline{IV}^{(n)} & \text{otherwise .} \end{array}$$

If G is (t, ϵ) atk secure, then CE_1 is $(t, \epsilon - 1/2^n)$ atk secure; we refer to Appendix A.1 for the proof. From the construction of CE_1, it is clear that $\mathcal{SMD}_{\mathrm{CE}_1}(K, M) = IV$ for all $M \in \{0,1\}^*$. Hence, the adversary can output any message M' as its (second) preimage. ▮

Linear hash. The *Linear Hash* ($\mathcal{LH}$) [BR97] uses ℓ different keys for ℓ-block messages, because it calls the compression function on a different key at every iteration. The Linear Hash is known to preserve eSec-security for same-length messages, but Bellare and Rogaway claim [BR97] that length-strengthening does not suffice to preserve eSec for different-length messages. The following theorem confirms their claim, and also shows that $\mathcal{LH}$ does not preserve Coll. The counterexample CE_1 of Theorem 3.2 can be used to disprove the preservation of Sec, aSec, Pre and aPre-security. A proof similar to that of Theorem 3.1 can be used to show that $\mathcal{LH}$ does preserve ePre-security.

Theorem 3.3 For any atk $\in \{\text{Coll}, \text{eSec}\}$, if there exists a (t, ϵ) atk-secure compression function $\mathrm{G} : \{0,1\}^k \times \{0,1\}^{b+n} \to \{0,1\}^{n-2}$, then there exists a (t, ϵ) atk-secure compression function $\mathrm{CE}_2 : \{0,1\}^k \times \{0,1\}^{b+n} \to \{0,1\}^n$ and an adversary A running in one step time with atk-advantage 1/4 in breaking $\mathcal{LH}_{\mathrm{CE}_2}$.

Proof: For any compression function G, consider CE_2 given by

$$\begin{array}{rcll} \mathrm{CE}_2(K, m\|h) & = & IV & \text{if } m\|h = 010^{b-2}\|IV \\ & = & 0^{n-1} \;\|\; \overline{IV}^{(n)} & \text{if } (K^{(1)} = 0 \text{ and } m\|h = \langle 1 \rangle_b \| IV) \\ & & & \text{or } (K^{(1)} = 1 \text{ and } m\|h = \langle b+1 \rangle_b \| IV) \\ & = & \mathrm{G}(K, m\|h) \;\|\; 1 \;\|\; \overline{IV}^{(n)} & \text{otherwise ,} \end{array}$$

In Appendix A.2 we prove that if G is (t, ϵ) atk-secure for atk $\in \{\text{Coll}, \text{eSec}\}$, then CE_2 is (t, ϵ) atk-secure. When iterating CE_2 through $\mathcal{LH}_{\mathrm{CE}_2}$ with independent keys $K_1\|K_2\|K_3$, one can easily see that if $K_2^{(1)} = 0$ and $K_3^{(1)} = 1$, then messsages $M = 0$ and $M' = 010^{b-1}$ both hash to $0^{n-1}\|\overline{IV}^{(n)}$. Since in the Coll and eSec games this case happens with probability 1/4, we have attacks satisfying the claim in the theorem. ▮

XOR-Linear Hash. The *XOR-Linear Hash* ($\mathcal{XLH}$) [BR97] uses keys that consist of a compression function key K and ℓ masking keys $K_1, \ldots, K_\ell \in \{0,1\}^n$. It is known to preserve eSec security [BR97]. It can also be seen to preserve Coll and ePre by similar arguments as used for $\mathcal{SMD}$ and $\mathcal{LH}$. Counterexample CE_1 can be used to show that aSec and aPre are not preserved: the adversary gets to choose the key in these notions, so it can choose $K_1 = \ldots = K_\ell = 0^n$ so that $\mathcal{XLH}$ boils down to $\mathcal{SMD}$. In the following we show that the $\mathcal{XLH}$ construction does not preserve Sec or Pre security either.

Theorem 3.4 For any atk $\in \{\text{Sec}, \text{Pre}\}$, if there exists a (t, ϵ) atk-secure compression function $\mathrm{G} : \mathcal{K} \times \{0,1\}^{b+n} \to \{0,1\}^{n-1}$, then there exists a $(t, \epsilon + 1/2^b)$ atk-secure compression function $\mathrm{CE}_3 : \mathcal{K} \times \{0,1\}^{b+n} \to \{0,1\}^n$ and an adversary A running in one step time with atk[λ]-advantage one in breaking $\mathcal{XLH}_{\mathrm{CE}_3}$.

Proof: For any $\lambda \leq 2^b$ and compression function G, consider CE_3 given by

$$\begin{array}{rcll} \mathrm{CE}_3(K, m\|h) & = & 0^n & \text{if } m = \langle\lambda\rangle_b \\ & = & \mathrm{G}(K, m\|h)\|1 & \text{otherwise .} \end{array}$$

In Appendix A.3 we prove that if G is (t, ϵ) Sec or Pre-secure, then CE_3 is $(t, \epsilon + 1/2^b)$ Sec or Pre-secure. It is easy to see that, when iterated through $\mathcal{XLH}_{\mathrm{CE}_3}$, the hash of any λ-bit message is 0^n. A Pre[λ] adversary can therefore simply output any $M' \in \{0,1\}^\lambda$, a Sec[λ] adversary can output any $M' \neq M \in \{0,1\}^\lambda$. ▮

Shoup's Hash. The iteration due to Shoup ($\mathcal{SH}$) [Sho00] is similar to the XOR-Linear hash but uses a different key scheduling that reduces the key length to logarithmic in the message length, rather than linear. Shoup's hash is known to preserve eSec-security [Sho00], and it can be shown to preserve Coll and ePre-security as well. The proofs are very similar to the case of SMD, and hence omitted. Counterexample CE_1 disproves preservation of aSec and aPre-security, and counterexample CE_3 disproves preservation of Sec and Pre.

Prefix-free Merkle-Damgård. It was shown in [BR06] that the *prefix-free Merkle-Damgård* construction ($\mathcal{PfMD}$) [CDMP05] does not preserve Coll security. The counterexample of [BR97] can also be used to show that it does not preserve eSec, and counterexample CE_1 can be used to disprove the preservation of Sec, aSec, Pre and aPre. Finally, using a proof similar to that for $\mathcal{SMD}$, one can show that ePre-security is preserved.

Another variant of $\mathcal{PfMD}$ by [CDMP05] prepends the message length encoding to the message in advance. The security results of this scheme easily follow from the ones for the $\mathcal{SMD}$ construction.

Randomized hash. The *Randomized Hash* ($\mathcal{RH}$) [HK06] XORs each message block with a random value $R \in \{0,1\}^b$. The construction was originally proved to be eSec secure by making stronger assumptions on the underlying compression function. Its pure security preservation characteristics (i.e., assuming only the eSec security of the compression function) were never studied. In our security analysis of $\mathcal{RH}$ treating the value R as either randomness per message or fixed long term key yields identical results with respect to seven property preservation.

By arguments similar to the case of $\mathcal{SMD}$, one can show that $\mathcal{RH}$ preserves Coll and ePre security, but none of the other notions are preserved. Counterexample CE_1 can be used to contradict preservation of Sec, aSec, Pre, and ePre, and the counterexample of [BR97] can be used to contradict preservation of eSec.

HAIFA. While the newly proposed *HAsh Iterative FrAmework* ($\mathcal{HAIFA}$) [BD06] does preclude a number of specific attacks [Dea99, KS05, KK06] to which $\mathcal{SMD}$ admits, they perform exactly the same in terms of preservation of our security notions. Similar proofs as for $\mathcal{SMD}$ can be used to show that $\mathcal{HAIFA}$ preserves Coll and ePre-security, counterexample CE_1 can be used to contradict the preservation of Sec, aSec, Pre, and aPre, and the counterexample of [BR97] applies to contradict preservation of eSec.

Enveloped Merkle-Damgård. The *enveloped Merkle-Damgård* ($\mathcal{EMD}$) construction [BR06] is known to preserve Coll, pseudorandom-oracle, and pseudo-random function behavior. For the seven security notions that we consider, however, it does not perform better than $\mathcal{SMD}$. Counterexample CE_1 of Theorem 3.2 can be used (setting $IV = IV_2$) to show that neither of Sec, aSec, Pre, or aPre are preserved. An adaptation of the counterexample of [BR97] shows that eSec is not preserved either. Preservation of ePre on the other hand can be proved in a similar way as done in Theorem 3.1.

3.2 Tree Iterations

Strengthened Merkle Tree. We consider here the strengthened Merkle tree [Mer80], the Tree Hash [BR97], and the XOR Tree Hash [BR97]. For conciseness we do not cover other tree iterations that have appeared in the literature (e.g. [LCL+03, Sar05]). The Merkle tree [Mer80] in its most basic form (i.e., without length strengthening) suffers from a similar anomaly as basic Merkle-Damgård in that it does not preserve Coll for arbitrary-length messages. We therefore consider the strengthened variant $\mathcal{SMT}$ here, depicted in Figure 1. We believe $\mathcal{SMT}$ is commonly known to preserve Coll, but we reprove this in Appendix A.4 for completeness. The notion of ePre is easily seen to be preserved as well. It can be seen not to preserve eSec by a counterexample similar to that of [BR97] given in Appendix A.4. $\mathcal{SMT}$ also fails to preserve Sec, aSec, Pre, and aPre however, as shown in the following theorem.

Theorem 3.5 For any atk $\in \{\mathrm{Sec}, \mathrm{aSec}, \mathrm{Pre}, \mathrm{aPre}\}$, if there exists a (t', ϵ') atk-secure compression function $\mathrm{G} : \mathcal{K} \times \{0,1\}^{an} \to \{0,1\}^{n-2}$, then there exists a (t, ϵ) atk-secure compression function $\mathrm{CE}_4 : \mathcal{K} \times \{0,1\}^{an} \to \{0,1\}^n$ for $\epsilon = \epsilon' + 1/2^{n-1}$, $t = t'$, and an adversary A running in one step time with atk$[\lambda]$ advantage 1 in breaking $\mathcal{SMT}_{\mathrm{CE}_4}$.

Proof: For any compression function G, consider CE_4 given by

$$\begin{array}{rcll} \mathrm{CE}_4(K, m_1\|\ldots\|m_a) & = & 0^n & \text{if } m_a = 0^n \\ & = & 1^n & \text{if } m_{a-1} = 0^n \text{ and } m_a \neq 0^n \\ & = & \mathrm{G}(K, m_1\|\ldots\|m_a) \,\|\, 10 & \text{otherwise .} \end{array}$$

We prove in Appendix A.4 that the bounds mentioned above hold for the atk security of CE_4. It is easy to see that, due to the one-zeroes padding to a^d bits, any message of length $a^{d-1} - 1 \leq \lambda \leq a^d - 1$ hashes to 1^n, leading to trivial constant-time attacks for any such length λ. ▮

Tree Hash. The unstrengthened Tree Hash ($\mathcal{TH}$) was proposed in [BR97] for same-length messages; we consider the strengthened variant here. It is a variant of $\mathcal{SMT}$ where at each level i of the tree the compression functions use an independent key K_i. It can be seen to preserve ePre for the same reasons as the $\mathcal{SMT}$ construction. Our counterexample CE_4 can be used to exhibit the non-preservation of Sec, aSec, Pre and aPre security. The case of Coll and eSec are a bit more subtle, but the counterexample below shows that $\mathcal{TH}$ does not preserve these either.

Theorem 3.6 For any atk $\in \{\mathrm{Coll}, \mathrm{eSec}\}$, if there exists a (t', ϵ') atk-secure compression function $\mathrm{G} : \{0,1\}^k \times \{0,1\}^{an} \to \{0,1\}^{n-1}$, then there exists a (t, ϵ) atk-secure compression function $\mathrm{CE}_5 : \{0,1\}^k \times \{0,1\}^{an} \to \{0,1\}^n$ for $\epsilon = \epsilon'$, $t = t'$, such that there exists an eSec-adversary breaking the eSec security of $\mathcal{TH}_{\mathrm{CE}_5}$ in constant time with advantage 1/4.

Proof: For any compression function G, consider CE_5 given by

$$\begin{array}{rcll} \mathrm{CE}_5(K, M) & = & 10^{n-1} & \text{if } M = (10^{n-1})^a \\ & = & 1^n & \text{if } \big(K^{(1)} = 0 \text{ and } M = (10^{n-1})^{a-1} \| \langle (a-1)n \rangle_n\big) \\ & & & \text{or } \big(K^{(1)} = 1 \text{ and } M = (10^{n-1})^{a-1} \| \langle (a^2-1)n \rangle_n\big) \\ & = & 0 \,\|\, \mathrm{G}(K, M) & \text{otherwise .} \end{array} \qquad (1)$$

We prove in Appendix A.5 that CE_5 is (t, ϵ) atk-secure whenever G is (t, ϵ) atk-secure, for atk $\in \{\mathrm{Coll}, \mathrm{eSec}\}$.

Note that $\mathrm{tpad}\big(M = (10^{n-1})^{a-1}\big) = (10^{n-1})^a$ and $\mathrm{tpad}\big(M' = (10^{n-1})^{a^2-1}\big) = (10^{n-1})^{a^2}$, where tpad is the tree padding algorithm of Figure 1. If $\mathcal{TH}_{\mathrm{CE}_5}$ is instantiated with keys $K_1 \| K_2 \| K_3$ such that $K_2^{(1)} = 0$

and $K_3^{(1)} = 1$, then one can verify that $\mathcal{TH}_{\mathrm{CE}_5}(K_1\|K_2\|K_3, M') = \mathcal{TH}_{\mathrm{CE}_5}(K_1\|K_2\|K_3, M) = 1^n$. Hence, the adversary that outputs M and M' as colliding message pair has advantage 1/4 in winning the Coll and eSec games. ▌

XOR Tree. The unstrengthened XOR Tree ($\mathcal{XTH}$) was proposed in [BR97] for fixed-length messages; we consider the strengthened variant here. It is again a variant of the Merkle tree, where the inputs to the compression functions on level i are XORed with a key $K_i \in \{0,1\}^{an}$. As for all other iterations, it is straightforward to see that $\mathcal{XTH}$ preserves ePre; we omit the proof. Quite remarkably, the masking of the entire input to the compression function makes it the only iteration in the literature that preserves Pre, while at the same time it seems to stand in the way of even proving preservation of Coll. It does not preserve aSec or aPre because the adversary can choose $K_i = 0^{an}$ and apply counterexample CE_4. We were unable to come up with either proof or counterexample for Coll, Sec, and eSec, leaving these as an open question. We prove the preservation of Pre below.

Theorem 3.7 If F is (t', ϵ') Pre-secure, then $\mathcal{XTH}_{\mathrm{F}}$ is (t, ϵ) Pre[λ]-secure for all $\lambda \in \mathbb{N}$, $\epsilon = \epsilon'$ and $t = t'$.

Proof: The proof is quite straightforward. The crux actually lies in the fact that, due to the random choice of K_{d+1} that is XORed with the entire input to the compression function, the image of a random message $M \in \{0,1\}^\lambda$ through $\mathcal{XTH}_{\mathrm{F}}$ follows the same distribution as the image of a random message $M \in \{0,1\}^{an}$ through F.

Given a Pre[λ]-adversary A against $\mathcal{XTH}_{\mathrm{F}}$, consider the following Pre-adversary B against F. When B is given a random key K and target point Y, it chooses $K_1, \ldots, K_{d+1} \stackrel{\$}{\leftarrow} \{0,1\}^{an}$ and runs A on input $K\|K_1\|\ldots\|K_{d+1}, Y$. When A returns a preimage M, B computes $h_{d,1}$ as defined in the description of $\mathcal{XTH}_{\mathrm{F}}$ in Figure 1, and returns $\big(h_{d,1}\|\langle|M|\rangle_{n(a-1)}\big) \oplus K_{d+1}$ as its own preimage. ▌

4 The ROX Construction

We are now ready to present in detail our Random-Oracle-XOR (ROX) construction. Let $\mathrm{F} : \{0,1\}^k \times \{0,1\}^{b+n} \to \{0,1\}^n$ be a fixed-length compression function. Let 2^l be the maximum message length in bits; typically one would use $k = 80$ and $l = 64$. The construction uses two random oracles $\mathrm{RO}_1 : \{0,1\}^k \times \{0,1\}^k \times \{0,1\}^{\lceil \log l \rceil} \to \{0,1\}^n$ and $\mathrm{RO}_2 : \{0,1\}^k \times \{0,1\}^l \times \{0,1\}^{\lceil \log b \rceil} \to \{0,1\}^{2n}$. These random oracles can be built from a single one by adding an extra bit to the input that distinguishes calls to RO_1 and RO_2. Our construction can be thought of as a variant of Shoup's hash, but with the masks being generated by RO_1 and the padding being generated by RO_2. More precisely, on input a message M, our padding function rox-pad outputs a sequence of b-bit message blocks

$$m_1\|\ldots\|m_\ell \;=\; M \;\|\; \mathrm{RO}_2(\mathfrak{m}, \langle\lambda\rangle, \langle 1\rangle) \;\|\; \mathrm{RO}_2(\mathfrak{m}, \langle\lambda\rangle, \langle 2\rangle) \;\|\; \ldots\;,$$

where $\mathfrak{m}$ are the first k bits of M and $\lambda = |M|$. The padding adds a number of bits generated by RO_2 such that the final block m_ℓ contains at least $2n$ bits generated by RO_2, possibly resulting in an extra block consisting solely of padding. It is worth noting though that we do not have a separate length strengthening block. We assume that $\lambda \geq k$ because aPre security, and therefore seven-property-preservation as a whole, do not make sense for short messages. Indeed, the adversary can always exhaustively try the entire message space. To hash shorter messages, one should add a random salt to the message.

Let $\nu(i)$ be the largest integer j such that 2^j divides i, let $IV \in \{0,1\}^n$ be an initialization vector, and let $\mathfrak{m}$ be the first k bits of the message M. Our construction is described in pseudocode below; a graphical representation is given in Figure 2.

```
Algorithm $\mathcal{ROX}_{\mathrm{F}}^{\mathrm{RO}_1,\mathrm{RO}_2}(K,M)$:
$m_1\|\ldots\|m_\ell \leftarrow \text{rox-pad}^{\mathrm{RO}_2}(M)$ ; $h_0 \leftarrow IV$
For $i = 0,\ldots,\lfloor\log_2(\ell)\rfloor$ do
    $\mu_i \leftarrow \mathrm{RO}_1(K,\mathfrak{m},\langle i\rangle)$
For $i = 1\ldots\ell$ do
    $g_i \leftarrow h_{i-1} \oplus \mu_{\nu(i)}$ ; $h_i \leftarrow \mathrm{F}(K, m_i\|g_i)$
Return $h_\ell$ .
```

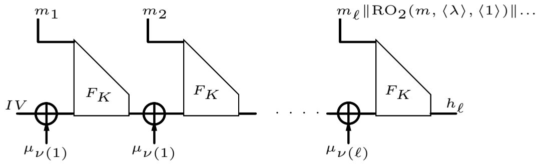

Figure 2: **The ROX Construction.** On the left we give the algorithmic description of ROX. The figure on the right is the graphic representation of ROX. The message is padded with bits generated by $\mathrm{RO}_2(K,\mathfrak{m},\langle\lambda\rangle,\langle i\rangle)$, where $\mathfrak{m}$ are the first k bits of M. The last block must contain at least $2n$ padding bits, otherwise an extra padding block is added. In the picture above, IV is the initialization vector, $\nu(i)$ is the largest integer j such that $2^j|i$, and the masks $\mu_i \leftarrow \mathrm{RO}_1(K,\mathfrak{m},\langle i\rangle)$.

We want to stress that that the ROX construction does not require that the compression function accept an additional input that might be influenced by the attacker (such as a salt or a counter). We see this as an important advantage, since imposing additional requirements on the compression function may make compression functions even harder to design or less efficient.

It is quite standard in cryptography for new primitives to first find instantiations in the random oracle model, only much later to be replaced with constructions in the standard model. It is interesting to see how the random oracles in the ROX construction can be instantiated if one were to implement it in practice. For an 80-bit security level, our results suggest that we should take $k = 80$ and $n = 160$. This means that we need a random oracle that reduces about 170 bits to 160 bits. A first suggestion is to re-use the compression function with, say, three times as many rounds as normal, and with different values of the constants. This approach violates good cryptographic hygiene, however, by having the design of the random oracle depend on that of the surrounding scheme. A better proposal is to use one or more calls to a blockcipher like AES that is designed independently of the compression function.

5 Properties Preserved by the ROX Construction

The following theorem states that the ROX construction preserves all seven security properties that we consider here. We give a proof sketch for the preservation of Coll and a full proof for aSec below; the other proofs can be found in Appendix B. We only note that the proofs for Sec, aSec and eSec are in the programmable random oracle model [Nie02]; that for the case of Pre and aPre non-programmable random oracles suffice; and that Coll and ePre are preserved in the standard model.

Theorem 5.1 For atk $\in$ {Coll, Sec, eSec, aSec, Pre, ePre, aPre}, if the compression function $\mathrm{F} : \{0,1\}^k \times \{0,1\}^{b+n} \to \{0,1\}^n$ is (t',ϵ') atk-secure, then the iterated function $\mathcal{ROX}_{\mathrm{F}}$ is $(t, q_{\mathrm{RO}}, \epsilon)$ atk-secure for

$$\epsilon = \epsilon' + \frac{q_{\mathrm{RO}}^2}{2^{2n}}\,, \qquad t = t' - 2\ell\cdot\tau_{\mathrm{F}} \qquad \text{for atk} = \text{Coll} \tag{2}$$

$$\epsilon = \ell\cdot\epsilon' + \frac{q_{\mathrm{RO}}^2}{2^{2n}}\,, \qquad t = t' - 2\ell\cdot\tau_{\mathrm{F}} \qquad \text{for atk} = \text{Sec} \tag{3}$$

$$\epsilon = \ell\cdot\epsilon' + \frac{q_{\mathrm{RO}}}{2^{k}} + \frac{q_{\mathrm{RO}}^2}{2^{2n}}\,, \qquad t = t' - 2\ell\cdot\tau_{\mathrm{F}} \qquad \text{for atk} = \text{eSec} \tag{4}$$

$$\epsilon = \ell\cdot\epsilon' + \frac{q_{\mathrm{RO}}}{2^{k}} + \frac{q_{\mathrm{RO}}^2}{2^{2n}}\,, \qquad t = t' - 2\ell\cdot\tau_{\mathrm{F}} \qquad \text{for atk} = \text{aSec} \tag{5}$$

$$\epsilon = \epsilon'\,, \qquad t = t' - \ell\cdot\tau_{\mathrm{F}} \qquad \text{for atk} \in \{\text{Pre}, \text{ePre}\} \tag{6}$$

$$\epsilon = \epsilon' + \frac{q_{\mathrm{RO}}}{2^{k}}\,, \qquad t = t' - \ell\cdot\tau_{\mathrm{F}} \qquad \text{for atk} = \text{aPre} \tag{7}$$

Here, τ_{F} is the time required for an evaluation of F and $\ell = \lceil(\lambda + 2n)/b\rceil$ where $\lambda = |M|$.

We repeat that above we do not model the compression function as a random oracle, but it is worth considering what the equations tell us if we do. Assuming for simplicity that $\tau_{\mathrm{F}} = 1$, we know that a collision adversary running in $t' = 2^{n/2}$ steps has probability about 1/2 to find collisions in F, due to the birthday paradox, but only has probability $\epsilon' = 2^{-n/2}$ to find preimages or second preimages. Nevertheless, existing iterations cannot guarantee (second) preimage resistance against $2^{n/2}$-time adversaries, because they merely inherit their (second) preimage resistance by implication from collision resistance.[2] The ROX construction, on the other hand, can. Assuming that queries to $\mathrm{RO}_1, \mathrm{RO}_2$ take unit time and taking $k = n$, Equations (2), (3), (6) imply that an adversary running in time $t = 2^{n/2} - 2\ell \approx 2^{n/2}$ steps has probability at most $\epsilon = \ell \cdot 2^{-n/2} + 2^{n/2-k} + 2^{-n} \approx (\ell + 1) \cdot 2^{-n/2}$ to find second preimages, and has probability at most $\epsilon' = 2^{-n/2} + 2^{n/2-n} \approx 2^{-n/2+1}$ to find preimages.

An interesting question is whether Equation (3) implies that ROX protects against the attack by Kelsey and Schneier [KS05] that finds second preimages in SMD for a 2^{κ}-block message in $\kappa \cdot 2^{n/2+1} + 2^{n-\kappa+1}$ time. On the positive side, the same attack does not seem to extend to ROX, because no two consecutive blocks are ever processed using the same masks. To the best of our knowledge, the only way to build the expandable message structure without ruining the mask schedule is by computing κ-multi-collisions instead of collisions, which come at a cost of $2^{(\kappa-1)n/\kappa}$ each. On the negative side, Equation (3) loses a factor ℓ in the reduction, so in principle it cannot exclude the existence of a Sec attack for $2^{n/2}$-block messages in $2^{n/2}$ work. It does however exclude such attacks for short messages. As shown in Section 3, SMD cannot provide such a guarantee.

Proof of Equation (2): If M, M' is a pair of colliding messages, then consider the two chains of compression function calls in the computation of $\mathcal{ROX}_{\mathrm{F}}(K, M) = \mathcal{ROX}_{\mathrm{F}}(K, M')$. If the inputs to the final call to F are different for M and M', then these inputs form a collision on F and we're done. If they are the same, then remember that at least $2n$ bits of these inputs are the output of $\mathrm{RO}_2(\mathfrak{m}, \langle\lambda\rangle, \langle i\rangle)$ and $\mathrm{RO}_2(\mathfrak{m}', \langle\lambda'\rangle, \langle j\rangle)$, respectively. If these are different queries to RO_2, yet their outputs are the same, then the adversary must have found a collision on RO_2; the odds of it doing so are bounded by $q_{\mathrm{RO}}^2/2^{2n}$. If these queries are the same, however, then we have that $\mathfrak{m} = \mathfrak{m}'$ and $\lambda = \lambda'$, and therefore that the masks in both chains $\mu_i = \mu_i' = \mathrm{RO}_1(K, \mathfrak{m}, \langle i\rangle)$. Identical chaining inputs to ℓ-th call to F must therefore be caused by identical outputs of the $(\ell - 1)$-st call to F. If the inputs to the $(\ell - 1)$-st call are different then we have a collision on F here, otherwise we repeat the argument to the $(\ell - 2)$-nd call, and so on. A collision on F will be found unless $M = M'$. More formally, given a Coll adversary A against $\mathcal{ROX}_{\mathrm{F}}$, we will construct a Coll adversary B against F. On input key K, B runs A on the same key K to obtain two distinct messages M and M', responding to A's random oracle queries by means of the subroutines $\mathsf{RO\text{-}Sim}_1, \mathsf{RO\text{-}Sim}_2$ depicted in Figure 3. Let the messages be parsed as $m_1\| \ldots \|m_\ell \leftarrow \text{rox-pad}^{\mathrm{RO}_2}(M)$ and $m_1'\| \ldots \|m_{\ell'}' \leftarrow \text{rox-pad}^{\mathrm{RO}_2}(M')$. Let $\mathfrak{m}, \mathfrak{m}'$ be the first k bits of M, M', respectively, and let g_i, h_i and g_i', h_i' be the intermediate values obtained during the computation of $\mathcal{ROX}_{\mathrm{F}}^{\mathrm{RO}_1,\mathrm{RO}_2}(K, M)$ and $\mathcal{ROX}_{\mathrm{F}}^{\mathrm{RO}_1,\mathrm{RO}_2}(K, M')$, i.e. $h_0 = h_0' = IV$ and

$$\begin{aligned} g_i &= h_{i-1} \oplus \mu_{\nu(i)} \ , & h_i &= \mathrm{F}(K, m_i \| g_i) \quad \text{where } \mu_{\nu(i)} = \mathrm{RO}_1(K, \mathfrak{m}, \langle\nu(i)\rangle) \\ g_i' &= h_{i-1}' \oplus \mu_{\nu(i)}' \ , & h_i' &= \mathrm{F}(K, m_i \| g_i') \quad \text{where } \mu_{\nu(i)}' = \mathrm{RO}_1(K, \mathfrak{m}', \langle\nu(i)\rangle) \ . \end{aligned}$$

If A is successful, then we know that $h_\ell = h_{\ell'}'$. If $m_\ell \| g_\ell \neq m_{\ell'}' \| g_{\ell'}'$, then these two strings form a pair of colliding inputs to $\mathrm{F}(K, \cdot)$ that B can output. Otherwise, we have that $m_\ell = m_{\ell'}'$. Due to the padding algorithm, each of these strings contains at least one complete $2n$-bit random oracle response $\mathrm{RO}_2(\mathfrak{m}, \langle|M|\rangle, \langle i\rangle)$ and $\mathrm{RO}_2(\mathfrak{m}', \langle|M'|\rangle, \langle i\rangle)$. If these are different queries to RO_2, then the fact that

[2] For the Prefix-free MD [CDMP05] and EMD [BR06] iterations this is a bit paradoxical, because they were designed to preserve "random oracle behavior". Surely, (second) preimage resistance should fall under any reasonable definition of "random oracle behavior"? The caveat here is that the proof [BR06, Theorem 5.2] bounds the distinguishing probability to $O(q_{\mathrm{RO}}^2/2^n)$, so that the theorem statement becomes moot for $q_{\mathrm{RO}} = 2^{n/2}$.

$m_\ell = m'_{\ell'}$ implies that A found a collision in RO_2, in which case B aborts. The probability that A does so using q_{RO} random oracle queries however is at most $q_{\mathrm{RO}}(q_{\mathrm{RO}}-1)/2^{2n+1} \leq q_{\mathrm{RO}}^2/2^{2n}$. So most likely these will actually be identical random oracle queries, meaning that $|M| = |M'| = \lambda$ and $\mathfrak{m} = \mathfrak{m}'$. This means that $\ell = \ell'$ and that $\mu_{\nu(\ell)} = \mu'_{\nu(\ell)}$; hence, the fact that $g_\ell = g'_\ell$ implies that $h_{\ell-1} = h'_{\ell-1}$.

We can now repeat the same argument for $h_{\ell-1}$ and $h'_{\ell-1}$. If $m_{\ell-1}\|g_{\ell-1} \neq m'_{\ell-1}\|g'_{\ell-1}$, then these strings form a pair of colliding inputs to $\mathrm{F}(K,\cdot)$. If they are equal, then because again the masks are the same, it must hold that $h_{\ell-2} = h'_{\ell-2}$. We can continue this argument for $m_{\ell-2}\|g_{\ell-2}$ and $m'_{\ell-2}\|g_{\ell-2}$ and propagate from right to left throughout the chain. However, there must exist an index $I > 0$ such that $h_I = h'_I$ but $m_I\|g_I \neq m'_I\|g'_I$, because otherwise $M = M'$ and the collision is not valid.

Let ABORT be the event that B aborts. It is clear that B wins whenever A wins and B does not abort, meaning that it is successful with probability at least

$$\begin{aligned} \epsilon' &\geq \Pr\left[\,\mathsf{A}\text{ wins} \wedge \overline{\text{ABORT}}\,\right] \\ &= \Pr\left[\,\mathsf{A}\text{ wins} : \overline{\text{ABORT}}\,\right]\cdot\Pr\left[\,\overline{\text{ABORT}}\,\right] \\ &\geq \epsilon\left(1-\frac{q_{\mathrm{RO}}^2}{2^{2n}}\right) \geq \epsilon - \frac{q_{\mathrm{RO}}^2}{2^{2n}}\ . \end{aligned}$$

The running time of B is that of A plus up to 2ℓ compression function evaluations. ▮

Proof of Equation (5): Given an aSec[λ] adversary A against $\mathcal{ROX}_{\mathrm{F}}$ for any $\lambda \in \mathbb{N}$, we will construct an aSec adversary B against F. The overall strategy will be that B "embeds" his own challenge message at a random point in the chain, and hopes that A's output yields a second preimage at exactly the point in the chain where B has embedded his challenge.

Algorithm B runs A to obtain a key $K \in \{0,1\}^k$, responding to its random oracle queries by maintaining associative arrays $T_1[\cdot], T_2[\cdot]$. B outputs the same key K and is then given as input a random challenge message $m\|g \in \{0,1\}^{b+n}$. It chooses a random index $i^* \xleftarrow{\$} \{1,\ldots,\ell = \lceil(\lambda+2n)/b\rceil\}$. We first explain how B can construct a message M of length λ so that $m_{i^*} = m$ in $m_1\|\ldots\|m_\ell \leftarrow \text{rox-pad}^{\mathrm{RO}_2}(M)$; the rest of the message blocks are randomly generated. After that, we will show how g can be embedded into the chain such that $g_{i^*} = g$. If $i^* = 1$ then B sets $\mathfrak{m}$ to the first k bits of m, otherwise it chooses $\mathfrak{m} \xleftarrow{\$} \{0,1\}^k$ and sets the first k bits of M to $\mathfrak{m}$. We distinguish between Type-I message blocks that only contain bits of M, Type-II message blocks of which the first $\lambda_b = (\lambda \bmod b)$ bits are the last λ_b bits of M and the remaining bits are generated by RO_2, and Type-III message blocks that consist entirely of bits generated by RO_2. Embedding m in a Type-I message block can simply be done by setting b bits of M to m starting at bit position $(i^*-1)b+1$. To embed m in a Type-II message block, B sets the last λ_b bits of M to the first λ_b bits of m, and programs the first $(b-\lambda_b)$ bits of $T_2[\mathfrak{m},\langle\lambda\rangle,\langle 1\rangle] \,\|\, T_2[\mathfrak{m},\langle\lambda\rangle,\langle 2\rangle] \,\|\, \ldots$ to be the last $(b-\lambda_b)$ bits of $\mathfrak{m}$. For Type-III blocks, B chooses M completely at random and sets b bits of $T_2[\mathfrak{m},\langle\lambda\rangle,\langle 1\rangle] \,\|\, T_2[\mathfrak{m},\langle\lambda\rangle,\langle 2\rangle] \,\|\, \ldots$ to m, starting at the $(b-\lambda_b+1)$-st bit position. Bits of M and $T_2[\mathfrak{m},\cdot]$ that are still undefined are chosen at random. If any of these table entries were defined during A's first run, then B aborts. Notice however that A's view during the first run is independent of $\mathfrak{m}$, so its probability of making such a query is at most $q_{\mathrm{RO}}/2^k$.

To enforce that $g_{i^*} = g$ in the computation of $\mathcal{ROX}_{\mathrm{F}}^{\mathrm{RO}_1,\mathrm{RO}_2}(K,M)$, algorithm B runs the reconstruction algorithm of [Sho00, Mir01] that, given message blocks $m_1,\ldots,m_{i^*}$ and chaining value g_{i^*}, outputs random mask values $\mu_0,\ldots,\mu_t$ such that the chaining input to the i^*-th compression function call is g_{i^*}. (We recall this algorithm in Figure 3 in appendix.) B's goal is to program these masks into RO_1 by setting $T_1[K,\mathfrak{m},\langle i\rangle] \leftarrow \mu_i$ for $0 \leq i \leq t$, such that it is possible to check that the value for g_{i^*} obtained during the hash computation is indeed g. However, if any of the hash table entries $T_1[K,\mathfrak{m},\langle i\rangle]$ for $0 \leq i \leq t$ has already been defined, then B aborts. This can only occur when A asked a query

$\mathrm{RO}_1(K, \mathfrak{m}, \langle i \rangle)$ during its first phase, but again, the probability of it doing so is at most $q_{\mathrm{RO}}/2^k$ because its view is independent of $\mathfrak{m}$.

Algorithm B then runs A again on input target message M, responding to its random oracle queries as before, until it outputs a second preimage M'. Let $m'_0 \| \ldots \| m'_{\ell'} \leftarrow \text{rox-pad}^{\mathrm{RO}_2}(M')$ be the parsed messages. For the same arguments as in the proof of Equation (2) above, there must exist an index $I > 0$ such that $h_I = h'_I$ but $m_I \| g_I \neq m'_I \| g'_I$, unless A found a collision in the random oracle RO_2. If $i^* = I$, then B outputs $m'_I \| g'_I$.

B wins the game whenever A does and $i^* = I$, unless A succeeded in causing a collision in RO_2 or any of the values that are programmed in $\mathrm{RO}_1, \mathrm{RO}_2$ were already queried. Let E_1 be the event that at least one of the preprogrammed values is queried by A on a different input and E_2 be the event that A manages to find at least one collision in RO_2. Let ABORT be the event that B aborts, then

$$\Pr[\,\text{ABORT}\,] \;=\; \Pr[\,\mathrm{E}_1\,] + \Pr\left[\,\mathrm{E}_2 \;:\; \overline{\mathrm{E}_1}\,\right] \;\leq\; \Pr[\,\mathrm{E}_1\,] + \Pr[\,\mathrm{E}_2\,]\,.$$

Since B perfectly simulates A's environment, the advantage of B is given by

$$\begin{aligned} \epsilon' \;&\geq\; \Pr[\,\mathsf{A}\text{ wins} \wedge i^* = I \;:\; \overline{\text{ABORT}}\,] \cdot \Pr[\,\overline{\text{ABORT}}\,] \\ &\geq\; \frac{\epsilon}{\ell}\left(1 - \left(\frac{q_{\mathrm{RO}}}{2^k} + \frac{q_{\mathrm{RO}}^2}{2^{2n}}\right)\right) \;\geq\; \frac{1}{\ell}\left(\epsilon - \frac{q_{\mathrm{RO}}}{2^k} - \frac{q_{\mathrm{RO}}^2}{2^{2n}}\right)\,. \end{aligned}$$

The running time of B is that of A plus at most 2ℓ evaluations of F. Equation (5) follows. ▌

Proof of Equation (6): Here we present the proof for Pre preservation. Given a Pre[λ] adversary A against $\mathcal{ROX}_{\mathrm{F}}$, we construct a Pre adversary B against F. B, on input a key K and a target value $Y \in \{0,1\}^n$, runs A on the same input K, Y. Note that the target hash value $Y = \mathrm{F}(K, m\|g)$ for randomly chosen $m\|g \in \{0,1\}^{b+n}$ and $K \in \{0,1\}^k$, and it also has the correct distribution for A (in A's game Y is computed on inputs a random message block and the one-but-last chaining output in the computation of $\mathcal{ROX}_{\mathrm{F}}$ XORed with a random mask string).

To simulate the responses of A's random oracle queries, B runs algorithms $\mathsf{RO\text{-}Sim}_1$ and $\mathsf{RO\text{-}Sim}_2$ as specified in Figure 3. Unlike the aSec and eSec games, adversary B here does not need to program any of the entries in either table. Next, B obtains a preimage M' from A such that $\mathcal{ROX}_{\mathrm{F}}^{\mathrm{RO}_1,\mathrm{RO}_2}(K, M') = Y$. Let $m'_1 \| \ldots \| m'_{\ell'} \leftarrow \text{rox-pad}^{\mathrm{RO}_2}(M')$, and let $h'_{\ell'-1}$ be the output of the one-but-last compression function call in the computation of $\mathcal{ROX}_{\mathrm{F}}^{\mathrm{RO}_1,\mathrm{RO}_2}(K, M')$. Let $g'_{\ell'} = h'_{\ell'-1} \oplus \mathrm{RO}_1(K, \mathfrak{m}', \nu(\ell'))$ where $\mathfrak{m}'$ are the first k bits of M'. Finally, B outputs $m'_{\ell'} \| g'_{\ell'}$ as its own valid preimage. B wins the game whenever A does, so its advantage is $\epsilon' \geq \epsilon$. The running time of B is that of A plus at most ℓ evaluations of F. ▌
The rest of the proofs for the ROX construction are presented in Appendix B.

Possible tweaks. The scheme can be simplified not all seven properties need to be preserved. For example, if the key K is dropped from the input to RO_1, the $\mathcal{ROX}$ construction fails to preserve eSec and ePre, but still preserves all other notions. Dropping the message bits $\mathfrak{m}$ from the input of either RO_1 or RO_2 destroys the preservation of aSec and aPre, but leaves the preservation of other notions unharmed.

Acknowledgements

We would like to thank David Cash and 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 and the second by a Postdoctoral Fellowship from the Flemish Research Foundation (FWO - Vlaanderen). The fourth author was supported by NSF CNS-0627752.

References

[ANPS07a] Elena Andreeva, Gregory Neven, Bart Preneel, and Thomas Shrimpton. Seven-property-preserving iterated hashing: ROX. In Kaoru Kurosawa, editor, *Advances in Cryptology – ASIACRYPT 2007*, volume ???? of *Lecture Notes in Computer Science*, pages ???–???, Kuching, Malaysia, December 2–6, 2007. Springer-Verlag, Berlin, Germany. (Cited on page i.)

[ANPS07b] Elena Andreeva, Gregory Neven, Bart Preneel, and Thomas Shrimpton. Three-property preserving iterations of keyless compression functions. ECRYPT Hash Workshop 2007, 2007. (Cited on page 2.)

[BCK96] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In Neal Koblitz, editor, *Advances in Cryptology – CRYPTO'96*, volume 1109 of *Lecture Notes in Computer Science*, pages 1–15, Santa Barbara, CA, USA, August 18–22, 1996. Springer-Verlag, Berlin, Germany. (Cited on page 3.)

[BD06] Eli Biham and Orr Dunkelman. A framework for iterative hash functions – HAIFA. Second NIST Cryptographic Hash Workshop, 2006. (Cited on pages 3 and 8.)

[BJ01] Daniel R. L. Brown and Donald B. Johnson. Formal security proofs for a signature scheme with partial message recovery. In David Naccache, editor, *Topics in Cryptology – CT-RSA 2001*, volume 2020 of *Lecture Notes in Computer Science*, pages 126–142, San Francisco, CA, USA, April 8–12, 2001. Springer-Verlag, Berlin, Germany. (Cited on page 1.)

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73, Fairfax, Virginia, USA, November 3–5, 1993. ACM Press. (Cited on page 1.)

[BR97] Mihir Bellare and Phillip Rogaway. Collision-resistant hashing: Towards making UOWHFs practical. In Burton S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 470–484, Santa Barbara, CA, USA, August 17–21, 1997. Springer-Verlag, Berlin, Germany. (Cited on pages 1, 2, 3, 5, 7, 8, 9, 10 and 20.)

[BR06] Mihir Bellare and Thomas Ristenpart. Multi-property-preserving hash domain extension: The EMD transform. In Xuejia Lai and Kefei Chen, editors, *Advances in Cryptology – ASIACRYPT 2006*, volume 4284 of *Lecture Notes in Computer Science*, pages 299–314, Shanghai, China, December 3–7, 2006. Springer-Verlag, Berlin, Germany. (Cited on pages 3, 8 and 12.)

[BR07] Mihir Bellare and Thomas Ristenpart. Hash functions in the dedicated-key setting: Design choices and MPP transforms. In L. Arge, C. Cachin, and A. Tarlecki, editors, *34th International Colloquium on Automata, Languages and Programming – ICALP 2007*, volume 4596 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, Germany, 2007. (Cited on page 3.)

[CDMP05] Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya. Merkle-Damgård revisited: How to construct a hash function. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 430–448, Santa Barbara, CA, USA, August 14–18, 2005. Springer-Verlag, Berlin, Germany. (Cited on pages 3, 8 and 12.)

[CS03] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167–226, 2003. (Cited on page 1.)

[Dam90] Ivan Damgård. A design principle for hash functions. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 416–427, Santa Barbara, CA, USA, August 20–24, 1990. Springer-Verlag, Berlin, Germany.

(Cited on pages 1, 3 and 5.)

[Dea99] Richard D. Dean. *Formal Aspects of Mobile Code Security*. PhD thesis, Princeton University, 1999. (Cited on pages 1 and 8.)

[HK06] Shai Halevi and Hugo Krawczyk. Strengthening digital signatures via randomized hashing. In Cynthia Dwork, editor, *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 41–59, Santa Barbara, CA, USA, August 20–24, 2006. Springer-Verlag, Berlin, Germany. (Cited on pages 3 and 8.)

[KK06] John Kelsey and Tadayoshi Kohno. Herding hash functions and the Nostradamus attack. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 183–200, St. Petersburg, Russia, May 28 – June 1, 2006. Springer-Verlag, Berlin, Germany. Available from `http://eprint.iacr.org/2005/281`. (Cited on pages 3 and 8.)

[KS05] John Kelsey and Bruce Schneier. Second preimages on n-bit hash functions for much less than 2^n work. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 474–490, Aarhus, Denmark, May 22–26, 2005. Springer-Verlag, Berlin, Germany. (Cited on pages 1, 8 and 12.)

[LCL$^+$03] Wonil Lee, Donghoon Chang, Sangjin Lee, Soo Hak Sung, and Mridul Nandi. New parallel domain extenders for UOWHF. In Chi-Sung Laih, editor, *Advances in Cryptology – ASIACRYPT 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 208–227, Taipei, Taiwan, November 30 – December 4, 2003. Springer-Verlag, Berlin, Germany. (Cited on page 9.)

[LM92] Xuejia Lai and James L. Massey. Hash functions based on block ciphers. In Rainer A. Rueppel, editor, *Advances in Cryptology – EUROCRYPT'92*, volume 658 of *Lecture Notes in Computer Science*, pages 55–70, Balatonfüred, Hungary, May 24–28, 1992. Springer-Verlag, Berlin, Germany. (Cited on page 1.)

[LR89] Michael Luby and Charles Rackoff. A study of password security. *Journal of Cryptology*, 1(3):151–158, 1989. (Cited on page 1.)

[Luc05] Stefan Lucks. A failure-friendly design principle for hash functions. In Bimal K. Roy, editor, *Advances in Cryptology – ASIACRYPT 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 474–494, Chennai, India, December 4–8, 2005. Springer-Verlag, Berlin, Germany. (Cited on page 3.)

[Mer80] Ralph C. Merkle. Protocols for public key cryptosystems. In *IEEE Symposium on Security and Privacy*, pages 122–134. IEEE Computer Society Press, 1980. (Cited on pages 3 and 9.)

[Mer90a] Ralph C. Merkle. A certified digital signature. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 218–238, Santa Barbara, CA, USA, August 20–24, 1990. Springer-Verlag, Berlin, Germany. (Cited on page 1.)

[Mer90b] Ralph C. Merkle. One way hash functions and DES. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 428–446, Santa Barbara, CA, USA, August 20–24, 1990. Springer-Verlag, Berlin, Germany. (Cited on page 3.)

[Mir01] Ilya Mironov. Hash functions: From Merkle-Damgård to Shoup. In Birgit Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 166–181, Innsbruck, Austria, May 6–10, 2001. Springer-Verlag, Berlin, Germany. (Cited on pages 1, 2, 3, 13 and 22.)

[Mir06] Ilya Mironov. Collision-resistant no more: Hash-and-sign paradigm revisited. In Moti Yung, editor, *PKC 2006: 9th International Workshop on Theory and Practice in Public Key Cryptography*, volume 3958 of *Lecture Notes in Computer Science*, pages 140–156, New

York, NY, USA, April 24–26, 2006. Springer-Verlag, Berlin, Germany. (Cited on page 3.)

[Nie02] Jesper Buus Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 111–126, Santa Barbara, CA, USA, August 18–22, 2002. Springer-Verlag, Berlin, Germany. (Cited on page 11.)

[NY89] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *21st Annual ACM Symposium on Theory of Computing*, pages 33–43, Seattle, Washington, USA, May 15–17, 1989. ACM Press. (Cited on page 1.)

[Rog06] Phillip Rogaway. Formalizing human ignorance: Collision-resistant hashing without the keys. In *Vietcrypt 2006*, volume 4341 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, Germany, 2006. (Cited on page 2.)

[RS04] Phillip Rogaway and Thomas Shrimpton. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In Bimal K. Roy and Willi Meier, editors, *Fast Software Encryption 2004*, volume 3017 of *Lecture Notes in Computer Science*, pages 371–388. Springer-Verlag, Berlin, Germany, 2004. (Cited on pages i, 1, 2, 3, 4 and 5.)

[Sar05] Palash Sarkar. Masking-based domain extenders for UOWHFs: bounds and constructions. *IEEE Transactions on Information Theory*, 51(12):4299–4311, 2005. (Cited on page 9.)

[Sho00] Victor Shoup. A composition theorem for universal one-way hash functions. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 445–452, Bruges, Belgium, May 14–18, 2000. Springer-Verlag, Berlin, Germany. (Cited on pages 1, 2, 3, 8 and 13.)

[WG00] David Wagner and Ian Goldberg. Proofs of security for the Unix password hashing algorithm. In Tatsuaki Okamoto, editor, *Advances in Cryptology – ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 560–572, Kyoto, Japan, December 3–7, 2000. Springer-Verlag, Berlin, Germany. (Cited on page 1.)

[WY05] Xiaoyun Wang and Hongbo Yu. How to break MD5 and other hash functions. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 19–35, Aarhus, Denmark, May 22–26, 2005. Springer-Verlag, Berlin, Germany. (Cited on page 1.)

[WYY05] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding collisions in the full SHA-1. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 17–36, Santa Barbara, CA, USA, August 14–18, 2005. Springer-Verlag, Berlin, Germany. (Cited on page 1.)

A Proofs and Counterexamples for Existing Constructions

A.1 Proof of Counterexample CE_1

Lemma A.1 If G is (t', ϵ') atk-secure for some atk $\in \{\mathrm{Sec}, \mathrm{aSec}, \mathrm{Pre}, \mathrm{aPre}\}$, then CE_1 is (t, ϵ) atk-secure for all $\epsilon = \epsilon' + 1/2^n$ and $t = t'$.

Proof: We present the proofs for the case of Sec and Pre separately here. Those of aSec and aPre are almost identical and therefore omitted.

atk = Sec Given an adversary A that (t, ϵ)-breaks the Sec-security of CE_1, we build an adversary B against the Sec-security of G. Algorithm B, on input a random message $M = m \| h \in \{0,1\}^{b+n}$ and a random key $K \in \{0,1\}^k$, runs A on the same input (M, K). B obtains A's second preimage $M' = m' \| h'$ and outputs it as its own colliding message. We say that M is in Case 1 if $h = IV$ and in Case 2 if $h \neq IV$; Cases 1 and 2 for M' are defined analogously. If A is successful, i.e. $\mathrm{CE}_1(K, M) = \mathrm{CE}_1(K, M')$ and $M \neq M'$, then M and M' must be in the same case, due to the last bit of the hash value being different in both cases. Since M is chosen at random, it is only in Case 1 with probability $1/2^n$. If M and M' are both in Case 2 then we must have that $\mathrm{G}(K, M) = \mathrm{G}(K, M')$, so M' is a valid second preimage for M with respect to $\mathrm{G}(K, \cdot)$. The running time of B is equal to that of A since all it does is relaying inputs and outputs. Its advantage is given by

$$\begin{aligned} \mathbf{Adv}^{\mathrm{G}}_{\mathrm{Sec}}(\mathsf{B}) &= \Pr[\,\mathsf{A} \text{ wins } \wedge h \neq IV\,] \\ &\geq \Pr[\,\mathsf{A} \text{ wins}\,] - \Pr[\,h = IV\,] \\ &= \mathbf{Adv}^{\mathrm{CE}_1}_{\mathrm{Sec}}(\mathsf{A}) - 1/2^n \ . \end{aligned}$$

atk = Pre Given a Pre-adversary A against CE_1, we build a Pre-adversary B against G. Algorithm B, on input a random key K and a hash value $Y \leftarrow \mathrm{G}(K, M)$ for a random message M, runs A with input $Y \| \overline{IV}_n$ to obtain a preimage M'. B returns M' as its own output.

Let $M = m \| h$ be the message chosen at random by the experiment. (Note that B only sees the hash value Y, not M itself.) When $h \neq IV$, the input to A is correctly distributed and B wins its game whenever A does. In the event where $h = IV$ however, the input to A is not correctly distributed: it is given $\mathrm{G}(K, M) \| \overline{IV}_n$, while it should be given $\mathrm{CE}_1(K, M) = IV$. This event only occurs with probability $1/2^n$, and thus yields

$$\mathbf{Adv}^{\mathrm{G}}_{\mathrm{Pre}}(\mathsf{B}) \geq \mathbf{Adv}^{\mathrm{CE}_1}_{\mathrm{Pre}}(\mathsf{A}) - 1/2^n \ .$$

The running time of B is simply equal to that of A. ▌

A.2 Proof of Counterexample CE_2

Lemma A.2 For any atk $\in \{\mathrm{Coll}, \mathrm{eSec}\}$, if G is (t', ϵ') atk-secure, then CE_2 is (t, ϵ) atk-secure for all $\epsilon = \epsilon'$ and $t = t'$.

Proof: We present the proofs for Coll and eSec separately.

atk = Coll Given an adversary A that (t, ϵ)-breaks the Coll-security of CE_2, we build the following adversary B against the Coll-security of G. Algorithm B, on input a random key K, runs A on input the same key K, to obtain distinct colliding messages M and M' from A. B outputs M and M' as its own colliding pair. We claim that B succeeds whenever A does, or that $\mathrm{G}(K, M) = \mathrm{G}(K, M')$ whenever $\mathrm{CE}_2(K, M) = \mathrm{CE}_2(K, M')$.

To see this, consider the following cases for (K, M). We say that (K, M) is in Case 1 if $M = 010^{b-2} \| IV$; in Case 2 if $K^{(1)} = 0$ and $M = \langle 1 \rangle_b \| IV$, or if $K^{(1)} = 1$ and $M = \langle b+1 \rangle_b \| IV$; and in Case 3 otherwise. Cases 1, 2, and 3 for (K, M') are defined analogously. For M and M' to collide, (K, M) and (K, M') have to be in the same case, because the last two bits of the hash value are different for different cases. They cannot be both in Case 1, because then $M = M'$. They cannot be both in Case 2, because for a single value of K that would mean $M = M'$ as well. So they have to be both in Case 3, which means that $\mathrm{G}(K, M) = \mathrm{G}(K, M')$, so M, M' is a valid collision on $\mathrm{G}(K, \cdot)$. Thus, we measure the advantage of B by

$$\mathbf{Adv}^{\mathrm{G}}_{\mathrm{Coll}}(\mathsf{B}) \quad \geq \quad \mathbf{Adv}^{\mathrm{CE}_2}_{\mathrm{Coll}}(\mathsf{A}) \; .$$

The running time of B is equal to that of A since all it does is relaying inputs and outputs.

<u>atk = eSec</u> Given an eSec adversary against CE_2, we build an eSec adversary B against the compression function G. B runs A to obtain a target message M and outputs the same message as its target. On obtaining a random key $K \in \{0,1\}^k$, B runs A on the same key K until it outputs its colliding message M'. B outputs the same M' as its second preimage.

By a case analysis similar to that for the case of Coll above, one can show that that $\mathrm{G}(K, M) = \mathrm{G}(K, M')$ whenever $\mathrm{CE}_2(K, M) = \mathrm{CE}_2(K, M')$, and hence that

$$\mathbf{Adv}^{\mathrm{G}}_{\mathrm{eSec}}(\mathsf{B}) \;\geq\; \mathbf{Adv}^{\mathrm{CE}_2}_{\mathrm{eSec}}(\mathsf{A}) \; .$$

The running time of B is the same as that of A. ▌

A.3 Proof of Counterexample CE_3

Lemma A.3 If G is (ϵ', t') atk-secure for some atk $\in \{\mathrm{Sec}, \mathrm{Pre}\}$ then CE_3 is (t, ϵ) atk-secure for all $\epsilon = \epsilon' + 1/2^b$ and $t = t'$.

Proof: We prove the lemma for atk = Sec and atk = Pre separately.

<u>atk = Sec</u> Given an adversary A that (t, ϵ)-breaks the Sec-security of CE_3, consider the following adversary B against the Sec-security of G. Algorithm B, on input a random key $K \in \mathcal{K}$ and a random message $m \| h \in \{0,1\}^{b+n}$, runs A on the same input $(K, m\|h)$ to obtain a second preimage M'. B returns the same message $m' \| h'$ as its own second preimage. If A is successful, meaning that $\mathrm{CE}_4(K, m\|h) = \mathrm{CE}_4(K, m'\|h')$ and $m\|h \neq m'\|h'$, we distinguish between the case that $m = \langle \lambda \rangle_b$ and the case that $m \neq \langle \lambda \rangle_b$. In the former case, $\mathrm{CE}_3(K, m\|h') = 0^n$ for any $h' \in \{0,1\}^n$, but since m is chosen at random, this case only occurs with probability $1/2^b$. In the latter case, we also have that $m' \neq \langle \lambda \rangle_b$ because otherwise the last bits of the hashes are different. Therefore, we must have that $\mathrm{G}(K, m\|h) = \mathrm{G}(K, m'\|h')$.

The running time of B is the same as that of A since all it does is relaying inputs and outputs. It is clear that B wins whenever A wins and B does not abort, meaning that it is successful with probability

$$\begin{aligned} \mathbf{Adv}^{\mathrm{G}}_{\mathrm{Sec}}(\mathsf{B}) \quad &\geq \quad \Pr\left[\,\mathsf{A} \text{ wins} \;\wedge\; m \neq \langle \lambda \rangle_b \,\right] \\ &\geq \quad \mathbf{Adv}^{\mathrm{CE}_3}_{\mathrm{Sec}}(\mathsf{A}) - \frac{1}{2^b} \; . \end{aligned}$$

<u>atk = Pre</u> Given a Pre-adversary A against CE_3, we construct the following Pre-adversary B against G. Algorithm B, on input a random key K and a hash value $Y \leftarrow \mathrm{G}(K, m\|h)$ (computed for random K and $m\|h$), runs A on input $Y\|1$ to obtain a preimage $m'\|h'$. B returns $m'\|h'$ as its own preimage.

It is easy to see that when $m \neq \langle\lambda\rangle_b$, then the input to A is correctly distributed and B wins its game whenever A does. In the event that $m = \langle\lambda\rangle_b$, the input to A is not correctly distributed, but this only occurs with probability $1/2^b$, thus yielding

$$\mathbf{Adv}^{\mathrm{G}}_{\mathrm{Pre}}(\mathsf{B}) \;\geq\; \mathbf{Adv}^{\mathrm{CE}_3}_{\mathrm{Pre}}(\mathsf{A}) - 1/2^b \ .$$

▮

A.4 Proofs for Strengthened Merkle Tree and Counterexample CE_4

Theorem A.4 If F is (t', ϵ') Coll-secure, then $\mathcal{SMT}_{\mathrm{F}}$ is (t, ϵ) Coll-secure for $\epsilon = \epsilon'$ and $t = t' - 2\ell \cdot \tau_{\mathrm{F}}$.

Proof: Given a Coll-adversary A against $\mathcal{SMT}_{\mathrm{F}}$, we build the following Coll-adversary B against the Coll-security of F. On input a key K, B runs A on the same input. A then outputs distinct colliding messages M and M'. Let $h_{i,j}$ and $h'_{i,j}$ be the intermediate hash values generated in the computation of $\mathcal{SMT}(K, M) = h_{d+1,1}$ and $\mathcal{SMT}(K, M') = h'_{d'+1,1}$ as depicted in Figure 1. If $|M| \neq |M'|$, then $(h_{d,1}\|\langle|M|\rangle_{(a-1)n}) \neq (h'_{d',1}\|\langle|M'|\rangle_{(a-1)n})$ form a valid pair of colliding messages for $\mathrm{F}(K, \cdot)$. Else if $|M| = |M'|$, then the generated trees for M and M' are of equal depth $d = d'$. If $h_d \neq h'_{d',1}$ then B proceeds as in the previous case. Otherwise, it goes up one level. If $h_{d-1,1}\| \dots \|h_{d-1,a} \neq h'_{d-1,1}\| \dots \|h'_{d-1,a}$, then these are a valid collision. Otherwise, it goes up another level and checks whether $h_{d-2,(i-1)a+1}\| \dots \|h_{d-2,ia} \neq h'_{d-2,(i-1)a+1}\| \dots \|h'_{d-2,ia}$ for some $1 \leq i \leq n$. If so, then these form a colliding pair. This process continues until a colliding pair is found, which must happen unless $M = M'$. The running time of B equals that of A plus up to 2ℓ compression function evaluations. ▮

Theorem A.5 For $k \in \mathbb{N}$, if there exists a (t', ϵ') eSec-secure compression function $\mathrm{G} : \{0,1\}^k \times \{0,1\}^{an} \rightarrow \{0,1\}^{n-k}$, then there exists a (t, ϵ) eSec-secure compression function $\mathrm{CE}_6 : \{0,1\}^k \times \{0,1\}^{an} \rightarrow \{0,1\}^n$ with $\epsilon = \epsilon' + 1/2^{k-1}$, $t = t'$. There exists an adversary running in constant time with eSec[λ]-advantage $1 - 1/2^k$ in breaking $\mathcal{MT}_{\mathrm{CE}_6}$ for some $\lambda \in \mathbb{N}$.

Proof: Given the compression function G, consider CE_6 given by

$$\begin{array}{rcll} \mathrm{CE}_6(K, m_1\|m_2) & = & 1^n & \text{if } m_1 = K \\ & = & K\|\mathrm{G}(K, m_1\|m_2) & \text{otherwise ,} \end{array}$$

where $|m_1| = k$ and $|m_2| = 2n - k$.

Lemma A.6 If G is (ϵ', t') eSec- secure then CE_4 is (t, ϵ) eSec-secure for all $\epsilon = \epsilon' + 1/2^{k-1}$ and $t = t'$.

The only difference with the counterexample of [BR97] is the placement of the key inside the hash (at the beginning instead of at the end). Their proof is easily seen to extend to the lemma above. When iterated as $\mathcal{SMT}_{\mathrm{CE}_4}$, one can see that any message of length $a^{2i} - 1 \leq \lambda \leq a^{2i+1} - 2$ for $i \in \mathbb{N}$ hashes to 1^n, leading to easy constant-time eSec attacks. ▮

Lemma A.7 If G is (ϵ', t') atk secure for some $\mathrm{atk} \in \{\mathrm{Sec}, \mathrm{aSec}, \mathrm{Pre}, \mathrm{aPre}\}$ then CE_4 is (t, ϵ) atk-secure for all $\epsilon = \epsilon' + 1/2^{n-1}$ and $t = t'$.

Proof: We prove the lemma for atk = Sec and atk = Pre separately. The proofs for aSec and aPre-security are almost identical, and they are omitted here.

atk = Sec Given an adversary A that (t, ϵ)-breaks the Sec-security of CE_4, consider the following adversary B against the Sec-security of G. Algorithm B, on input a random key $K \in \mathcal{K}$ and a random message $M = m_1 \| \ldots \| m_a \in \{0,1\}^{an}$ and , runs A on the same inputs to obtain a second preimage $M' = m'_1 \| \ldots \| m'_a$. B returns the same message as its own preimage. If A is successful, meaning that $\mathrm{CE}_4(K, M) = \mathrm{CE}_4(K, M')$ and $M \neq M'$, we distinguish between the cases that (1) $m_a = 0^n$, that (2) $m_{a-1} = 0^n$ and $m_a \neq 0^n$, and (3) that $m_{a-1} \neq 0^n$ and $m_a \neq 0^n$. Since M is chosen at random, the first two cases happen with probability $1/2^n$ each. In the third case, we also have that $m'_{a-1} \neq 0^n$ and $m'_a \neq 0^n$, because otherwise the last two bits of the hashes of M and M' are different. Therefore, we have that $\mathrm{G}(K, M) = \mathrm{G}(K, M')$, so B can output M' as its own second preimage. The running time of B is equal to that of A since all it does is relaying inputs and outputs. The advantage of B is that of A minus the probability that we end up in cases (1) or (2), i.e. $\epsilon' \geq \epsilon - 2/2^n$.

atk = Pre Given a Pre-adversary A against CE_4, we construct the following Pre-adversary B against G. Algorithm B, on input a random key K and a hash value $Y \leftarrow \mathrm{G}(K, M)$ (computed for random M and K), runs A on input $Y \| 10$ to obtain a preimage M'. B returns M' as its own preimage.

Let $M = m_1 \| m_a$ be the message chosen at random by the experiment. It is easy to see that when $m_{a-1} \neq 0^n$ and $m_a \neq 0^n$, then the input to A is correctly distributed and B wins its game whenever A does. Otherwise, the input to A is not correctly distributed, because the target value should be $\mathrm{CE}_4(K, M) = 1^n$ or $\mathrm{CE}_4(K, M) = 0^n$. These events only occur with probability $1/2^n$ each, thus yielding $\epsilon' \geq \epsilon - 2/2^n$. The running time of B is simply equal to that of A. ▌

A.5 Proof of Counterexample CE_5

Lemma A.8 For any atk $\in$ {Coll, eSec}, if G is (t', ϵ') atk-secure then CE_5 is (t, ϵ) atk-secure for all $\epsilon = \epsilon'$ and $t = t'$.

Proof: We first describe the reductions for both Coll and eSec, and then explain why they are successful. For the case of Coll, given a Coll-adversary A against CE_5, consider the Coll-adversary B against G that, given a key K, runs $M, M' \overset{\$}{\leftarrow} \mathsf{A}(K)$ and outputs M, M'. For the case of eSec, given an eSec-adversary A against CE_5, consider the eSec-adversary B against G that runs A to obtain a target message M from A, and outputs M as its own target message. On obtaining a random key K, B runs A on input K until it returns a second preimage M'. B returns the same second preimage M'.

We say that M is in Case 1 if $\mathrm{CE}_5(K, M) = 10^{n-1}$, corresponding to the first line of CE_5 above, in Case 2 if $\mathrm{CE}_5(K, M) = 1^n$, corresponding to the second line, and in Case 3 otherwise, corresponding to the last line. The hash values of different cases are distinct, so for M, M' to be a valid collision, they must be in the same case. They cannot be both in Case 1 or 2, because for a single key K there is only a single message in each case. Therefore, they must both be in Case 3, so that $\mathrm{CE}_5(K, M) = \mathrm{CE}_5(K, M')$ implies that $\mathrm{G}(K, M) = \mathrm{G}(K, M')$, meaning that M, M' is also a valid collision for G. Hence, B is successful whenever A is. The running time of B is also equal to that of A. ▌

B Proofs of the ROX Construction (Theorem 5.1)

Proof of Equation (3): Given a Sec[λ]-adversary A against $\mathcal{ROX}_{\mathrm{F}}$, we build the following Sec-adversary B against F. B is given as input a random key $K \in \{0,1\}^k$ and challenge message $m \| g \in$

```
Algorithm MaskRec$(K, m_1, \ldots, m_r, g)$
  $j \leftarrow r$ ; For $i = 0, \ldots, t = \lfloor \log_2 r \rfloor$ do $\mu_i \leftarrow \bot$
  Repeat while $j > 0$
    $j' \leftarrow j - 2^{\nu(j)}$ ; $g' \xleftarrow{\$} \{0,1\}^n$
    For $i = j'+1, \ldots, j-1$ do if $\mu_i = \bot$ then $\mu_i \xleftarrow{\$} \{0,1\}^n$
    If $j' = 0$ then $h_0 \leftarrow IV$ else $h_{j'} \leftarrow \mathrm{F}(K, m_{j'} \| g')$
    For $i = j'+1, \ldots, j-1$ do
      $g_i \leftarrow h_{i-1} \oplus \mu_{\nu(i)}$ ; $h_i \leftarrow \mathrm{F}(K, m_i \| g_i)$
    $\mu_{\nu(j)} \leftarrow g_{j-1} \oplus g'$ ; $j \leftarrow j'$ ; $g \leftarrow g'$
  For $i = 0, \ldots, t$ do if $\mu_i = \bot$ then $\mu_i \xleftarrow{\$} \{0,1\}^n$
  Return $(\mu_0, \ldots, \mu_t)$
```

```
Algorithm RO-Sim$_1(K, \mathfrak{m}, x)$
  If $T_1[K, \mathfrak{m}, x] = \bot$ then
    $T_1[K, \mathfrak{m}, x] \xleftarrow{\$} \{0,1\}^n$
  Return $T_1[K, \mathfrak{m}, x]$
```

```
Algorithm RO-Sim$_2(\mathfrak{m}, x, y)$
  If $T_2[\mathfrak{m}, x, y] = \bot$ then
    $T_2[\mathfrak{m}, x, y] \xleftarrow{\$} \{0,1\}^{2n}$
  Return $T_2[\mathfrak{m}, x, y]$
```

Figure 3: **Algorithms used in the proofs of our construction.** On the left, we recall the mask reconstruction algorithm of [Mir01] (M is parsed as b-bit blocks). On the right, we give the algorithms used to simulate the random oracles (m_0 is the first s-bit block of M).

$\{0,1\}^{b+n}$. It chooses a random index $i^* \xleftarrow{\$} \{1, \ldots, \ell = \lceil (\lambda + 2n)/b \rceil\}$. If $i^* = 1$, then $\mathfrak{m}$ is set to equal the first k bits of m; otherwise, B chooses $\mathfrak{m} \xleftarrow{\$} \{0,1\}^k$ and sets the first k bits of M to $\mathfrak{m}$. It also maintains associative arrays $T_1[\cdot]$, $T_2[\cdot]$ to simulate A's random oracle queries.

B needs to "embed" its own challenge message at a random point in the chain, and construct a full target message M of length λ for A. B proceeds as shown in the Proof of Equation (5). To enforce that $g_{i^*} = g$ in the computation of $\mathcal{ROX}_{\mathrm{F}}^{\mathrm{RO}_1,\mathrm{RO}_2}(K, M)$, algorithm B runs the mask reconstruction algorithm of Figure 3 to generate mask values $(\mu_0, \ldots, \mu_t) \xleftarrow{\$} \mathsf{MaskRec}(K, m_1, \ldots, m_{i^*}, g)$. It programs these into RO_1 by setting $T_1[K, \mathfrak{m}, \langle i \rangle] \leftarrow \mu_i$ for $0 \leq i \leq t$. This way, one can check that the value for g_{i^*} obtained during the hash computation is indeed g. The difference from the aSec game in the Sec one is that B programs the mask values into RO_1 before A is run (thus A has not made any random oracle queries so far) and B has no chance to abort here.

After generating M as a concatenation of random blocks with $m_{i^*} = m$, B runs A on inputs K and mask values $\mu_0 \ldots \mu_t$ to obtain a second preimage M'. As explained in the Proof of Equation (5), in the computation of $\mathcal{ROX}_{\mathrm{F}}$ messages M and M' collide, if a collision occurs either in the final F call for $|M| \neq |M'|$ or for $|\mathfrak{m} \neq \mathfrak{m}'|$, or in an internal F call for $|M| = |M'|$ and $\mathfrak{m} = \mathfrak{m}'$. Note however that in the former case B aborts when a collision in RO_2 occurs. As shown in the Proof of Equation (5), this happens only with probability less or equal than $q_{\mathrm{RO}}^2/2^{2n}$.

It is clear that B wins the game whenever A does and $i^* = I$, unless A succeeded in causing a collision in RO_2. Let $\textsc{Coll}$ be the event that A manages to find at least one collision in RO_2. Since B perfectly simulates A's environment, the advantage of B given by

$$\begin{aligned} \epsilon' &\geq \Pr\left[\mathsf{A}\text{ wins} \wedge i^* = I \wedge \overline{\textsc{Coll}}\right] \\ &= \Pr\left[\mathsf{A}\text{ wins} \wedge i^* = I \,:\, \overline{\textsc{Coll}}\right] \cdot \Pr\left[\overline{\textsc{Coll}}\right] \\ &\geq \frac{\epsilon}{\ell}\left(1 - \frac{q_{\mathrm{RO}}^2}{2^{2n}}\right) \geq \frac{1}{\ell}\left(\epsilon - \frac{q_{\mathrm{RO}}^2}{2^{2n}}\right) . \end{aligned}$$

The running time of B is that of A plus at most 2ℓ evaluations of F. Equation (3) follows. ▮

Proof of Equation (4): Given an eSec-adversary A against $\mathcal{ROX}_{\mathrm{F}}$, we build the following eSec-adversary B against F. B runs A to obtain the target message M. Let $|M| = \lambda$ and let $\mathfrak{m}$ be the first k

bits of M. Throughout this game B simulates the answers to A's queries to RO_1 and RO_2 by running algorithms RO-Sim$_1$ and RO-Sim$_2$ as specified in Figure 3.

B chooses an index $i^* \xleftarrow{\$} \{1, \ldots, \ell = \lceil (\lambda + 2n)/b \rceil\}$. As its challenge message $m \| g$, B sets m to be the i^*-th message block of $m_1 \| \ldots \| m_\ell$ where $m_1 \| \ldots \| m_\ell \leftarrow \text{rox-pad}^{\mathrm{RO}_2}(M)$ and chooses $g \xleftarrow{\$} \{0,1\}^n$. It submits $m \| g$ to the challenger and gets the compression function key K in exchange.

To enforce that $g = g_{i^*}$ in the computation of $\mathcal{ROX}_{\mathrm{F}}^{\mathrm{RO}_1,\mathrm{RO}_2}(K, M)$, algorithm B runs the mask reconstruction algorithm of Figure 3 to generate mask values $(\mu_1, \ldots, \mu_t) \xleftarrow{\$} \mathsf{MaskRec}(K, m_1, \ldots, m_{i^*}, g)$. It programs these into RO_1 by setting $T_1[K, \mathfrak{m}, \langle i \rangle] \leftarrow \mu_i$ for $0 \leq i \leq t$. This way, one can check that the value for g_{i^*} obtained during the hash computation is indeed g. If any of the hash table entries $T_1[K, \mathfrak{m}, \langle i \rangle]$ were already defined, then B aborts. This means that A was able to predict K though, which happens with probability $q_{\mathrm{RO}}/2^k$.

Algorithm B then runs A again on input K until it outputs its second preimage M'. As explained in the Proof of Equation (5), in the computation of $\mathcal{ROX}_{\mathrm{F}}$ messages M and M' collide, if a collision occurs either in the final F call for $|M| \neq |M'|$ (or for $\mathfrak{m} \neq \mathfrak{m}'$), or for an internal F call in the chain for $|M| = |M'|$ and $\mathfrak{m} = \mathfrak{m}'$. As shown in the Proof of Equation (5), whenever a collision in RO_2 occurs (padding collision), B aborts.

B wins the game whenever A does and $i^* = I$, unless A succeeded in causing a collision in RO_2 or any of the mask values to be programmed in RO_1 for a given input has been already output by the RO_1 on a different input, in which cases B aborts. Let E_1 be the event that at least one of the preprogrammed masks is queried on a different input and E_2 be the event that A manages to find at least one collision in RO_2. Let ABORT be the event that B aborts, then

$$\begin{aligned} \Pr[\,\textsc{abort}\,] &= \Pr[\,\mathrm{E}_1\,] + \Pr[\,\mathrm{E}_2 \,:\, \overline{\mathrm{E}_1}\,] \\ &\leq \Pr[\,\mathrm{E}_1\,] + \Pr[\,\mathrm{E}_2\,] \,. \end{aligned}$$

Since B perfectly simulates A's environment, the advantage of B is given by

$$\begin{aligned} \epsilon' &\geq \Pr[\,\mathsf{A}\text{ wins} \wedge i^* = I \wedge \overline{\textsc{abort}}\,] \\ &= \Pr[\,\mathsf{A}\text{ wins} \wedge i^* = I \,:\, \overline{\textsc{abort}}\,] \cdot \Pr[\,\overline{\textsc{abort}}\,] \\ &\geq \Pr[\,\mathsf{A}\text{ wins} \wedge i^* = I \,:\, \overline{\textsc{abort}}\,] \cdot (1 - (\Pr[\,\mathrm{E}_1\,] + \Pr[\,\mathrm{E}_2\,])) \\ &\geq \frac{\epsilon}{\ell}\left(1 - (\frac{q_{\mathrm{RO}}}{2^k} + \frac{q_{\mathrm{RO}}^2}{2^{2n}})\right) \geq \frac{1}{\ell}\left(\epsilon - \frac{q_{\mathrm{RO}}}{2^k} - \frac{q_{\mathrm{RO}}^2}{2^{2n}}\right) \,. \end{aligned}$$

The running time of B is that of A plus at most 2ℓ evaluations of F. Equation (5) follows. ∎

Proof of Equation (6): atk = ePre Given an ePre adversary A against $\mathcal{ROX}_{\mathrm{F}}$, we construct an ePre adversary B against F. B runs A to obtain target point Y, where $Y \in \{0,1\}^n$. B outputs the same target value Y to get a random key input $K \in \{0,1\}^k$. On obtaining the same key K, A outputs a preimage M', such that $\mathcal{ROX}_{\mathrm{F}}^{\mathrm{RO}_1,\mathrm{RO}_2}(K, M') = Y$. B outputs $m'_{\ell'} \| g'_{\ell'}$ as its own preimage. Here $m'_{\ell'}$, $g'_{\ell'}$ and the random oracle responses to A throughout the game are defined as in the proof of Pre above. B wins the game whenever A does and and its advantage is $\epsilon' \geq \epsilon$. The running time of B is that of A plus at most ℓ evaluations of F. Equation (6) follows. ∎

Proof of Equation (7): Given an aPre[λ] adversary A against $\mathcal{ROX}_{\mathrm{F}}$, we construct an aPre adversary B against F for any $\lambda \in \mathbb{N}$. B runs A to obtain a key K. To simulate the responses of A's random oracle queries, B runs algorithms RO-Sim$_1$ and RO-Sim$_2$ as specified in Figure 3. B also outputs a key K and obtains a target point $Y = \mathrm{F}(K, m\|g)$ for a randomly chosen $m\|g \in \{0,1\}^{b+n}$. A gets the same target point Y from B.

We have to argue why the target point Y is correctly distributed to serve as an input for A. Namely, A expects a target point following the distribution induced by $\mathrm{F}(K, m_\ell\|g_\ell)$ where m_ℓ, g_ℓ are as obtained in the computation of $\mathcal{ROX}_{\mathrm{F}}^{\mathrm{RO}_1,\mathrm{RO}_2}(K, M)$ for a random message $M \in \{0,1\}^\lambda$, while that of B is induced by a random choice of m_ℓ, g_ℓ. These distributions can be seen to be identical when conditioned on the event that A did not query RO_1 or RO_2 on the correct value for $\mathfrak{m}$, the first k bits of M, during its first execution, which happens with probability at least $1 - (q_{\mathrm{RO}}/2^k)$. On input Y, algorithm A outputs a preimage M' such that $\mathcal{ROX}_{\mathrm{F}}^{\mathrm{RO}_1,\mathrm{RO}_2}(K, M') = Y$. B outputs $m'_{\ell'}\|g'_{\ell'}$ as its own preimage, where $m'_{\ell'}$, $g'_{\ell'}$ are as defined in the Proof of Equation (7).

B wins the game whenever A does, unless the distribution of the target value Y for A differs from that of B. Since that happens with probability only $q_{\mathrm{RO}}/2^k$, we measure the advantage of B by

$$\epsilon' \quad \geq \quad \epsilon - \frac{q_{\mathrm{RO}}}{2^k} \ .$$

The running time of B is that of A plus at most ℓ evaluations of F. Equation (7) follows. ▌