Provable Second Preimage Resistance Revisited

Charles Bouillaguet[1(✉)] and Bastien Vayssière[2]

[1] LIFL, Université Lille-1, Lille, France
charles.bouillaguet@lifl.Fr
[2] PRISM Lab, Université de Versailles/Saint-Quentin-en-Yvelines, Versailles, France
Bastien.Vayssiere@prism.uvsq.fr

Abstract. Most cryptographic hash functions are iterated constructions, in which a mode of operation specifies how a compression function or a fixed permutation is applied. The Merkle-Damgård mode of operation is the simplest and more widely deployed mode of operation, yet it suffers from generic second preimage attacks, even when the compression function is ideal.

In this paper we focus on provable security against second preimage attacks. Based on the study of several existing constructions, we describe simple properties of modes of operation and show that they are sufficient to allow some form of provable security, first in the random oracle model and then in the standard model. Our security proofs are extremely simple. We show for instance that the claims of the designers of Haifa regarding second preimage resistance are valid.

Lastly, we give arguments that proofs of second preimage resistance by a black-box reduction incur an unavoidable security loss.

Keywords: Hash function · Second preimage resistance · Security proof · Unavoidable security loss · Black-box reductions

1 Introduction

Of all major cryptographic primitives, hash functions have been continuously avoiding the theoretical nirvana other cryptographic primitives enjoy. While ciphers, encryption schemes, message authentication codes and signature schemes have well understood theoretical foundations, acknowledged security definitions, and some can be idealized using primitives which are considered natural and fair, hash functions have remained as elusive as they were. There is however a consensus to consider that a cryptographic hash function $H : \{0,1\}^* \to \{0,1\}^n$ cannot be "good" if it does not simultaneously resist:

1. Collision adversaries up to about $2^{n/2}$ queries
2. Preimage adversaries up to about 2^n queries
3. Second-Preimage adversaries up to about 2^n queries

T. Lange, K. Lauter, and P. Lisoněk (Eds.): SAC 2013, LNCS 8282, pp. 513–532, 2014.
DOI: 10.1007/978-3-662-43414-7_26, © Springer-Verlag Berlin Heidelberg 2014

Many modern hash functions are usually built by combining a *compression function*, hashing a small number of bits (typically 512) into a smaller number (typically 256), and of a *mode of operation*, describing how the compression function should be used to process arbitrarily big messages. The most popular and well-known mode of operation is the *Merkle-Damgård* construction, introduced in 1989 and named after its two independent inventors [6,14]. Besides its simplicity and elegance, the most distinctive feature of this mode of operation is that it promotes the collision-resistance of the compression function to that of the full hash function. Indeed, its inventors proved that there exist an efficient algorithm which, given two messages $M \neq M'$ such that $H^f(M) = H^f(M')$, extracts two compression-function inputs $x \neq x'$ such that $f(x) = f(x')$, where H^f denotes the Merkle-Damgård iteration of the compression function f.

The Merkle-Damgård mode of operation therefore enjoys a form of *provable security*, since the whole hash function is not less secure than the compression function with respect to collision adversaries. This allows hash function designers to focus on designing collision-resistant compression functions, arguably an easier task than designing a full-blown hash function. A comparable result holds for preimage resistance, since a preimage on the full hash function would lead to a pseudo-preimage on the compression function.

The situation is however not as good for the remaining classical security notion, namely second preimage resistance. In fact, it turned out that the Merkle-Damgård iteration of a secure compression function is not as secure as the compression function itself: in 2005, Kelsey and Schneier described an attack [12] that finds a second preimage of an ℓ-block message with $2^n/\ell$ evaluations of the compression function, even if it is ideal (i.e., a public random function).

The existence of several generic attacks [10–12] demonstrated that there was definitely a problem with the Merkle-Damgård construction, and motivated further research, and new modes of operations have emerged. It also motivated hash function designers to provide *proofs* that their mode of operation is sounds, and that it does not suffer from generic attacks.

An elegant solution, both theoretically and practically appealing, is the *wide-pipe* hash proposed by Lucks in 2005 [13]. The underlying idea is simple: make the internal state twice as big as the output. This makes the construction provably resistant to second preimage attacks in the standard model, because a second preimage on the iteration yields either an n-bit second preimage or a $2n$-bit collision on the compression function. This construction is also very practical, and it is implemented by 4 out of the 5 SHA-3 finalists. However, the memory footprint of a wide-pipe construction is as least twice as big compared to Merkle-Damgård, so in some cases where memory is restricted, it would be beneficial to have a "narrow-pipe" mode of operation.

In this paper, we focus on *narrow-pipe*[1] modes of operation, where several questions remain unanswered. For instance, the exact resistance to generic second preimage attack of the Merkle-Damgård construction is in fact unknown.

[1] We call "narrow-pipe" a construction where the internal state has the same length as the digest.

Existing attacks give an upper-bound above the birthday paradox, and the fact that a second preimage is also a collision give a birthday lower-bound. The generic second preimage security of Merkle-Damgård is thus known to lie somewhere between $2^{n/2}$ and $2^n/\ell$ queries, for messages of size ℓ.

Our Goal and Our Results. The objective of this paper is to describe very simple conditions that, when satisfied by a narrow-pipe mode of operations, are sufficient to provide some form of provable resistance against second preimage attacks beyond the birthday bound.

Provable security against second preimage attack comes in several flavors. One possible setting to discuss the security of a mode of operation is the *random oracle model*, i.e., assuming that the compression function is a public random function. Proofs that there cannot exist second preimage attacks under the assumption that the compression function is a random oracle show that the mode of operation is immune to *generic attacks*, i.e., attacks that target the mode of operation itself and thus work for any compression function. The second preimage attacks of Kelsey-Schneier and that of Andreeva *et al.* [2] are generic attacks.

We show that a simple tweak to the Merkle-Damgård mode is sufficient to prevent all generic second preimage attacks. This modification, namely the inclusion of a round counter, is one of the distinctive features of Haifa. Biham and Dunkelman proposed Haifa in 2006 [8], as a collection of tweaks to the original Merkle-Damgård mode of operation; they claimed a security level of 2^n against second preimage adversaries, without providing proofs. We thus show that their claim is valid.

The assumption that hash functions, or just components thereof, are random, is strong and unrealistic enough to make some uncomfortable, so that we would like to get rid of it. Constructions of *keyed* hash functions provably achieving a form of second preimage resistance without relying on the existence of public random functions, but instead based on the hardness of a general assumption have been known for quite a while [9,15], under the name of *Universal One-Way Hash Functions* (UOWHFs). Later on, modes of operation of keyed hash functions that promote a form of second preimage resistance from the compression function to the whole construction have been designed [4,17].

The security of the latter modes of operation is established by a *black-box reduction*, namely an algorithm that turns a successful attacker against the hash function into a (somewhat less) successful attacker against the compression function. Thus, the iteration remains secure, up to some level, as long as the compression functions are themselves secure.

Inspired by these constructions we again isolate a specific property of modes of operation which is sufficient to provide this kind of "reductionist" security, without heavy assumptions on the compression function. This feature is, again, simple: given a bit string x, it must be possible to forge a message M such that $f(x)$ is evaluated while computing $H^f(M)$. We then describe a "generic" reduction that solely requires this specific property to show that a mode of

operation promotes the second preimage resistance of the compression function. This proof is, to some extent, an abstraction of the security proofs of several existing schemes.

Lastly, we observe that in all these proofs of second preimage security by reduction there is always a *security loss* proportional to the size of hashed messages (i.e., security is guaranteed up to a level of $2^n/\ell$ where ℓ denotes the size of hashed messages). We give arguments hinting that this security loss is unavoidable, and is caused by the proof technique itself.

Organisation of the Paper. In Sect. 2 we recall the security notions we are concerned with. Then in Sect. 3 we introduce a generic narrow-pipe mode of operation, and we show that all the particular constructions that we consider are instances of this generic framework. In Sect. 4 we discuss the generic attacks that apply to the known provably second-preimage resistant constructions we consider, and we show how to make them immune to these attacks. Lastly, in Sect. 5 we show our main result, namely that the security loss in the security proofs is unavoidable.

2 Definitions

We recall the definition of the usual second preimage notions. The Spr notion is folklore and applies to unkeyed hash functions, while Sec and eSec security notions have been defined in [16] and applies to families of hash functions indexed by a key.

- **Spr** The adversary receives a (random) challenge M and has to find a second message M' such that $H(M) = H(M')$ with $M \neq M'$. The advantage of the adversary is its success probability (taken over the random coins used by the adversary and the random choice of the challenge).
- **Sec** The adversary receives a random challenge message and a random key, and she has to produce a colliding message for the given key. The advantage is the success probability of the adversary (over the random coins used by the adversary and the random choice of the challenge).
- **eSec** The adversary chooses the challenge message. Then, she receives a random key and has to find a colliding message under this key. The advantage is the maximum taken over the choice of M by the adversary of her success probability (taken over the random coins used and the random choice of the key).

Historically, eSec-secure hash function families have been called Universal One-Way Hash Functions (UOWHFs). It must be noted that a Sec-adversary can be used to win the eSec security game (just generate the challenge message randomly-first). Therefore, if $H^{(\cdot)}$ is eSec-secure, then it is also Sec-secure.

Note that the size of the challenges plays an important role in the discussion of second preimage resistance. The known generic attacks are faster when the

challenges become longer. For this reason, the second preimage security notions are often parametrized by the size of the challenges. When the challenge consists of an ℓ-block long message, the notions are denoted by $\mathsf{Spr}[\ell], \mathsf{Sec}[\ell]$ and $\mathsf{eSec}[\ell]$.

We say that an adversary against a security notion (t, ε)-*breaks* the security notion if it terminates in time t and wins the game with probability ε. Let us note a fact that will have some importance later on. In all these notions, the success probability is taken over the random coins of the adversary *and over the choice of the challenge*. This means that an adversary implementing an attack against "weak messages" or "weak keys" may succeed on a small fraction of the challenge space and fail systematically on non-weak messages, while still having a non-zero advantage. A consequence is that it is not possible to increase the success probability of adversaries against a single challenge by repeating them until they succeed.

How to compare the efficiency of adversaries that have different running time and success probability? If an adversary (t, ε)-breaks a security notion, then the expected number of repetitions of the experiment defining the notion before the adversary wins is $1/\varepsilon$. This represents a total of t/ε time units, and this is a meaningful scale. Intuitively, it represents "how much time do we have to wait before the adversary shows me what she is capable of". We call the *global complexity* of an adversary the ratio between its time complexity and its success probability. As an example, notice that the global complexity of exhaustive search is 2^n (for all second preimage notions).

Following the notations in use in the existing literature, we will denote by $\mathcal{A}_H$ an adversary against an iterated hash function, and by $\mathcal{A}_f$ the adversary against the corresponding compression function. Hopefully, things will be clear by the context.

3 Abstract Narrow-Pipe Modes of Operations

Because we would like to state results that are as generic as possible, we introduce a framework of abstract modes of operation, which encompasses all the narrow-pipe modes of operation known to the authors. This framework will enable us to show that our results hold for any mode of operation satisfying a minimum set of conditions.

We will consider that a *narrow-pipe mode of operation* $H^{(\cdot)}$ is a circuit that takes as its input M (the full message), K (the key, if present), h_i (the current chaining value) and i (the block counter). This circuit is responsible for preparing the input to the compression function. The next chaining value h_{i+1} is the output of the compression function on the input prepared by the circuit. The output of the whole hash function is the output of the compression function on its last invocation. The circuit activate a special wire "last call" to indicate that the hash process is terminated. We denote by $\mathfrak{e} : \mathbb{N} \to \mathbb{N}$ the function that returns the number of calls to the compression function given the size of M. We thus implicitly assume that the number of calls to the compression function does not depend on the input of the hash function (i.e., on M and K), but only on the size of M. We are inclined to believe that this restriction is natural.

The incoming chaining value is set to a predefined value (say, zero) on the first invocation. This particular class of modes of operation imposes that the output of the full hash function comes out of the compression function without post-treatment, in particular without truncation. This, coupled with the fact that the circuit has no internal memory makes it a narrow-pipe mode of operation. Apart from that, $H^{(\cdot)}$ may include a "final transformation", or process each message block multiple times. Formally, the hash process works according to the pseudo-code shown in Algorithm 1.

Algorithm 1. Formal definition of the hash process with an abstract mode of operation

```
function ABSTRACT-MODE-OF-OPERATION(M, K)
    h_{-1} ← 0
    i ← 0
    while not finished do
        x_i ← H^(·)(M, K, i, h_{i-1})
        h_i ← f(x_i)
        i ← i + 1
    end while
    return h_{i-1}
end function
```

There are constructions that are apparently not narrow-pipe, but that still fit in this framework, such as the GOST hash function (the checksum can be computed in the last invocation, and does not need to be transmitted between each invocation of the compression function). Note that this requires the full message M to be given to the mode of operation at each invocation.

Note that by choosing $H^{(\cdot)}$ to be a *circuit*, we implicitly admit the existence of an upper-bound on the size of the messages (if only because the block counter comes on a finite number of wires). In the sequel, by "mode of operation", we implicitly mean "a narrow-pipe mode of operation that fits the above framework". This does not seem to be a restriction, as we are not aware of any narrow-pipe construction using a single compression function that does not fit the above definition.

3.1 Collision-Resistance Preserving Modes of Operation

While we tried to make the definition of a mode of operation as generic as it gets, we are not interested in *really bad* modes of operation. We are not interested in non-collision resistant constructions, for instance. In this section, we characterize a few properties modes of operation should have not to be totally worthless.

We say that a mode of operation **is strengthened** if the binary encoding of the size of the processed message is contained in the input to the last invocation of the compression function. It is well-known that the Merkle-Damgård

mode of operation is strengthened, which is the key in establishing its important collision-resistance preservation. However, in general, being strengthened is not completely sufficient to be collision-resistance preserving. Some further technicalities are required.

We say that a mode of operation is **message-injective** if for all functions f and all keys K, the function that maps the message M to the sequence of compression-function inputs (x_i) is injective. This implies that hashing two different messages M and M' cannot generate the same sequence of inputs (x_i). This property is necessary for collision-resistance preservation: if $H^{(\cdot)}$ is not message-injective, there exists a function f and a key K such that there exist two colliding messages M and M' generating the same hash, without causing a collision in the compression function.

We also say that a mode of operation is **chaining-value-injective** if for all f and all K, there exists a (deterministic) function that maps x_i to h_{i-1}. The combination of these three properties is sufficient to ensure collision-resistance preservation.

Lemma 1. *A mode of operation $H^{(\cdot)}$ simultaneously message-injective, chaining-value-injective and strengthened is collision-resistance preserving.*

This lemma is just a restatement of the well-known result of Merkle and Damgård, but we include its proof, because it is a good warm-up, and because it will be useful later on.

Proof. Suppose we have two messages $M \neq M'$ such that $H^f(K, M) = H^f(K, M')$, for some compression function f. Then:

- Either $\left|M\right| \neq \left|M'\right|$. In this case, because $H^{(\cdot)}$ is strengthened, the inputs of the last invocation of the compression are not the same when hashing M and M', and because M and M' collide, we have found a collision on f (on its last invocation).
- Or $\left|M\right| = \left|M'\right|$. Suppose that the compression function is invoked $r = \mathfrak{e}(|M|)$ times in both cases. In this case, there are again two possibilities. Either $x_r \neq x'_r$, and we have a collision since $h_r = h'_r$, or $x_r = x'_r$. By chaining-value-injectivity, we have $h_{r-1} = h'_{r-1}$. The argument repeats. Either we find a collision along the way, or we reach the conclusion that $x_i = x'_i$, for all i, which is impossible by message-injectivity. □

Because of this lemma, we call a mode $H^{(\cdot)}$ "collision-resistance preserving" if it satisfies these three conditions.

3.2 Some Particular Modes of Operations

We briefly describe the Merkle-Damgård mode of operation and HAIFA, as well as the three provably second-preimage resistant modes of operations mentioned in the introduction. Figure 1 shows a possible implementation of the corresponding modes of operation in our generic framework.

```
function MD(M, K, i, h_{i-1})
    let (m_0, ..., m_ℓ) ← Pad(M)
    return (h_{i-1}, m_i)
end function
```

```
function HAIFA(M, K, i, h_{i-1})
    let (m_0, ..., m_ℓ) ← Pad(M)
    return (h_{i-1}, m_i, i)
end function
```

```
function SHOUP(M, K, i, h_{i-1})
    let (k, μ_0, ..., μ_κ) ← K
    let (m_0, ..., m_ℓ) ← Pad(M)
    return (k, h_{i-1} ⊕ μ_{ν_2(i)}, m_i)
end function
```

```
function SPLIT-PADDING(M, K, i, h_{i-1})
    let (m_0, ..., m_ℓ) ← Special-Pad(M)
    return (h_{i-1}, m_i)
end function
```

```
function BCM(M, K, i, h_{i-1})
    let (K_1, K_2) ← K
    let (m_0, ..., m_ℓ) ← Pad(M)
    if i = 0 then return (K_0 ⊕ m_1, m_0)
    if 0 < i < ℓ − 1 then return (h_{i-1} ⊕ m_{i+1}, m_i)
    if i = ℓ − 1 then return (h_{ℓ-2} ⊕ m_ℓ ⊕ K_2, m_ℓ ⊕ K_1)
    if i = ℓ then return (h_{ℓ-1} ⊕ K_1, m_ℓ ⊕ K_2)
end function
```

Fig. 1. Pseudo-code of possible implementations of the modes of operations considered in Sect. 3.2 in the generic framework for narrow-pipe constructions.

Merkle-Damgård. The Merkle-Damgård mode of iteration was independently suggested in 1989 by Merkle [14] and Damgård [6]. It is an unkeyed mode of operation, so the circuit $H^{(\cdot)}$ just ignores the key input. In this mode, the input to the compression function is usually considered to be formed of two parts playing different roles: the chaining value input, on n bits, and the message block input, on m bit, the output of the function being n-bit wide.

The padding is done usually by appending a single '1' bit followed by as many '0' bits as needed to complete an m-bit block including the length of M in bits (the well-known Merkle-Damgård strengthening). However, for the sake of simplicity, we will consider in the sequel a *simplified* padding scheme: the last block is padded with zeroes, and the message length in bits is included in an extra block.

HAIFA. The HAsh Iterative FrAmework (HAIFA), introduced in 2006 by Biham and Dunkelman [8], is a Merkle-Damgård-like construction where a counter and salt are added to the input of the compression function. In this paper, we consider a *simplified* version of HAIFA (amongst other things, we disregard the salt). For our purposes, the definition we use is of course equivalent. In HAIFA, the compression function $f\colon \{0,1\}^n \times \{0,1\}^m \times \{0,1\}^{64} \to \{0,1\}^n$ takes three inputs: the chaining value, the message block, and the round counter (we arbitrarily limit the number of rounds to 2^{64}). The designers of HAIFA claimed that the round counter was sufficient to prevent all generic second preimage attacks.

Shoup's UOWHF. Shoup's Universal One-Way Hash Function works just like Merkle-Damgård by iterating an eSec-secure compression function family $f : \{0,1\}^k \times \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$ to obtain a (keyed) eSec-secure hash function (i.e., a UOWHF).

The scheme uses a set of masks $\mu_0, \ldots, \mu_{\kappa-1}$ (where $2^\kappa - 1$ is the length of the longest possible message), each one of which is a random n-bit string. The key of the whole iterated function consists of the key k of the compression function and of these masks. The size of the key is therefore logarithmic in the maximal size of the messages that can be hashed. The order in which the masks are applied is defined by a specified sequence: in the i-th invocation of the compression function, the $\nu_2(i)$-th mask is used, where $\nu_2(i)$ denotes the largest integer ν such that 2^ν divides i. As advertised before, this construction enjoys a form of provable second-preimage security in the standard model: it promotes the eSec security of the compression function to that of the whole hash function.

Theorem 1 [17]**.** *Let $H^{(\cdot)}$ denote Shoup's mode of operation. If an adversary is able to break the $\mathsf{eSec}[\ell]$ notion of H^f with probability ε in time T, then one can construct an adversary that breaks the eSec notion of f in time $T + \mathcal{O}(\ell)$, with probability ε/ℓ.*

The Backwards Chaining Mode. Andreeva and Preneel described in [3] the *Backwards Chaining Mode* (BCM) which promotes the second-preimage resistance of an unkeyed compression function to the Sec notion of the (keyed) full hash function. We will assume for the sake of simplicity that the message block and the chaining values have the same size. The iteration is keyed, and the key is formed by a triplet (K_0, K_1, K_2) of n-bit strings (note that the size of the key is independent of the size of the messages).

This construction also enjoys a form of provable second-preimage security in the standard model. It promotes the Spr security of the compression function to the Sec-security of the whole hash function.

Theorem 2 [3]**.** *Let $H^{(\cdot)}$ denote the BCM mode of operation. If an adversary is able to break the $\mathsf{Sec}[\ell]$ notion of H^f with probability ε in time T, then one can construct an adversary that breaks the Spr notion of f in time $T + \mathcal{O}(\ell)$, with probability ε/ℓ.*

The Split Padding. Yasuda introduced the *Split Padding* in 2008 [18], as a minor but clever tweak to the Merkle-Damgård strengthening. For the sake of simplicity, we will assume that the message block is twice bigger than the chaining values (i.e., it is $2n$-bit wide). The tweak ensures that any message block going into the compression function contains at least n bits from the original message (this is not necessarily the case in the last block of the usual Merkle-Damgård padding scheme).

It promotes a kind of eSec-security of the compression function to the Spr-security of the (unkeyed) iteration. More precisely, the security notion required of

the compression function is the following: the adversary chooses a chaining value h and the first n bits of the message block m_1, and is then challenged with the last n bits of the message block m_2. She has to find a new pair $(h', m') \neq (h, m_1 \,||\, m_2)$ such that $f(h, m_1 \,||\, m_2) = f(h', m')$. To some extent, this is the eSec security notion, but here the "key" of the compression function is the last n bits of the message block.

Theorem 3 [18]**.** *Let $H^{(\cdot)}$ denote the Split Padding mode of operation. If an adversary is able to break the Spr$[\ell]$ notion of H^f with probability ε in time T, then one can construct an adversary that breaks the eSec-like notion of f in time $T + \mathcal{O}(\lambda)$, with probability ε/ℓ.*

4 How to Make Your Mode of Operation Resistant Against Second Preimage Attacks?

In this section, we describe two simple properties of modes of operation, and we show that these properties allows some kind of security results against second preimage adversaries.

4.1 Resistance Against Generic Attacks

Generic attacks are attacks against the modes of operation, i.e., attacks that do not exploit any property of the compression function, and that could therefore work regardless of its choice. Generic attacks can therefore break the hash function even if the compression function does not have any weakness, and they could work even if the compression function were a random oracle (a public, perfectly random function).

Symmetrically, an attack against a hash function where the compression is perfectly random is necessarily an attack against the mode of operation (since it is impossible to break a perfectly random function).

We will therefore follow the existing literature [1,2,7,10–12] by assuming that the compression function is random. In the random oracle model, the relevant measure of efficiency of an adversary is the number of query sent to the random oracle, rather than time. Indeed, the adversaries cannot obtain any kind of advantage by computation alone without querying the random function. In this particular setting, we say that an adversary (q, ε)-breaks a security notion if she sends at most q queries to the random oracle and wins with probability at least ε.

We now show that a very simple criterion, directly inspired from Haifa, is sufficient to obtain an optimal level of provable resistance to generic second preimage attacks.

Definition 1. *A mode of operation $H^{(\cdot)}$ has* domain separation *if there exist a deterministic algorithm* **idxEx** *which, given an input to the compression function x_i produced when evaluating $H^f(K, M)$, recovers i, regardless of the choice of M, K and f.*

Amongst all the modes of operation considered above, only HAIFA has domain separation: the round counter is part of the input to the compression function. The following theorem show that HAIFA is optimally resistant to generic second preimage attacks, as was claimed by its designers.

Theorem 4. *Let $H^{(\cdot)}$ be a mode of operation satisfying the conditions of Lemma 1 and also having domain separation, and let f be a public random function. Let $\mathcal{A}$ be a second-preimage adversary that (q, ε)-break the $\mathsf{Spr}[\ell]$ notion for H^f. Then:*

$$\varepsilon \leq q/2^{n-1}.$$

Proof. Suppose that the adversary, challenged with an ℓ-block message M, succeeds and finds $M' \neq M$ such that $H^f(M) = H^f(M')$. Then:

1. Either $\left|M\right| \neq \left|M\right|$, and because $H^{(\cdot)}$ is strengthened, then the adversary has found a (second) preimage of $H^f(M)$ for the compression function f. Since f is a random oracle, each query has a probability 2^{-n} to give this preimage.
2. Or M and M' have the same size. Because $H^{(\cdot)}$ is strengthened, injective and extractable, we know (by looking at the proof of Lemma 1) that there exists a collision on f of the form:

$$f(x_i) = f(x'_i) = h_i$$

It is important to notice that the same value of i occurs in the three members of this equation. The "index extractor" **idxEx** of the domain separation mechanism can be used to partition the possible inputs to f into disjoint classes (corresponding to the preimages of integers). In the collision above, x_i and x'_i belong to the same, "i-th" class. When submitting a query x to f, the adversary implicitly chooses the index $i = \mathbf{idxEx}(x)$ of the class to which x belong. The collision above can only be found if $f(x) = h_{\mathbf{idxEx}(x)}$, meaning that for each query, there is only one target value that ensures victory. Therefore, because f is a random oracle, each query hits the single target with probability 2^{-n}.

Now, each query sent by the adversary has probability $2^{-n} + 2^{-n}$ of fulfilling a sufficient success condition, which proves the result. □

4.2 Resistance Against All Attacks

The assumption that the compression function is random is the crux of the proof of the previous result. While it is completely unrealistic, results proved under this assumption still say something meaningful: they show that the mode of operation itself does not exhibit obvious weaknesses, and that the adversaries have to look into the compression function to break the iteration.

Nevertheless, it would be more satisfying to drop this requirement. In that case, the adversary "knows" the source code of the compression function, so that she does not need an external oracle interface to evaluate it. The relevant

measure of her complexity is thus her running time. We say that an adversary (t, ε)-break a hash function (or a compression function) if she runs in time at most t and succeeds with probability at least ε.

For this, we show that another simple criterion is enough to offer a non-trivial level of security. This criterion is directly inspired by the three constructions with provable security in the standard model discussed above.

Definition 2. *Given a mode of operation $H^{(\cdot)}$ and a compression function f, let $P(i, y)$ denote the set of pairs (M, K) such that when evaluating $H^f(M, K)$, then the i-th input to f is y (i.e., $x_i = y$ in Algorithm 2).*

We say that a mode of operation $H^{(\cdot)}$ allows for embedding *if $P(i, y) \neq \emptyset$ for any y and if it is computationally easy to sample random elements in $P(i, y)$.*

Shoup's UOWHF allows for embedding, yet proving it is not so easy. We refer the reader to [17] for the full details, but here is an intuitive version. Controlling the message block in the i-th iteration is easy, but controlling the chaining value is not so obvious. Clearly, the mask used in the i-th iteration must be chosen carefully, but the problem is that choosing it will also randomize the output of the previous iterations. The key idea is that between two arbitrary points of the iteration, there is always a mask that is used only *once* (the one with the greatest index). By choosing this particular mask *after* all the others, it is possible to control the chaining value at this particular point, regardless of the other masks. This yields a recursive procedure to control the chaining value between the first and the i-th iterations: observe that the chaining value can be set to (say) zero in the iteration where the mask with the greatest index occur before the i-th iteration, independently of what happens afterward. Suppose that this mask happens in iteration j. Then, we are left with the problem of controlling the chaining value between the j-th and the i-th iteration, a strictly smaller problem, to which the same technique can be applied recursively.

The backwards chaining mode easily allows for embedding. To embed in the first block, just set K_0 appropriately. To embed at any other index smaller than $\ell - 1$, just choose m_i and m_{i+1} with care. Finally, to embed at index $\ell - 1$ or ℓ, pick the message at random and choose K_1 and K_2 accordingly (the keys are necessary to embed in the last blocks because of the padding scheme). The split-padding does not allows for this definition of embedding, but it allows to embed n bits of message block into any compression function input.

Theorem 5. *Let $H^{(\cdot)}$ be a mode of operation satisfying the hypotheses of Lemma 1 and that additionally allows for embedding.*

If an adversary is able to break the $\mathsf{Sec}[\ell]$ notion of H^f with probability ε in time T, then one can construct an adversary that breaks the Spr notion of f in time $T + \mathcal{O}(\mathfrak{e}(\ell))$, with probability $\varepsilon/\mathfrak{e}(\ell)$.

Proof. The proof works by exhibiting a reduction $\mathcal{R}$ that turns an adversary $\mathcal{A}_H$ against the iteration into an adversary against the compression function. The reduction $\mathcal{R}$ is described by the pseudo-code of Algorithm 2.

The reduction starts by forging a random message M that "embeds" the challenge x at a random position i, and then it sends this to the adversary $\mathcal{A}_H$. If the adversary succeeds in producing a second preimage M', then M and M' collide. If the collision happen just at position i, then a second preimage of the challenge x is readily found.

The sequence of compression function inputs (the x_i in Algorithm 2) generated during the iteration of $H^f(M, K)$ is denoted by $blocks(f, M, K)$.

Algorithm 2. Formal definition of the generic reduction.

```
1: function REDUCTION[ℓ](x)
2:     i ←$ {0, 1, ..., 𝔢(ℓ)}
3:     (M, K) ←$ P(i, x)
4:     M' ← A_H(f, M, K)
5:     if M' = ⊥ then return ⊥
6:     x_0, ..., x_{𝔢(ℓ)−1} ← blocks(f, M, K)
7:     x'_0, ..., x'_{𝔢(ℓ')−1} ← blocks(f, M', K)
8:     j ← 1
9:     while x_{𝔢(ℓ)−j} = x'_{𝔢(ℓ')−j} do
10:        j ← j + 1
11:    if 𝔢(ℓ) − j = i then return x'_{𝔢(ℓ')−j} else ⊥
12: end function
```

The running time of the reduction is clearly that of $\mathcal{A}_H$ plus the time needed to hash both M and M'. Clearly, M' cannot be larger that the running time of $\mathcal{A}_H$, so that the running time of $\mathcal{R}$ is essentially that of the adversary.

It remains to determine the success probability of the reduction. First of all, the adversary succeeds with probability ε on line 3. Note that the challenge fed to $\mathcal{A}_H$ is uniformly random: the challenge x given to $\mathcal{R}$ is supposed to be chosen uniformly at random, and (M, K) is uniformly random amongst the possibilities that place the random block x at a random position i.

Next, we show that when the adversary $\mathcal{A}_H$ succeeds, the reduction itself succeeds with probability $1/\mathfrak{e}(\ell)$. First, we claim that at the beginning of line 11, we have $x_{\mathfrak{e}(\ell)-j} \neq x'_{\mathfrak{e}(\ell')-j}$ and $f\left(x_{\mathfrak{e}(\ell)-j}\right) = f\left(x'_{\mathfrak{e}(\ell')-j}\right)$. The reasoning behind this is exactly the same as that in the proof of Lemma 1. This establishes the correctness of the reduction in passing.

Finally, we see that the reduction succeeds if and only if $\mathfrak{e}(\ell) - j = i$. Because i has been chosen uniformly at random, this happens with probability $1/\mathfrak{e}(\ell)$, regardless of the value of j (which is under the control of the adversary). □

Discussion. All the proof of resistance considered above (Theorems 1, 2, 3 and 5) only provide a security level of $2^n/\ell$. In some cases, this makes perfect sense, because a generic attack of this complexity is applicable. However, such generic attacks could be made impossible by including a counter in the mode of

operation, and yet it seems impossible to provide better security proofs in the standard model.

It is then natural to ask whether these security proofs could be improved to reflect the effect of the patch on the security of the schemes. In other terms, we ask whether it is it possible to prove the patched schemes resistant to second preimage attacks in the standard model up to a level of roughly 2^n?

The last contribution of this paper is to show that this is in fact impossible with the "usual" proof technique.

5 Unavoidable Security Loss in Black-Box Reduction

Resistance against second preimage attacks in the standard model of a mode of operation $H^{(\cdot)}$ is often announced by theorem formulated similar to the following "typical" result.

Theorem 6 (informal and typical). *There is a black-box reduction $\mathcal{R}(\cdot,\cdot)$ such that $\mathcal{R}(f, \mathcal{A}_H)$ is a second-preimage adversary against the compression function f that $(t + t', \alpha \cdot \varepsilon + \beta)$-breaks f, for all compression functions f and all second preimage adversaries $\mathcal{A}_H$ that (t, ε)-break H^f.*

The reduction is given *black-box* access to both the adversary and the compression function f, and this is a way of formalizing that the reduction must work for any adversary and any compression function. For the sake of simplicity, in this paper we allow the reduction to issue *only one query* to the adversary. To some extent, this narrows our study a little, but all the reductions we are aware of (in [3,17,18]) fit into this category. Note also that the adversary $\mathcal{A}_H$ may fail deterministically on a given challenge, so that it is pointless to re-run it again and again to increase its success probability.

In setting of our security theorem above, there are three parties: the challenger, the reduction and the adversary. To make the discussion simpler we will focus on the Sec security notion, but our reasoning extends to other notions. In the Sec game, the challenger sends the reduction a challenge made of an input x to f, and a "key" k for f. The reduction has to find a distinct input x' such that $f_k(x) = f_k(x')$. For this purpose, the reduction may use the $\mathcal{A}_H$ adversary: the reduction sends the adversary a challenge made of a message M of at most ℓ message blocks, and a key K. The adversary may either returns a message M' such that $H^f(K, M) = H^f(K, M')$ or fail. The precise sequence of interactions is the following:

$$\begin{array}{lll|l}
\text{Challenger} & \xrightarrow{x,k} & \text{Reduction} & & \\
 & & \text{Reduction} & \xrightarrow{M,K} \text{ Adversary} & \\
 & & \text{Reduction} & \xleftarrow{M'} \text{ Adversary} & M \neq M' \quad H_K(M) = H_K(M') \\
\text{Challenger} & \xleftarrow{x'} & \text{Reduction} & & x \neq x' \qquad f_k(x) = f_k(x')
\end{array}$$

If the compression function f is secure, then the "time/success probability" ratio of any adversary against f is greater than 2^n. The interest of the reductions

```
function f-SIMULATOR(x, k)
    if Log[x, k] = ⊥ then Log[x, k] ←$ {0,1}^n
    return Log[x, k]
end function
```

Fig. 2. A dummy random function simulator

is that given an adversary $\mathcal{A}_H$ against H^f, one must have: $(t+t')/(\alpha\cdot\varepsilon+\beta) \geq 2^n$, and therefore the "time/advantage" ratio of $\mathcal{A}_H$ is lower-bounded by:

$$\frac{t}{\varepsilon} \geq 2^n\alpha + \frac{2^n\beta - t'}{\varepsilon}. \tag{1}$$

The right-hand side of Eq. (1) is the *provable security level* that the reduction offers. Note that it bizarrely depends on the success probability of the adversary, but this seems unavoidable.

Reductions are generally assumed to have to simulate the legitimate input challenge distribution the adversary is normally expecting. In our case, this means that the distribution of the challenges M, K must be indistinguishable from random. Note that if M, K were biased, then the adversary could detect that it is "being used", and fail deterministically. In any case, when we mention the success probability ε of the adversary $\mathcal{A}_H$, we assume that its input distribution is uniformly random.

When considering a single run of the reduction, its success probability should depend very much on whether the adversary succeeds or not. Therefore, it makes sense to write:

$$\mathbb{P}\left[\mathcal{R} \text{ succeeds}\right] = \varepsilon\cdot\mathbb{P}\left[\mathcal{R} \text{ succeeds} \mid \mathcal{A}_H \text{ succeeds}\right] + (1-\varepsilon)\mathbb{P}\left[\mathcal{R} \text{ succeeds} \mid \mathcal{A}_H \text{ fails}\right]$$

This justifies why we assumed the success probability of the reduction to be of the form $\alpha \cdot \varepsilon + \beta$, and in fact we have:

$$\alpha = \mathbb{P}\left[\mathcal{R} \text{ succeeds} \mid \mathcal{A}_H \text{ succeeds}\right] - \mathbb{P}\left[\mathcal{R} \text{ succeeds} \mid \mathcal{A}_H \text{ fails}\right]$$
$$\beta = \mathbb{P}\left[\mathcal{R} \text{ succeeds} \mid \mathcal{A}_H \text{ fails}\right]$$

Now, while our objective is to understand what happens when $\mathcal{A}_H$ *succeeds*, it is easier to get a glimpse of what happens when $\mathcal{A}_H$ *fails*. In this setting, the reduction is just a randomized Turing machine trying to break the second preimage resistance of an arbitrary black-box function, which cannot be done faster than exhaustive search. For instance, f could be a Pseudo-Random Function with a randomly-chosen secret key. We could even use the algorithm shown in Fig. 2 to simulate a "truly" random function. In any case, it follows that $\beta \leq t'/2^n$. The provable security level offered by a reduction is thus upper-bounded by $\alpha \cdot 2^n$. We will thus say that a reduction is *useable* if $\alpha > t'/2^n$, as this implies that the reduction offers a provable security level better than that of exhaustive search (or equivalently, that the reduction actually makes use of the adversary).

5.1 How Do Reductions Use the Adversary?

In the sequel, we will make the natural assumption that the $\mathcal{A}_H$ adversary the reduction has access to has a non-zero success probability. We will also restrict our attention to useable reductions. By doing so we rule out modes of operation for which no useable reduction is known (such as the Merkle-Damgård construction), but at the same time we rule out bogus modes that would have been a problem.

Let us consider a provably secure mode of operation H also satisfying the hypotheses of Lemma 1. (i.e., injective, extractable and strengthened). We need to define yet another property to make our argument work. We say that a mode of operation is *suffix-clonable* if given an ℓ-block message M, a key K and an integer $0 < i \leq \ell$, and the sequence $h_0, \ldots, h_{\ell+1}$ of compression function outputs generated during the evaluation of $H^f(M, K)$, it is always possible to find a different ℓ-block message M' such that:

(i) $H^{(\cdot)}(K, M, i-1, h_{i-2}) \neq H^{(\cdot)}(K, M', i-1, h_{i-2})$
(ii) For all j such that $i \leq j \leq \ell$, $H^{(\cdot)}(K, M, j, h_{j-1}) = H^{(\cdot)}(K, M', j, h_{j-1})$

This is a bit technical, but is required by a part of the proof. Intuitively, it means that it is possible to find a message that would generate the same compression function inputs after the i-th iteration if a collision occurred, while generating a different input for the i-th iteration.

The Merkle-Damgård construction (and therefore HAIFA) and the split-padding are easily seen to be suffix clonable: it is sufficient to change the i-th message block while leaving all the subsequent messages blocks untouched. Shoup's construction is also easily seen to be suffix-clonable: it suffices to leave K untouched and to modify the beginning of M. Lastly, the BCM mode of operation is also suffix-clonable (and it again suffices to keep the right suffix of M).

We will thus assume that our mode of operation $H^{(\cdot)}$ is suffix-clonable. Since it is provably secure, there exists a reduction $\mathcal{R}$ with a reasonably high success probability. Our objective, and the main technical contribution of this section, is to show the following theorem:

Theorem 7. *We always have $\alpha \leq 1/\ell + t'/2^n$. It follows that the provable security level offered by $\mathcal{R}$ cannot be higher than $2^n/\ell + t'$.*

The remaining of this section is devoted to the proof of this result. The general idea of the proof is to build an environment around the reduction $\mathcal{R}$ that simulates a legitimate "world" for $\mathcal{R}$, but in which it is easy to see that $\mathcal{R}$ has a low success probability. Then because the security level offered by $\mathcal{R}$ has to hold in all legitimates environment, it follows that in general $\mathcal{R}$ cannot offer more in general than in the simulated world.

Connection Point. Before going any further, let us observe what happens when the adversary finds a second preimage. Let us denote by x_i and h_i (resp.

x'_i and h'_i) the sequence of inputs and outputs of f while evaluating $H^f(M)$ (resp. $H^f(M')$). Since M and M' collide, and because H satisfies the hypotheses of Lemma 1, then a second preimage of one of the x_i input values can be readily obtained from M'. If we look closely at the proof of Lemma 1, we will see that if $\left|M\right| \neq \left|M'\right|$, then we obtain a second preimage of f on the last invocation. Otherwise, there exists an index i such that $f(x_i) = f(x'_i)$ and $x_i \neq x'_i$. In the sequel, we call this particular index i the "connection point", and we note that at this particular index a second preimage of x_i for f is revealed, which we call "the second preimage at connection point".

Embedding. The strategy used by all the reductions we are aware of is to *embed* the small challenge (x, k) into the big challenge (M, K). Following our definition, we say that (x, k) is *embedded* into (M, K) at location i if and only if $f_k(x)$ is evaluated during the i-th iteration of the main loop of Algorithm 2 during the evaluation of $H^k(K, M)$. We will show that the second preimage returned by the adversary can only be used by the reduction if the second preimage at connection points directly gives a solution to the small challenge. Let us denote by ♣ the condition "the second preimage at connection point is a second preimage of the small challenge sent by the Challenger to $\mathcal{R}$". Formally, this means that:

$$\mathbb{P}\left[\clubsuit\right] = \mathbb{P}\left[\Big(\exists i.\ \ x_i = (x,k) \text{ in Algorithm 2}\Big) \wedge (x_i \neq x'_i) \wedge (h_i = h'_i)\right]$$

We can then write:

$$\begin{aligned}\mathbb{P}\left[\mathcal{R} \text{ succeeds} \mid \mathcal{A} \text{ succeeds}\right] = {} & \mathbb{P}\left[\mathcal{R} \text{ succeeds} \mid \mathcal{A} \text{ succeeds} \wedge \clubsuit\right] \cdot \mathbb{P}\left[\clubsuit\right] \\ + {} & \mathbb{P}\left[\mathcal{R} \text{ succeeds} \mid \mathcal{A} \text{ succeeds} \wedge \neg\clubsuit\right] \cdot \mathbb{P}\left[\neg\clubsuit\right] \quad (2)\end{aligned}$$

We first argue that the challenge cannot be embedded more than once. If the challenge were embedded twice or more, the input distribution of the adversary would not be random, because we would have $x_i = x_j$ for $i \neq j$ in Algorithm 2, something that is highly unlikely when M and K are drawn at random. This is not allowed in the first place, and the adversaries could straightforwardly detect it and abort.

Next, we claim that in order to be usable, a reduction *must* embed the challenge (x, k) into (M, K). This justifies *a posteriori* our observation that the three schemes of interest all allow some form of embedding. To establish this result, we first show that a legitimate world with various interesting properties can be built around the reduction. When we argued that β was small, we used the (somewhat informal) argument that f could be implemented by a Random Function simulator, and that inverting such a function faster than exhaustive search is impossible. We now make this argument more formal, with the additional feature that we will be able to choose whether the adversary succeeds or fails, and where it connects.

Simulation. The easy case is when we want $\mathcal{A}_H$ to fail, as it is sufficient to let f simulate an arbitrary random function, and let $\mathcal{A}_H$ return a random string, or fail explicitly. The more interesting case is when we want $\mathcal{A}_H$ to succeed. The difficulty comes from the fact that the view of the reduction must be consistent: after having received M' from the $\mathcal{A}_H$, the reduction must be able to check that $H_K^f(M) = H_K^f(M')$ by querying f. This is in fact quite easy to achieve, by *programming* the function f. We thus simulate a complete environment around the execution with the following procedure:

1. Before $\mathcal{R}$ sends its query (M, K) to $\mathcal{A}_H$, we simulate f by generating random answers and storing them (for consistency), "implementing" f with the random function simulator of Fig. 2.
2. When $\mathcal{R}$ sends its query (M, K) to $\mathcal{A}_H$, we choose an integer $i \in \{0, \ldots, \ell\}$ (this will be the connection point), and we use the suffix-clonability property of the mode of operation to generate a different message $M' \neq M$ satisfying the conditions of the definition of suffix-clonability.
3. We evaluate $H^f(M')$ in a special way. On the first $i-1$ iterations we use the random function simulator in place of f. On the i-th iteration we program f so that $f(x'_i) = h_i$, thus "connecting" M' to M in iteration i.
4. We return M' as the answer of $\mathcal{A}_H$ to the reduction, and keep simulating f. The reduction will be able to check that $H_K^f(M) = H_K^f(M')$ by sending the appropriate queries to f.

When running inside this environment, the view of the reduction is consistent and legitimate. In this environment, we are able to choose the connection point at will. For instance, we can make sure that the ♣ event never happens. In this case, the reduction, even though it knows a collision on f, cannot find a second preimage on f faster than exhaustive search (because each new query to f returns an independent random answer, and thus each query yields a second preimage with probability 2^{-n}).

It follows if a reduction does not embed its challenge, then it cannot be usable. We conclude that a usable reduction must embed its challenge exactly once with non-zero probability. As a matter of fact, the reductions of the three schemes considered in the introduction published in the literature embed their challenge with probability one. Equation (2) then gives:

$$\mathbb{P}\left[\mathcal{R} \text{ succeeds} \mid \mathcal{A} \text{ succeeds}\right] \leq \mathbb{P}\left[\mathcal{R} \text{ succeeds} \mid \mathcal{A} \text{ succeeds} \wedge \clubsuit\right] \cdot \mathbb{P}\left[\clubsuit\right] + \frac{t'}{2^n} \quad (3)$$

Now, to prove Theorem 7, we upper-bound the probability that the ♣ condition occurs. The reduction cannot control "where" the adversary will "connect" to the big challenge M. Conversely, if the adversary could guess where the challenge is embedded, then she could systematically refuse to connect precisely there. In fact, we need not even worry about this complication, since the adversary can foil all the reduction's plan by connecting randomly. In our simulation procedure, if we choose the connection point uniformly at random between 0 and ℓ, then the ♣ event only happens with probability $1/\ell$. Combining this with

Eq. (3) yields:

$$\mathbb{P}\left[\mathcal{R} \text{ succeeds} \mid \mathcal{A}_H \text{ succeeds}\right] \leq \frac{1}{\ell} + \frac{t'}{2^n}$$

And this is exactly what we needed to complete the proof of Theorem 7. We conclude by pondering on this intriguing situation, where some narrow-pipe modes of operations are provably resistant to generic second preimage attacks, yet this cannot be shown in the standard model.

References

1. Andreeva, E., Bouillaguet, C., Dunkelman, O., Kelsey, J.: Herding, second preimage and trojan message attacks beyond Merkle-Damgård. In: Jacobson Jr, M.J., Rijmen, V., Safavi-Naini, R. (eds.) SAC 2009. LNCS, vol. 5867, pp. 393–414. Springer, Heidelberg (2009)
2. Andreeva, E., Bouillaguet, C., Fouque, P.-A., Hoch, J.J., Kelsey, J., Shamir, A., Zimmer, S.: Second preimage attacks on dithered hash functions. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 270–288. Springer, Heidelberg (2008)
3. Andreeva, E., Preneel, B.: A three-property-secure hash function. In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 228–244. Springer, Heidelberg (2009)
4. Bellare, M., Rogaway, P.: Collision-resistant hashing: towards making UOWHFs practical. In: Kaliski Jr, B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 470–484. Springer, Heidelberg (1997)
5. Brassard, G. (ed.): CRYPTO 1989. LNCS, vol. 435. Springer, Heidelberg (1990)
6. Damgård, I.B.: A design principle for hash functions. In: Brassard [5], pp. 416–427
7. Dean, R.D.: Formal aspects of mobile code security. Ph.D. thesis, Princeton University, Jan 1999
8. Eli Biham, O.D.: A framework for iterative hash functions – HAIFA. Presented at the second NIST hash workshop, 24–25 Aug 2006
9. Impagliazzo, Russell, Naor, Moni: Efficient cryptographic schemes provably as secure as subset sum. J. Cryptol. **9**(4), 199–216 (1996)
10. Joux, A.: Multicollisions in iterated hash functions. Application to cascaded constructions. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 306–316. Springer, Heidelberg (2004)
11. Kelsey, J., Kohno, T.: Herding hash functions and the Nostradamus attack. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 183–200. Springer, Heidelberg (2006)
12. Kelsey, J., Schneier, B.: Second preimages on n-bit hash functions for much less than 2^n work. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 474–490. Springer, Heidelberg (2005)
13. Lucks, S.: A failure-friendly design principle for hash functions. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 474–494. Springer, Heidelberg (2005)
14. Merkle, R.C.: One way hash functions and DES. In: Brassard [5], pp. 428–446
15. Naor, M., Yung, M.: Universal one-way hash functions and their cryptographic applications. In: STOC, pp. 33–43. ACM (1989)
16. Rogaway, P., Shrimpton, T.: Cryptographic hash-function basics: definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 371–388. Springer, Heidelberg (2004)

17. Shoup, V.: A composition theorem for universal one-way hash functions. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 445–452. Springer, Heidelberg (2000)
18. Yasuda, K.: How to fill up Merkle-Damgård hash functions. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 272–289. Springer, Heidelberg (2008)