Revisiting Security Relations Between Signature Schemes and their Inner Hash Functions

Article · January 2007

CITATIONS 0
READS 25

10 authors, including:

Emmanuel Bresson
64 PUBLICATIONS 2,823 CITATIONS
SEE PROFILE

Pierre-Alain Fouque
Ecole Normale Supérieure de Paris
125 PUBLICATIONS 3,192 CITATIONS
SEE PROFILE

Louis Goubin
Université de Versailles Saint-Quentin
94 PUBLICATIONS 3,089 CITATIONS
SEE PROFILE

Aline Gouget
Gemalto
36 PUBLICATIONS 480 CITATIONS
SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Hash functions View project
White box crypto View project

All content following this page was uploaded by Louis Goubin on 23 August 2015.
The user has requested enhancement of the downloaded file.
Revisiting Security Relations Between Signature Schemes and their Inner Hash Functions

Emmanuel Bresson\(^3\) and Benoît Chevallier-Mames\(^1\) and Christophe Clavier\(^1\) and Blandine Debraize\(^4\) and Pierre-Alain Fouque\(^3\) and Louis Goubin\(^1\) and Aline Gouget\(^1\) and Gaetan Leurent\(^4\) and Phong Nguyen\(^4\) and Pascal Paillier\(^1\) and Thomas Peyrin\(^2\) and Sébastien Zimmer\(^4\)

\(^1\) Cryptography and Innovation, Gemalto Security Labs
\(^2\) France Telecom Division R&D
\(^3\) DCSSI
\(^4\) Ecole Normale Supérieure

Abstract. After years of almost full confidence in the security of common hash functions such as MD5 and SHA-1, the cryptographic community is now facing the unprecedented threat of seeing practical security applications succumb to concrete attacks. A way to cope with this crisis is to fasten the development of new hash functions, but another crucial task is to assess the implications these attacks on hash functions may have on cryptographic systems. This paper reports a thorough investigation on how recent attacks on hash functions impact the security of signature schemes. We suggest the notion of probabilistic hash-and-sign signatures and further classify signature schemes into various related categories which allow us to identify completely the nature of security relations between signature schemes and their inner hash functions. We also determine how using iterated hash functions a la Merkle-Damgård impacts the security of deterministic (resp. probabilistic) hash-and-sign signatures. We confirm that the security gain inherent to using the probabilistic hash-and-sign paradigm may be lost completely if instantiated with a Merkle-Damgård hash function and unwise operating mode.

1 Introduction

Undoubtedly, hash functions constitute an essential component of all sorts of systems and constructions in cryptography, should these stand in the public-key or secret-key paradigm. From basic primitives (encryption, signatures, commitments, etc) to advanced protocols (fair exchange, ecash, multiparty computations and so forth), they have spread all over the place, appreciated for the many and very different properties one expects these functions to fulfill. In [32] more than 50

Since the early days of modern cryptography however, the design of hash functions has remained mostly heuristic and for a large part based on ideas and concepts arising from the design of block ciphers. Even if most hash functions were conceived with little more than a paper and a pencil, a large confidence in the security of some hash functions such as SHA-1 has long been shared in the cryptographic community. The very few concrete attacks on trusted hash functions that came to public attention for more than a decade definitely played a role in reinforcing this belief.

Recently however, initiated by the works of Chabaud and Joux [10] and later Biham and Chen [4], powerful attack techniques were suddenly discovered and reported by Wang [38]. Applicable to a wide range of hash functions and leading sometimes to unexpectedly low workfactors, new attacks are now progressively emerging which collect and assemble techniques of independent nature [39, 40, 15, 25, 28]. It is now clear that hash functions designs cannot be trusted without adequate public scrutiny and the confidence is low that commonly used hash functions such as the SHS standard SHA-1 can withstand cryptanalytic efforts for ever [16].

**How broken hash functions impact cryptosystems.** Because of the systematic appearance of hash functions in cryptographic constructions, the security of a large variety of systems is now at risk [21, 37]. However the role played by hash functions in the overall security of a system is, in many constructions, so badly understood that it is not obvious at all to see how that system suffers from being based on broken hash functions. Does the security of the OAEP padding (used in conjunction with a trapdoor permutation to yield random-oracle secure encryption) vanish when a collision on one of the inner hash functions is discovered?

The goal of our work is to explore the interplay between these two fundamental concerns: the security of a cryptographic system \(S = S[H_1, \ldots, H_n]\) based on hash functions \(H_1, \ldots, H_n\) and the security of \(H_1, \ldots, H_n\).
Assuming for instance that a scheme $S$ involves a unique hash function $H$, we want to determine how the security of $H$ relates to the one of $S$. This amounts to computationally connect security notions for $S$ with security notions for $H$. Our approach is to exhaust all these connections (at least all the ones we can see) which we categorize into the four following types:

1. an attack, which we define as a polynomial reduction $\text{Break}(H) \geq \text{Break}(S)$ for well-defined security notions\(^1\). In this case, our reduction makes explicit how an attack of a given type on the hash function is enough to break the scheme in a prescribed way;
2. a security proof that is, a polynomial reduction $\text{Break}(H) \leq \text{Break}(S)$. Here, the reduction tells there is no attack of a certain type on $S$ unless one finds a particular weakness in $H$;
3. an impossibility of attack, namely a means to show that there is no reduction $\text{Break}(H) \geq \text{Break}(S)$. This is usually done based on a meta-reduction\(^2\): if $\text{Break}(H) \geq_{P} \text{Break}(S)$ then $\mathcal{R} \geq_{\mathcal{M}} P$ where $P$ is an auxiliary computational problem (hence this technique requires an extra assumption);
4. an impossibility of security proof i.e. evidence that a polynomial reduction $\text{Break}(H) \leq \text{Break}(S)$ does not exist (using the meta-reduction technique as well).

We view 2. and 3. as positive security results and 1. and 4. as negative security results.

Our contributions. This paper only deals with signature schemes. The overall goal of our work is to report as clearly as possible the implications of the current crisis in the field of hash functions and identify the new threats signature schemes are facing in practice. Applying the approach described above, we provide a number of (reductionist) relations between the security profile\(^2\) of a signature scheme $S=S[H]$ and the security profile of the hash function $H$ (we also suggest convenient security notions for hash function families on our way). We find that these relations heavily depend on the way $S$ makes use of $H$. We therefore classify signature schemes into different categories: deterministic hash-and-sign signatures and probabilistic hash-and-sign signatures. We also introduce the properties of primitiveness and injectivity which enlarge the scope of certain security relations. Our classification remains as general as possible while capturing the case of signature schemes of common practice such as RSA-PSS or ECDSA. The security relations we expose in this paper confirm that all signature schemes are not implicated on the same level by the recent attacks on hash functions.

We also determine how using iterated hash functions a la Merkle-Damgård impacts the security of deterministic (resp. probabilistic) hash-and-sign signatures. Our motivation here is to identify more specific results in the case of functions such as MD5 and SHA-1. We confirm that the security gain inherent to using the probabilistic hash-and-sign paradigm may be lost completely if instantiated with a Merkle-Damgård hash function and unwise operating mode. We finally give concrete attack workloads for attacking popular operating modes used in practical implementations of signature schemes.

Roadmap. The next section provides a number of preliminary facts on security reductions. We remind definitions for hash functions and adopt well-defined security notions in Section 3. In Section 4, we define the deterministic versus probabilistic hash-and-sign paradigms and the notions of primitive and injective schemes. We then review and classify a number of signature schemes based on these four basic concepts. Sections 5 and 6 present the security relations between a hash-and-sign signature scheme and its inner hash function in the deterministic and probabilistic cases respectively. Section 7 explores the particular case of Merkle-Damgård hash functions as more specific results can be stated in this context. We finally conclude with recommendations on the use of hash functions in signature schemes.

2 Preliminary Notions of Provable Security

2.1 Black-Box Reductions

We adopt the concrete-security setting [], as opposed to the asymptotic one. Given a computational problem $P$, a probabilistic algorithm is said to $(\tau, \varepsilon)$-solve\(^3\) $P$ when it halts after at most $\tau$ elementary steps and outputs a

\(^1\) More details on reductions are found in the next section.

\(^2\) The hierarchy of all the security levels related to a scheme.

\(^3\) We may say $(\tau, \varepsilon)$-break $P$ when $P$ is a security related problem.
solution of $P$ with success probability at least $\varepsilon$. The execution time $\tau$ relates to some fixed model of computation. $\text{Succ}(P, \tau)$ denotes the maximal success probability taken over all probabilistic algorithms solving $P$ in no more than $\tau$ elementary steps. A concrete black-box\(^4\) reduction $R$ between two computational problems $P_1$ and $P_2$ is a probabilistic algorithm $R$ which $(\tau_1, \varepsilon_1)$-solves $P_1$ given black-box access to an oracle $(\tau_2, \varepsilon_2)$-solving $P_2$. We write $P_1 \leq_R P_2$ when $R$ is known to reduce $P_1$ to $P_2$ with polynomial reduction cost, meaning that $\tau_1 = \text{poly}(\tau_2)$ and $\varepsilon_1 = \text{poly}(\varepsilon_2)$ where the polynomials may depend on additional reduction parameters. Note that $R$ can be polynomial even when no algorithm $(\tau_2, \varepsilon_2)$-solving $P_2$ is known to exist. $P_1 \leq P_2$ states that a polynomial $R$ exists such that $P_1 \leq_R P_2$ and $P_1 \equiv P_2$ means $P_1 \leq P_2$ and $P_1 \geq P_2$. We write $P_1 \leq_R P_2 \land P_3 \land \cdots \land P_n$ when $R$ solves $P_1$ given oracle access to solvers for $P_2, \ldots, P_n$ and $\tau_1 = \text{poly}(\tau_2, \ldots, \tau_n)$, $\varepsilon_1 = \text{poly}(\varepsilon_2, \ldots, \varepsilon_n)$. Relation $\leq$ is a transitive ordering among computational problems. Given a function $f$ with $t$ inputs, $\text{Time}(f, \ell_1, \ldots, \ell_t)$ stands for the minimum, taken over all algorithms $A_f$ computing $f$, of the worst case running time of $A_f$ for inputs of respective bitlength $\ell_1, \ldots, \ell_t$.

### 2.2 Constructive Security Reductions

Let $P_1, P_2$ be two computational problems. We defined earlier the predicate $P_1 \leq P_2$ as $(\exists R : P_1 \leq_R P_2)$ where we recall that $P_1 \leq_R P_2$ implies that $R$ is polynomial. A provable-security statement $P_1 \leq P_2$ (where $P_1$ and $P_2$ are now security-related problems) therefore indicates that if an efficient solver for $P_2$ exists then an efficient solver for $P_1$ exists as well. Such a statement is meaningful when there are reasons to believe that $P_1$ cannot be solved efficiently. When efficient solvers for $P_1$ are known to exist, however, the security statement vacuously holds. So is the statement $P_1 \leq P_2$ of any interest in this case?

Well, $P_1 \leq P_2$ is still meaningful if a) such an efficient solver for $P_1$ exists but is unknown and seems hard to construct, b) the proof of $P_1 \leq P_2$ is constructive meaning that an explicit black-box reduction $R$ is given such that $P_1 \leq_R P_2$. The statement can then be reinterpreted by saying that if one had access to an effective solver for $P_2$ then an effective solver for $P_1$ would be found. $P_1$ then remains unsolvable under the assumption that an effective solver for $P_1$ is hard to construct under current human knowledge. Note that this formulation is stronger than the previous, existential formulation. It implies for instance that $R$ can be used as a compiler which, given the source code of a program solving $P_2$, can be used to build a program solving $P_1$. We refer to [33] for more detail. From now on, $P_1 \not\equiv P_2$ states that we know a constructive proof for $P_1 \leq P_2$, meaning $(\exists R : P_1 \leq_R P_2)$ and such an $R$ is explicitly given by its pseudo-code in the proof. Constructive polynomial equivalence is noted $P_1 \equiv P_2$.

### 2.3 The Random Oracle Model

The random oracle (RO) model was implicitly introduced by Fiat and Shamir [18] and later formalized by Bellare and Rogaway [1]. It has been extensively used since. By opposition to using a fully detailed specification of a cryptosystem (its standard-model description), the RO model abstracts away the description of certain components such as hash functions or pseudo-random generators by making them external to the cryptosystem under the form of oracles. Generating or verifying a signature (and possibly generating keys as well) then requires oracle calls. It is striking to observe that (with a few exceptions) the intensive use of the RO model has rendered innocuous the use of hash functions in cryptographic designs. In the RO model, studying the security relations between a construction and its inner hash functions is completely vacuous. We therefore stick to the standard model in our investigations.

### 3 Hash Functions and Related Security Notions

We first give a number of definitions related to hash functions. We clarify that hash functions (as per our definitions) only handle bitstrings and must be composed with other specific formatting functions to form integer arrays, group elements, etc. The following formal notions of a hash function, a compression function and an iterated hash function are similar to those given by Black, Rogaway and Shrimpton in [5].

**Definition 1 (Hash function).** A function $H$ is a hash function if it maps $\{0, 1\}^*$ (the set of finite bitstrings) to $\{0, 1\}^m$ (the set of $m$-bit strings) for some integer $m > 0$ called the output size of $H$.

\(^4\) All reductions considered in this paper are concrete and fully black-box.
Definition 2 (Compression function). A compression function is a function \( f : \{0,1\}^m \times \{0,1\}^b \rightarrow \{0,1\}^m \) where \( m, b \) are integers such that \( m > 0 \) and \( b > 0 \).

Definition 3 (Iterated hash function). Let \( f : \{0,1\}^m \times \{0,1\}^b \rightarrow \{0,1\}^m \) be a compression function. Given an arbitrary function \( g : \{0,1\}^* \rightarrow \{\{0,1\}^b\}^* \) and \( IV_0 \in \{0,1\}^m \), the iterated hash function \( H \) constructed from \( (f,g,IV_0) \) is the hash function defined for \( M \in \{0,1\}^* \) by \( H(M) = h_t \) where \( X_1||...||X_t = g(M), h_0 = IV_0 \) and \( h_i = f(h_{i-1},X_i) \) for \( i \in [1..t] \). We call \( b \) the block size of \( H \).

3.1 Security Notions for Hash Functions

The definitions for (second) preimage-resistance and for collision-resistance were given by Brown in [9] and appear under various forms in [29]. These definitions are somewhat simplified here because we only deal with fixed hash functions rather than families of hash functions. Subsequently, the related security assumptions, most particularly the assumption about collision-resistance, are stronger than the usual assumptions for hash functions families.

Definition 4 (Preimage-resistance \( \text{PRE}^n[H] \)). Let \( H : \{0,1\}^* \rightarrow \{0,1\}^m \) be a hash function. The preimage problem \( \text{PRE}^n[H] \) consists, given a random \( m \leftarrow \{0,1\}^m \) in finding an \( n \)-bit string \( M \) such that \( H(M) = m \). Formally,

\[
\text{Succ} (\text{PRE}^n[H], \tau) = \max_{A_H} \text{Pr} [m \leftarrow \{0,1\}^m; M \leftarrow A_H(m) : (M \in \{0,1\}^n) \land (H(M) = h)]
\]

where the probability is taken over the random choices of \( m \) and \( A_H \) and the maximum is taken over all \( \tau \)-time probabilistic algorithms \( A_H \). \( H \) is said to be \((\tau, \varepsilon)\)-preimage-resistant on \( \{0,1\}^n \) if \( \text{Succ} (\text{PRE}^n[H], \tau) < \varepsilon \).

Definition 5 (Second-preimage-resistance \( \text{SEC}^n_1[H] \)). The second preimage problem \( \text{SEC}^n_1[H] \) consists, given a random \( M_1 \leftarrow \{0,1\}^n_1 \) in finding \( M_2 \in \{0,1\}^n_2 \) such that \( H(M_2) = H(M_1) \) and \( M_2 \neq M_1 \). Let us pose

\[
\text{Succ} (\text{SEC}^n_1[H], \tau) = \max_{A_H} \text{Pr} [M_1 \leftarrow \{0,1\}^n_1; M_2 \leftarrow A_H(M_1) : (M_2 \in \{0,1\}^n_2 \setminus \{M_1\}) \land (H(M_2) = H(M_1))]
\]

where the probability is taken over the random choices of \( M_1 \) and \( A_H \) and the maximum is taken over all \( \tau \)-time adversaries \( A_H \). \( H \) is said to be \((\tau, \varepsilon)\)-second-preimage-resistant with respect to \( (n_1, n_2) \) if \( \text{Succ} (\text{SEC}^n_1[H], \tau) < \varepsilon \).

For clarity, all security problems parameterized by the bitsize of their inputs or outputs are written with input sizes as subscripts and outputs sizes as superscripts.

Definition 6 (Collision-resistance \( \text{COL}^{n_1,n_2}[H] \)). Solving the collision-finding problem \( \text{COL}^{n_1,n_2}[H] \) consists in finding \( M_1 \in \{0,1\}^{n_1} \) and \( M_2 \in \{0,1\}^{n_2} \) such that \( M_1 \neq M_2 \) and \( H(M_1) = H(M_2) \). We then call \((M_1,M_2)\) an \((n_1,n_2)\)-collision. For any probabilistic algorithm \( A_H \), let

\[
\text{Succ}^{\text{COL}^{n_1,n_2}[H]} (A_H) = \text{Pr} [(M_1, M_2) \leftarrow A_H(\cdot) : (M_1 \in \{0,1\}^{n_1}) \land (M_2 \in \{0,1\}^{n_2} \setminus \{M_1\}) \land (H(M_2) = H(M_1))]
\]

where the probability is taken over the random choices of \( A_H \). We say that \( A_H \) \((\tau, \varepsilon)\)-solves \( \text{COL}^{n_1,n_2}[H] \) if \( A_H \) runs in time at most \( \tau \) and \( \text{Succ}^{\text{COL}^{n_1,n_2}[H]} (A_H) \geq \varepsilon \).

It is well-known that defining \((\tau, \varepsilon)\)-collision-resistance for (unkeyed) hash functions using an existential formulation i.e. by saying that

\[
\text{Succ} (\text{COL}^{n_1,n_2}[H], \tau) = \max_{A_H} \text{Succ}^{\text{COL}^{n_1,n_2}[H]} (A_H) < \varepsilon \tag{1}
\]

is futile [33]. Indeed either \( H \) admits no \((n_1,n_2)\)-collision in which case \( \text{Succ}^{\text{COL}^{n_1,n_2}[H]} (A_H) = 0 \) for any \( A_H \) (and \( \text{COL}^{n_1,n_2}[H] \) is trivially unsolvable), or there always exists \( A_H \) which simply outputs a fixed "hardwired" \((n_1,n_2)\)-collision \((M_1,M_2)\). So in this case \( \text{Succ} (\text{COL}^{n_1,n_2}[H], \tau) \) as defined above equals 1 as soon as \( \tau \) exceeds the time needed to print these two strings. We therefore adopt the human ignorance (or explicit reduction) approach of [33] which amounts to state security reductions in terms of knowledge. We elaborate on this in Section 2.2.

\( H \) is said \((\tau, \varepsilon)\)-collision-resistant with respect to \((n_1,n_2)\)-collisions if no explicit \( \tau \)-time adversary \( A_H \) is known such that \( \text{Succ}^{\text{COL}^{n_1,n_2}[H]} (A_H) \geq \varepsilon \). Alternately, one can rely on (1) by restricting the maximum to adversaries known to man. Note that by the birthday paradox, we know a collision-finding algorithm \( A_H \) with success probability \( \varepsilon = 1/2 \) and running time \( \tau \approx 2^{m/2} \). So no hash function \( H : \{0,1\}^* \rightarrow \{0,1\}^m \) is \((2^{m/2}, 2^{-1})\)-collision-resistant for general \( n_1,n_2 \)’s.
Lemma 7. Let $H : \{0, 1\}^* \to \{0, 1\}^m$ be a hash function. Then for any $n_1, n_2 > 0,$

$$\text{COL}^{n_1, n_2} [H] \iff \text{SEC}^{n_1}_1 [H], \text{SEC}^{n_2}_2 [H] \text{ and } \text{SEC}^{n_1}_1 [H] \iff \text{PRE}^{n_2} [H].$$

Proof. $(\text{COL}^{n_1, n_2} [H] \iff \text{SEC}^{n_2}_1)$. Assuming a $(\tau_2, \varepsilon_2)$-solver $A_2$ for $\text{SEC}^{n_2}_1 [H]$, one builds a reduction algorithm $A_1$ which $(\tau_1, \varepsilon_1)$-solves $\text{COL}^{n_1, n_2} [H]$ with $\varepsilon_1 = \varepsilon_2$ and $\tau_1 = \tau_2 + \text{Time}(\text{rand}, n_1)$ as follows. On no input, $A_1$ uniformly selects $M_1 \leftarrow \{0, 1\}^{n_1}$, runs $M_2 \leftarrow A_2(M_1)$ and returns $(M_1, M_2)$. A proof that $\text{SEC}^{n_2}_1 [H] \iff \text{PRE}^{n_2} [H]$ is given for instance in [34]. To yield an efficient reduction, however, it is required that $n_1, n_2 \gg m$. □

3.2 Hash Function Families and Related Security Notions

We recall the definition of a hash function family.

**Definition 8 (Hash function family).** A hash function family $F$ is a function $F : \{0, 1\}^* \times \{0, 1\}^\tau \to \{0, 1\}^m$ for integers $m, \tau > 0$ such that for any $r \in \{0, 1\}^\tau$, $F(\cdot, r)$ is a hash function of output size $m$.

The second argument $r$ in $F(M, r)$ is called the index (or key) of $F$. We define a collection of security notions with respect to a hash function family $F$ with index space $\{0, 1\}^\tau$. As for unkeyed hash functions, one essentially faces notions of preimage, second preimage and collision resistance. These security notions are defined in the concrete setting using $(\tau, \varepsilon)$-adversaries.

**Definition 9 (Existential Preimage Resistance E-PRE$^n [F]$).** Breaking E-PRE$^n [F]$ consists, given a random $m \leftarrow \{0, 1\}^m$ in finding a pair $(M, r) \in \{0, 1\}^m \times \{0, 1\}^\tau$ such that $F(M, r) = m$. Formally,

$$\text{Succ}(\text{E-PRE}^n [F], \tau) = \max_{A_F} \Pr [m \leftarrow \{0, 1\}^m, (M, r) \leftarrow A_F(m) : (|M| = n) \wedge (|r| = \tau) \wedge (F(M, r) = m)],$$

where the probability is taken over the random choices of $m$ and $A_F$ and the maximum is taken over all $\tau$-time probabilistic algorithms $A_F$. $F$ is said to be $(\tau, \varepsilon)$-existentially preimage-resistant on $\{0, 1\}^n$ if $\text{Succ}(\text{E-PRE}^n [F], \tau) < \varepsilon$.

**Definition 10 (Universal Preimage Resistance U-PRE$^n [F]$).** Breaking U-PRE$^n [F]$ consists, given random $m \leftarrow \{0, 1\}^m$ and $r \leftarrow \{0, 1\}^\tau$, in finding an $n$-bit string $M$ such that $F(M, r) = m$. We define

$$\text{Succ}(\text{U-PRE}^n [F], \tau) = \max_{A_F} \Pr [m \leftarrow \{0, 1\}^m, r \leftarrow \{0, 1\}^\tau, M \leftarrow A_F(m, r) : (|M| = n) \wedge (F(M, r) = m)],$$

the probability being taken over the random choices of $m, r$ and $A_F$ and the maximum over $\tau$-time probabilistic algorithms $A_F$. $F$ is said to be $(\tau, \varepsilon)$-universally preimage-resistant on $\{0, 1\}^n$ if $\text{Succ}(\text{U-PRE}^n [F], \tau) < \varepsilon$.

**Definition 11 (Existential Second Preimage Resistance E-SEC$^{n_2}_{n_1} [F]$).** Breaking E-SEC$^{n_2}_{n_1} [F]$ consists, given a random $M_1 \leftarrow \{0, 1\}^{n_1}$ in finding $(M_2, r) \in \{0, 1\}^{n_2} \times \{0, 1\}^\tau$ such that $F(M_2, r) = F(M_1, r)$ and $M_2 \neq M_1$. Let

$$\text{Succ}(\text{E-SEC}^{n_2}_{n_1} [F], \tau) = \max_{A_F} \Pr \left[ M_1 \leftarrow \{0, 1\}^{n_1}, (M_2, r) \leftarrow A_F(M_1) : (|M_2| = n_2) \wedge (|r| = \tau) \wedge (M_2 \neq M_1) \wedge (F(M_2, r) = F(M_1, r)) \right],$$

the probability being taken over the random choices of $M_1$ and $A_F$ and the maximum over all $\tau$-time $A_F$’s. $F$ is said to be $(\tau, \varepsilon)$-existentially-second-preimage-resistant with respect to $(n_1, n_2)$ if $\text{Succ}(\text{E-SEC}^{n_2}_{n_1} [F], \tau) < \varepsilon$.

**Definition 12 (Universal Second Preimage Resistance U-SEC$^{n_2}_{n_1} [F]$).** Given a random pair $(M_1, r) \leftarrow \{0, 1\}^{n_1} \times \{0, 1\}^\tau$, find $M_2 \in \{0, 1\}^{n_2} \setminus \{M_1\}$ such that $F(M_2, r) = F(M_1, r)$ and $M_2 \neq M_1$:

$$\text{Succ}(\text{U-SEC}^{n_2}_{n_1} [F], \tau) = \max_{A_F} \Pr \left[ M_1 \leftarrow \{0, 1\}^{n_1}, r \leftarrow \{0, 1\}^\tau, M_2 \leftarrow A_F(M_1, r) : (|M_2| = n_2) \wedge (M_2 \neq M_1) \wedge (F(M_2, r) = F(M_1, r)) \right],$$

the probability being taken over the random choices of $(M_1, r)$ and $A_F$ and the maximum over all $\tau$-time $A_F$’s. $F$ is said to be $(\tau, \varepsilon)$-universally-second-preimage-resistant with respect to $(n_1, n_2)$ if $\text{Succ}(\text{U-SEC}^{n_2}_{n_1} [F], \tau) < \varepsilon$. 

5
Definition 13 (Absolute Second Preimage Resistance $A$-$\text{SEC}^{|n_2|}_{n_1}[F]$). Given a random $M_1 \leftarrow \{0,1\}^{n_1}$, find $M_2 \in \{0,1\}^{n_2}$ such that $F(M_2,r) = F(M_1,r)$ for any $r \in \{0,1\}^r$:

$$\text{Succ} \left( A-\text{SEC}^{|n_2|}_{n_1}[F], \tau \right) = \max_{A_F} \Pr \left[ M_1 \leftarrow \{0,1\}^{n_1}, M_2 \leftarrow A_F(M_1) : \left( |M_2| = n_2 \right) \land \left( M_2 \neq M_1 \right) \land \left( F(M_2,r) \neq F(M_1,r) \right) \right],$$

where the probability is taken over the random choices of $A_F$ and the maximum over all $\tau$-time $A_F$'s. $F$ is said to be $(\tau, \varepsilon)$-absolutely-second-preimage-resistant with respect to $(n_1, n_2)$ if $\text{Succ} \left( A-\text{SEC}^{|n_2|}_{n_1}[F], \tau \right) < \varepsilon$.

Existential, universal and absolute collision-resistance are defined along the same lines:

Definition 14 (Existential Collision-resistance $E$-$\text{COL}^{|n_1,n_2|}[F]$). Solving $E$-$\text{COL}^{|n_1,n_2|}[F]$ consists in finding $M_1 \in \{0,1\}^{n_1}$, $M_2 \in \{0,1\}^{n_2}$ and $r \in \{0,1\}^r$ such that $F(M_1,r) \neq F(M_2,r)$ for any $r \in \{0,1\}^r$. We then call $(M_1, M_2, r)$ an $(n_1, n_2)$-collision on $F$. For any probabilistic algorithm $A_F$, let

$$\text{Succ} \left( E-\text{COL}^{|n_1,n_2|}[F], \tau \right) = \max_{A_F} \Pr \left[ (M_1, M_2, r) \leftarrow A_F() : \left( |M_1| = n_1 \right) \land \left( |M_2| = n_2 \right) \land \left( M_2 \neq M_1 \right) \land \left( F(M_2,r) = F(M_1,r) \right) \right],$$

where the probability is taken over the random choices of $A_F$ and the maximum over all $\tau$-time $A_F$'s. $F$ is said to be $(\tau, \varepsilon)$-existentially-collision-resistant with respect to $(n_1, n_2)$ if $\text{Succ} \left( E-\text{COL}^{|n_1,n_2|}[F], \tau \right) < \varepsilon$.

Definition 15 (Universal Collision-resistance $U$-$\text{COL}^{|n_1,n_2|}[F]$). Breaking $U$-$\text{COL}^{|n_1,n_2|}[F]$ consists, given a random $r \leftarrow \{0,1\}^r$, in finding $M_1 \in \{0,1\}^{n_1}$ and $M_2 \in \{0,1\}^{n_2}$ such that $M_1 \neq M_2$ and $F(M_2,r) = F(M_1,r)$. For any probabilistic algorithm $A_F$, let

$$\text{Succ} \left( U-\text{COL}^{|n_1,n_2|}[F], \tau \right) = \max_{A_F} \Pr \left[ r \leftarrow \{0,1\}^r, (M_1, M_2) \leftarrow A_F(r) : \left( |M_1| = n_1 \right) \land \left( |M_2| = n_2 \right) \land \left( M_2 \neq M_1 \right) \land \left( F(M_2,r) = F(M_1,r) \right) \right],$$

where the probability is taken over the random choices of $A_F$ and the maximum over all $\tau$-time $A_F$'s. $F$ is said to be $(\tau, \varepsilon)$-universally-collision-resistant with respect to $(n_1, n_2)$ if $\text{Succ} \left( U-\text{COL}^{|n_1,n_2|}[F], \tau \right) < \varepsilon$.

Definition 16 (Absolute Collision-resistance $A$-$\text{COL}^{|n_1,n_2|}[F]$). Solving $A$-$\text{COL}^{|n_1,n_2|}[F]$ consists in finding $M_1 \in \{0,1\}^{n_1}$ and $M_2 \in \{0,1\}^{n_2}$ such that $F(M_1,r) = F(M_2,r)$ for any $r \in \{0,1\}^r$. We then call $(M_1, M_2)$ an $(n_1, n_2)$-absolute collision on $F$. For any probabilistic algorithm $A_F$, let us define

$$\text{Succ} \left( A-\text{COL}^{|n_1,n_2|}[F], \tau \right) = \max_{A_F} \Pr \left[ (M_1, M_2) \leftarrow A_F() : \left( |M_1| = n_1 \right) \land \left( |M_2| = n_2 \right) \land \left( M_2 \neq M_1 \right) \land \left( F(M_2,r) \neq F(M_1,r) \right) \right],$$

where the probability is taken over the random choices of $A_F$ and the maximum over all $\tau$-time $A_F$'s. $F$ is said to be $(\tau, \varepsilon)$-absolutely-collision-resistant with respect to $(n_1, n_2)$ if $\text{Succ} \left( A-\text{COL}^{|n_1,n_2|}[F], \tau \right) < \varepsilon$.

Theorem 17. These security notions are connected via constructive black-box reductions as shown on Fig. 1.

![Diagram](image_url)

**Fig. 1.** Relations among security notions for hash function families

We finally note that $U$-$\text{PRE}$, $U$-$\text{SEC}$ and $U$-$\text{COL}$ are identical to the notions $\text{Pre}$, $\text{Sec}$ and $\text{Coll}$ of [34].
4 Hash-and-Sign Signatures and Related Security Notions

4.1 Definitions

This section gives a number of definitions for signature schemes. In particular, we define the notion of probabilistic hash-and-sign signatures which can be seen as a generalization of well-known hash-and-sign signatures (we will adopt the term of deterministic hash-and-sign signatures to designate those). It is crucial to note here that the notions of probabilistic (resp. deterministic) hash-and-sign signatures have nothing to do with the notions of probabilistic (resp. deterministic) signatures. The signature schemes we consider are most of the time probabilistic unless otherwise stated. What we care about is in fact the probabilistic or deterministic nature of their hash-and-sign mechanism.

To avoid any confusion, we make use of distinct notations for probabilistic and deterministic signatures.

Definition 18 (Signature scheme). A signature scheme $S$ with message space $M \subseteq \{0, 1\}^*$ is identified to a tuple $S \triangleq (S.Gen, S.Sign, S.Ver)$ of algorithms. $S.Gen()$ is a probabilistic algorithm which takes no input and outputs a pair of strings $(pk, sk)$. A signature on message $M \in M$ is an $s$-bit string $\sigma = S.Sign(sk, M, u)$ where $u \in \{0, 1\}^u$. $S.Ver(pk, M, \sigma)$ outputs 1 if $\sigma = S.Sign(sk, M, u)$ for some $u \in \{0, 1\}^u$, 0 otherwise. The message space is either $M = \{0, 1\}^m$ for some integer $m > 0$ or $M = \{0, 1\}^*$ when messages can be of arbitrary length.

Definition 19 (Deterministic hash-and-sign signatures). A deterministic hash-and-sign signature scheme is a pair $S = (H, \Sigma)$ where $H : \{0, 1\}^* \rightarrow \{0, 1\}^m$ is a hash function and $\Sigma$ is a fixed-length signature scheme signing $m$-bit messages under $u$ bits of randomness. $S.Gen \triangleq \Sigma.Gen$ and a signature on $M \in \{0, 1\}^*$ with respect to $S$ is computed as $\sigma = \Sigma.Sign(sk, m, u)$ where $m = H(M)$ and $u \leftarrow \{0, 1\}^u$. $S.Ver(pk, M, \sigma)$ outputs $\Sigma.Ver(pk, H(M), \sigma)$. Therefore $S$ supports arbitrary-length messages.

We view the deterministic hash-and-sign paradigm as a scheme compiler which, taking a hash function $H$ and a fixed-length, message-space-compatible signature scheme $\Sigma$ as inputs, builds the domain-extended scheme $S = (H, \Sigma)$. To properly capture probabilistic hash-and-sign signatures in the same fashion, we first need to take a closer look at the inner computations of a signature scheme.

Definition 20 (Two-step signatures). A signature scheme $S$ with message space $M \subseteq \{0, 1\}^*$ is said to be two-step if

i) there exist two deterministic algorithms $S_1, S_2$ such that for any pair $(M, u) \in M \times \{0, 1\}^u$, $S.Sign(sk, M, u) = S_2(sk, M, r, aux)$ where $(r, aux) = S_1(sk, u)$ and $r \in \{0, 1\}^r$. We impose that $r$ is uniform over $\{0, 1\}^r$ if $u$ is uniform over $\{0, 1\}^u$;

ii) there exist two deterministic algorithms $V_1, V_2$ such that $S.Ver(pk, M, \sigma) = V_2(pk, M, \sigma, \hat{r})$ where $\hat{r} = V_1(pk, \sigma)$;

iii) if there exists $u \in \{0, 1\}^u$ such that $\sigma = S.Sign(sk, M, u)$ then the strings $r, \hat{r}$ such that $(r, aux) = S_1(sk, u)$ for some aux and $\hat{r} = V_1(pk, \sigma)$ are equal. We define the type of $S$ as $(m, u, r, s)$ if $M = \{0, 1\}^m$ and $(s, u, r, s)$ if $M = \{0, 1\}^*$. The property of being two-step seems totally unrestrictive as we do not know any example of a non-two-step signature scheme. Definition 20 merely serves at introducing notations.

Definition 21 (Probabilistic hash-and-sign signatures). A probabilistic hash-and-sign signature scheme is a pair $S = (F, \Sigma)$ where $F : \{0, 1\}^* \times \{0, 1\}^r \rightarrow \{0, 1\}^m$ is a hash function family with index space $\{0, 1\}^r$ and $\Sigma$ is a two-step signature scheme of type $(m, u, r, s)$ for integers $m, u, s > 0$. The key generation of $S$ is identical to the one of $\Sigma$. Let $\Sigma_1, \Sigma_2, T_1, T_2$ be the inner functions of $\Sigma$. A signature on $M \in \{0, 1\}^*$ is computed as $\sigma = \Sigma_2(sk, m, r, aux)$ where $m = F(M, r)$, $(r, aux) = \Sigma_1(sk, u)$ and $u \leftarrow \{0, 1\}^u$. $S.Ver(pk, M, \sigma)$ first computes $\hat{r} = T_1(pk, \sigma)$, then $\hat{m} = F(M, \hat{r})$ and outputs $T_2(pk, \hat{m}, \sigma, \hat{r})$.

It is easily checked that $S = (F, \Sigma)$ is a two-step signature scheme of type $(s, u, r, s)$ with inner functions $S_1 \triangleq \Sigma_1$, $S_2(sk, M, r, aux) = \Sigma_2(sk, F(M, r), r, aux)$, $V_1 \triangleq T_1$ and $V_2(pk, M, \sigma, \hat{r}) = T_2(pk, F(M, \hat{r}), \sigma, \hat{r})$. Also, deterministic hash-and-sign signatures can be seen as a particular case of probabilistic hash-and-sign signatures where the index space of $F$ is empty i.e. when $F(M, r) = H(M)$. Definition 21 is then easily reformulated in the terms of Definition 19 by posing $aux \triangleq u$, $r = 0$ and $r = \emptyset$ (thus letting $\Sigma_1(sk, u) = (\emptyset, u)$) and setting $\Sigma_2 \triangleq \Sigma.Sign$, $T_1 \triangleq \emptyset$ and $T_2 \triangleq \Sigma.Ver$. We now define two additional properties that will be essential in further sections.
Definition 22 (Primitiveness). Let $S = \langle F, \Sigma \rangle$ be a probabilistic hash-and-sign signature scheme and let $\Sigma_1$, $\Sigma_2$, $T_1$, $T_2$ be the inner functions of $\Sigma$. $S$ is said to be $\tau$-primitive when we know a $\tau$-time probabilistic algorithm $S_{\text{Prim}}$ which, for all pairs $(pk, sk) \leftarrow S_{\text{Gen}}()$, takes $pk$ as input and outputs a random pair $(m, \sigma = S_{\text{Sign}}(sk, m, u))$ such that a) $m$ is uniformly distributed over $\{0, 1\}^m$; b) $u$ is uniformly distributed over $\{0, 1\}^\tau$.

Note that $S$ being primitive implies that $\Sigma$ is existentially forgeable (EF) under a key-only attack (KOA). In essence, if $S = \langle F, \Sigma \rangle$ is primitive then replacing $F$ by the identity function structurally destroys the EF-KOA security of the modified scheme. This notion captures the intuition that $F$ somehow plays an important role in the EF-KOA security of $S$.

Definition 23 (Injectivity). Let $S = \langle F, \Sigma \rangle$ be a probabilistic hash-and-sign signature scheme and $\Sigma_1$, $\Sigma_2$, $T_1$, $T_2$ as above. $S$ is said to be injective when for any key pair $(pk, sk)$ and any $\sigma \in \{0, 1\}^\tau$, there exists at most one pair $(m, r) \in \{0, 1\}^m \times \{0, 1\}^\tau$ such that $\sigma = \Sigma_2(sk, m, r, \text{aux})$ and $(r, \text{aux}) = \Sigma_1(sk, u)$ for some $u, \text{aux}$.

It appears that most signature schemes used in practice, beside being probabilistic hash-and-sign schemes, are also primitive and injective. This is at least the case with Schnorr, FDH and PSS signatures. Other practical schemes such as DSA or GHR are also of interest but do not seem to be primitive. This observation motivates us to categorize signatures schemes according to the four definitions above and study these separately.

4.2 Classifying Common Signature Schemes

Signature schemes obtained by applying the Fiat-Shamir transform to 3-move identification protocols are a typical example of probabilistic hash-and-sign signatures that are both primitive and injective. Some variants of Fiat-Shamir-converted schemes such as KCDSA also fall in this category. Finally, all classical hash-then-invert signature schemes based on a trapdoor one-way permutation such as PSS are complete in the sense of Definitions 22 and 23 as well. Other schemes are not. To illustrate this variety, we review and characterize the signature schemes most commonly found in good cryptographic practice. We report our classification results on Table 1.

<table>
<thead>
<tr>
<th>Signature Scheme</th>
<th>Deterministic Hash-and-Sign</th>
<th>Probabilistic Hash-and-Sign</th>
<th>Primitive</th>
<th>Injective</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schnorr</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>FDH</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>PFDH</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>PSS</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>EMSA-PSS</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>BLS</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Generic DSA</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>GHR</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>CS</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

Table 1. A classification of common signature schemes

We refer to Appendix A for a description of these signature schemes and evidence of their classification.

4.3 Security Notions for Signatures

We recall the security notions related to signature schemes that will be of interest in this paper. Security notions combine an adversarial goal with an attack model. The attacker is seen as a probabilistic polynomial time algorithm attempting to fulfill its goal while being given a number of computational resources. The attacker may interact with the scheme in different ways.
**Adversarial goals.** A signature scheme is said to be universally forgeable (UF) when there exists an adversary that returns a valid signature on a randomly chosen message \( M \leftarrow \mathcal{M} \) given as input. When \( \mathcal{M} = \{0,1\}^* \), \( M \) is uniformly selected from \( \{0,1\}^n \) for finite \( n > 0 \), thus defining a collection of goals \( \{\text{UF}_n\}_{n>0} \). The notion of existential forgery \( \text{EF} \) (resp. \( \{\text{EF}^n\}_{n>0} \)) is similar to UF but allows the adversary to choose freely the value of the signed message.

We also consider non-repudiation, a scenario where the attacker is the signer itself and therefore knows the secret key \( \text{sk} \) (See [27] for a general consideration on the subject). A universal repudiation (UR) occurs when, given a random message \( M_1 \leftarrow \mathcal{M} \), the adversary outputs a different message \( M_2 \in \mathcal{M} \setminus \{M_1\} \) and a signature \( \sigma \) such that \( \sigma \) is a valid signature on both \( M_1 \) and \( M_2 \). When \( \mathcal{M} = \{0,1\}^* \), we draw \( M_1 \) from \( \{0,1\}^n \) uniformly at random, impose \( M_2 \in \{0,1\}^{n_2} \) and define a collection of goals \( \{\text{UR}^n\}_{n_1,n_2>0} \). An existential repudiation \( \text{ER} \) (respectively \( \{\text{ER}^{n_1,n_2}\}_{n_1,n_2>0} \)) is a tuple \( (M_1,M_2,\sigma) \) where the adversary chooses \( M_1 \in \{0,1\}^{n_1}, M_2 \in \{0,1\}^{n_2} \) and \( \sigma \) freely, with the obvious restriction \( M_1 \neq M_2 \).

**Attack models.** We consider several attack scenarios. In a key only attack (KOA), the adversary is given nothing else than a public key as input. A known message attack (KMA) consists in giving as input to the attacker a list \( (M_1,\sigma_1), \ldots, (M_\ell,\sigma_\ell) \) of random and pairwise distinct message-signature pairs. When \( \mathcal{M} = \{0,1\}^* \), we draw \( M_i \) from \( \{0,1\}^n \) for \( i \in [1..\ell] \), thereby defining a collection of attacks \( \{\text{KMA}_n\}_{n>0} \). In a chosen message attack (CMA), the adversary is given adaptive access to a signing oracle.

**Relations among security levels.** We view security notions as computational problems. For instance \( \text{UF}_{n_1}\text{-KMA}_{n_2} [S] \) is the problem of computing a universal forgery under known message attack. This notation allows to relate security notions using reductions. In the case of KMA or CMA, we denote by \( \ell\text{-GOAL-}\text{ATK} [S] \) the problem of breaking \( \text{GOAL} \) in no more than \( \ell \) calls to the resource defined by \( \text{ATK} \). Thus, breaking \( \ell\text{-EF-}\text{CMA} [S] \) authorizes at most \( \ell \) calls to the signing oracle to break \( \text{EF} \). Fig. 2 displays the well-known black-box (and constructive) reductions among security levels. Indices \( n_1,n_2 \) reflecting the case \( \mathcal{M} = \{0,1\}^* \) are to be removed if \( \mathcal{M} = \{0,1\}^n \).

![Relations among security notions for signature schemes](image)

**Fig. 2.** Relations among security notions for signature schemes

## 5 Security Relations for Deterministic Hash-and-Sign Signatures

Our goal is to exhaust the security reductions standing between a deterministic hash-and-sign signature scheme \( S = (H,\Sigma) \) and its inner hash function \( H \). The signature scheme \( \Sigma \) plays the role of a parameter here. These reductions may have three different flavors. First, we show how breaking certain security properties of \( H \) allows to break the signature scheme (attacks). Then we show an equivalence between performing certain attacks on \( S \) and finding security failures in the underlying hash function \( H \) (security proofs). We finally show that certain security notions for \( S \) and \( H \) are computationally independent, meaning that they cannot be compared using black-box reductions. This independence shows the non-existence of certain attacks and security proofs. We start by listing attacks.
5.1 Attacking $S = \langle H, \Sigma \rangle$ by Attacking $H$

Lemma 24. Let $S = \langle H, \Sigma \rangle$ be a deterministic hash-and-sign signature scheme as per Definition 19. Then for any integers $n_1, n_2 > 0$,

\[
\begin{align*}
\text{COL}^{n_1, n_2}[H] &\Rightarrow 1\text{-EF}^{n_1, \text{CMA}}[S], 1\text{-EF}^{n_2, \text{CMA}}[S] \\
\text{COL}^{n_1, n_2}[H] &\Rightarrow \text{ER}^{n_1, n_2}[S] \\
\Sigma \text{EC}^{n_1}_1[H] &\Rightarrow 1\text{-UF}^{n_1, \Sigma\text{CMA}}[S] \\
\Sigma \text{EC}^{n_2}_1[H] &\Rightarrow 1\text{-EF}^{n_2, \Sigma\text{KMA}}[S] \\
\Sigma \text{EC}^{n_2}_1[H] &\Rightarrow \Sigma\text{UR}^{n_2}_1[S]
\end{align*}
\]

Proof (COL$^{n_1, n_2}[H] \Rightarrow 1\text{-EF}^{n_1, \text{CMA}}[S]$). Given a $(\tau_H, \varepsilon_H)$-solver $A_H$ for COL$^{n_1, n_2}[H]$ we build an EF$^{n_1, \text{CMA}}$ attacker $A_S$ breaking $S$ with probability $\varepsilon_S = \varepsilon_H$ and $\tau_S = \tau_H + \text{Time}(S, \text{Sign}, n_2)$ in exactly one call to the signing oracle. Given a random public key $pk \leftarrow S,\text{Gen}()$, our attacker $A_S$ runs $A_H$ to produce an $(n_1, n_2)$-collision $(M_1, M_2)$ such that $M_1 \neq M_2$ and $H(M_1) = H(M_2)$. Then $A_S$ requests a signature $\sigma = \Sigma,\text{Sign}(sk, H(M_2), u)$ on $M_2$ and produces $\sigma$ as a valid signature on $M_1 = M_2$. The same works with 1-EF$^{n_2, \Sigma\text{CMA}}$ as well by symmetry.

Proof (COL$^{n_1, n_2}[H] \Rightarrow \text{ER}^{n_1, n_2}[S]$). Given a $(\tau_H, \varepsilon_H)$-collision-finder $A_H$ we build a repudiator $A_S$ breaking ER$^{n_1, n_2}[S]$ with probability $\varepsilon_S = \varepsilon_H$ and time $\tau_S = \tau_H + \text{Time}(S, \text{Sign}, n_1)$. Given a random key pair $(pk, sk) \leftarrow S,\text{Gen}()$, $A_S$ runs $A_H$ to build an $(n_1, n_2)$-collision. $A_S$ then signs $M_1$ under randomness $u \in \{0, 1\}^u$ to get $\sigma = \Sigma,\text{Sign}(sk, H(M_1), u)$ and outputs the tuple $(M_1, M_2, \sigma)$.

Proof (SEC$^{n_1}_1[H] \Rightarrow 1\text{-UF}^{n_1, \Sigma\text{CMA}}[S]$). Assume $A_H(\tau_H, \varepsilon_H)$-solves SEC$^{n_1}_1[H]$. We build a UF$^{n_1, \Sigma\text{CMA}}$ attacker $A_S$ which $(\tau_S, \varepsilon_S)$-breaks $S$ in one call to the signing oracle with $\varepsilon_S = \varepsilon_H$ and time $\tau_S = \tau_H + \text{Time}(S, \text{Sign}, n_2)$. Given a public key $pk \leftarrow S,\text{Gen}()$ and $M_1 \leftarrow \{0, 1\}^m$, $A_S$ must produce a valid signature on $M_1$. $A_S$ runs $A_H(M_1)$ to build a second preimage $M_2 \neq M_1$ with $|M_2| = n_2$ and $H(M_2) = H(M_1)$. $A_S$ then requests a signature on $M_2$ and is given $\sigma = \Sigma,\text{Sign}(sk, H(M_2), u)$ for some $u \in \{0, 1\}^u$. $A_S$ then returns $(M_1, \sigma)$.

Proof (SEC$^{n_2}_1[H] \Rightarrow 1\text{-EF}^{n_2, \Sigma\text{KMA}}[S]$). Assume $A_H(\tau_H, \varepsilon_H)$-solves SEC$^{n_2}_1[H]$, we build an attacker $A_S$ which $(\tau_S, \varepsilon_S)$-solves 1-EF$^{n_2, \Sigma\text{KMA}}[S]$ with $\varepsilon_S = \varepsilon_H$ and $\tau_S = \tau_H$. $A_S$ is given a random $pk \leftarrow S,\text{Gen}()$ and a single random message-signature pair $(M_1, \sigma)$ where $M_1 \leftarrow \{0, 1\}^m$ and $\sigma = \Sigma,\text{Sign}(sk, H(M_1), u)$ for $u \leftarrow \{0, 1\}^u$. $A_S$ runs $A_H(M_1)$ to construct a second preimage $M_2 \neq M_1$ with $|M_2| = n_2$ and $H(M_2) = H(M_1)$. $A_S$ then returns the existential forgery $(M_1, \sigma)$.

Proof (SEC$^{n_2}_1[H] \Rightarrow \Sigma\text{UR}^{n_2}_1[S]$). Assuming $A_H(\tau_H, \varepsilon_H)$-solves SEC$^{n_2}_1[H]$, we build a $(\tau_S, \varepsilon_S)$-repudiator $A_S$ which on input a random key pair $(pk, sk) \leftarrow S,\text{Gen}()$ and a random message $M_1 \in \{0, 1\}^m$, returns a message $M_2 \neq M_1$, $|M_2| = n_2$ and $\sigma$ such that $\sigma$ is a valid signature on both $M_1$ and $M_2$. Here again, $\varepsilon_S = \varepsilon_H$ and $\tau_S = \tau_H + \text{Time}(S, \text{Sign}, n_2)$. $A_S$ runs $A_H$ on $M_1$ to produce a second message $M_2 \in \{0, 1\}^{n_2}$ with $H(M_1) = H(M_2)$. $A_S$ then produces a signature $\sigma$ on $M_2$ and outputs $(M_2, \sigma)$.

Lemma 25 (The case of primitive signatures). Let $S = \langle H, \Sigma \rangle$ be a deterministic hash-and-sign signature scheme and assume that $S$ is primitive. Then

\[
\forall n > 0, \quad \text{PRE}^n[H] \Rightarrow \text{EF}^n,\text{-KOA}[S].
\]

Proof (PRE$^n[H] \Rightarrow \text{EF}^n,\text{-KOA}[S]$). Assuming black-box access to $A_H$ which $(\tau_H, \varepsilon_H)$-breaks PRE$^n[H]$, we build an EF$^n,\text{-KOA}$ adversary $A_S$ which $(\tau_S, \varepsilon_S)$-breaks $S$ where $\varepsilon_S = \varepsilon_H$ and $\tau_S = \tau_H + \text{Time}(S, \text{Prim})$. $A_S$ is given a random key $pk \leftarrow S,\text{Gen}()$. Since $S$ is primitive, $A_S$ can generate a random pair $(m, \sigma = \Sigma,\text{Sign}(sk, m, u))$ by running $S,\text{Prim}(pk)$. Note that $m$ is uniformly distributed over $\{0, 1\}^m$. $A_S$ then runs $A_H(m)$ to produce $M \in \{0, 1\}^n$ such that $H(M) = m$. $A_S$ then produces a signature $\sigma$ on $M$ and outputs $(M, \sigma)$.

5.2 Proving $S = \langle H, \Sigma \rangle$ Secure Assuming $H$ Secure

We now turn to the positive security relations between $S$ and $H$. As confirmed later in the paper, there seems to be no security proof relating the unforgeability(ies) of $S$ to $H$. Nevertheless, we evidence that the levels of repudiability of $S$ can be guaranteed under security assumptions on $H$. This assumes, however, that $S$ be injective.
Lemma 26 (The case of injective signatures). Let $S$ be a deterministic hash-and-sign signature scheme and assume $S$ is injective. Then for any $n_1, n_2 > 0$,

$$\text{COL}^n_{1,n_2}[H] \Leftarrow \text{ER}^{n_1,n_2}[S]$$

(7)

and therefore $\text{SEC}^n_{1,n_2}[H] \Leftarrow \text{UR}^{n_2}_{n_1}[S]$ and $\text{COL}^n_{1,n_2}[H] \Leftarrow \text{ER}^{n_1,n_2}[S]$ in virtue of (3) and (6).

Proof /COL$^{n_1,n_2}[H] \Leftarrow \text{ER}^{n_1,n_2}[S]$. Let us assume an adversary $A_H$ which $(\tau_H, \varepsilon_H)$-breaks $\text{COL}^{n_1,n_2}[H]$ with $\varepsilon_H = \varepsilon_S$ and $\tau_H = \tau_S + \text{Time}(S.\text{Gen})$. $A_H$ generates a random key pair $(pk, sk) \leftarrow S.\text{Gen}()$ and runs $A_S(pk, sk)$. If $A_S$ outputs $(M_1, M_2, \sigma)$ where $M_1 \in \{0, 1\}^{n_1}$, $M_2 \in \{0, 1\}^{n_2}$, $M_2 \neq M_1$ and $\sigma$ is a valid signature on $M_1$ and $M_2$, then $A_H$ discards its and outputs $(M_1, M_2)$. If $\sigma$ is a signature on $M_1$ and $M_2$ simultaneously then $\sigma = \Sigma_2(s_k, H(M_1), r_1, aux_1) = \Sigma_2(s_k, H(M_2), r_2, aux_2)$ for some $aux_1, aux_2$ and $r_1, r_2 \in \{0, 1\}^*$. Since $S$ is injective, one must have $r_1 = r_2$ and $H(M_1) = H(M_2)$ so that $(M_1, M_2)$ is an $(n_1, n_2)$-collision. $\square$

Proof /SEC$^{n_1}_{1}[H] \Leftarrow \text{UR}^{n_2}_{n_1}[S]$. Assuming a $(\tau_S, \varepsilon_S)$-universal repudiator $A_S$, we build an algorithm $A_H$ which $(\tau_H, \varepsilon_H)$-breaks $\text{SEC}^{n_1}_{1}[H]$ with $\varepsilon_H = \varepsilon_S$ and $\tau_H = \tau_S + \text{Time}(S.\text{Gen})$. Given a random $M_1 \leftarrow \{0, 1\}^{n_1}$, $A_H$ generates a random key pair $(pk, sk) \leftarrow S.\text{Gen}()$ and runs $A_S(pk, sk, M_1)$ to construct a pair $(M_2, \sigma)$ where $M_2 \in \{0, 1\}^{n_2}$, $M_2 \neq M_1$ and $\sigma$ is a valid signature on $M_1$ and $M_2$. The latter condition implies $\sigma = \Sigma_2(s_k, H(M_1), r_1, aux_1) = \Sigma_2(s_k, H(M_2), r_2, aux_2)$ for some $aux_1, aux_2$ and $r_1, r_2 \in \{0, 1\}^*$. Since $S$ is injective, one must have $r_1 = r_2$ and $H(M_1) = H(M_2)$ so that $M_2$ yields an $n_2$-bit second preimage of $H(M_1)$. $A_H$ then outputs $M_2$. $\square$

5.3 Impossible Reductions Between Security Notions for $S = \langle H, \Sigma \rangle$ and $H$

Let $S = \langle H, \Sigma \rangle$ be a deterministic hash-and-sign signature scheme. The results above show that breaking $H$ in the strongest sense, namely breaking $\text{PRE}[H]$, is enough to break $\text{EF-ATK}[S]$ for $\text{ATK} \in \{\text{KOA, KMA, CMA}\}$ as well as $\text{UF-CMA}[S]$ but seems insufficient to break either $\text{UF-KOA}[S]$ or $\text{UF-KMA}[S]$. We now want to ascertain that it is actually impossible for these two security levels to fall even when one assumes breaking $\text{PRE}[H]$ is easy.

What does it take to break $\text{UF-KOA}[S]$? We consider $\text{UF-KOA}[S]$ and show that this security level does not fall when $\text{PRE}[H]$ is broken. This impossibility is based on the following simple observation:

Lemma 27. For any $n > 0$, $\text{UF}_n\cdot\text{KOa}[S] \Leftarrow \text{UF-KOA}[\Sigma] \Leftarrow \text{PRE}^n[H] \wedge \text{UF}_n\cdot\text{KOa}[S]$.

Proof. (Left $\Leftarrow$). Given a $(\tau_{\Sigma}, \varepsilon_{\Sigma})$-universal forger $A_{\Sigma}$ for $\Sigma$, we build as follows a reduction algorithm $A_S$ which $(\tau_S, \varepsilon_S)$-breaking $\text{UF}_n\cdot\text{KOa}[S]$ for any $n > 0$ with $\varepsilon_S = \varepsilon_{\Sigma}$ and $\tau_S = \tau_{\Sigma} + \text{Time}(H, n)$. Given a random $M \leftarrow \{0, 1\}^n$, $A_S$ computes $m = H(M)$, runs $A_{\Sigma}(m)$ and outputs $A_{\Sigma}$’s output. (Right $\Leftarrow$). Assume we are given two probabilistic algorithms $A_H$ and $A_S$ such that $(\tau_{H}, \varepsilon_{H})$-breaks $\text{PRE}^n[H]$ and $A_S(\tau_S, \varepsilon_S)$-breaks $\text{UF}_n\cdot\text{KOa}[S]$. We construct a reduction algorithm $A_{\Sigma}$ which $(\tau_{\Sigma}, \varepsilon_{\Sigma})$-breaks $\text{UF}_{n}\cdot\text{KOa}[\Sigma]$ with $\tau_{\Sigma} = \varepsilon_{H} + \varepsilon_{\Sigma}$ and $\tau_{\Sigma} = \tau_{H} + \tau_{\Sigma}$. Given a random $pk \leftarrow \Sigma.\text{Gen}$ and a random $m \leftarrow \{0, 1\}^n$, $A_{\Sigma}$ runs $A_{H}(m)$ to get $M \in \{0, 1\}^n$ such that $H(M) = m$ and runs $A_{S}(M)$ to get $r = \Sigma(H(M), r) = \Sigma(m, r)$ for some $r \in \{0, 1\}^*$. We now see that a polynomial reduction $R$ such that $\text{PRE}^n[H] \geq R \text{UF}_n\cdot\text{KOa}[S]$ is very unlikely. Indeed, assuming a reduction $R_1$ converting an algorithm solving $\text{PRE}^n[H]$ into an attacker $A_S$ against $\text{UF}_n\cdot\text{KOa}[S]$, we would build a second reduction $R_2$ which converts $A_H$ into an algorithm breaking $\text{UF}_{n}\cdot\text{KOa}[\Sigma]$ with a similar efficiency. Since $H$ and $\Sigma$ are two independent ingredients of $S$ and because $\Sigma$ is a scheme parameter here, such a hierarchy does not exist. If it existed, then $\Sigma$ and $H$ would admit some form of “morphological interaction” such as sharing identical components. This cannot be the case for general $\Sigma$’s. In fact, the only effect breaking $\text{PRE}[H]$ has on $S$ is that for any $n > 0$, $\text{UF}_n\cdot\text{KOa}[S] \Leftarrow \text{UF-KOA}[\Sigma]$, thus rendering $S$ exactly as secure as $\Sigma$ in the sense of UF-KOA security.

Can $\text{PRE}[H]$ break $\text{UF-KMA}[S]$ then? It is easily seen that the reductions of Lemma 27 cannot be readily extended to security levels $\{\text{UF}_{n}\cdot\text{KMA}\}_{n \geq 0}$. If the reduction $\ell\cdot\text{UF}_{n}\cdot\text{KMA}[S] \Leftarrow \ell\cdot\text{UF-KMA}[\Sigma]$ stands for any $n > 0$ and $\ell \geq 0$, the only security reduction $\ell\cdot\text{UF-KMA}[\Sigma] \Leftarrow \text{PRE}^n[H] \wedge \ell\cdot\text{UF}_{n}\cdot\text{KMA}[S]$ which seems to work consists in converting the $\ell$ message-signature pairs given to the $\ell\cdot\text{UF-KMA}[\Sigma]$ attacker into a list of $\ell$ message-signature pairs for $S$. This requires breaking $\text{PRE}^n[H]$ exactly $\ell$ times, thereby introducing a $\varepsilon_{H}^\ell$ term in the reduction cost. Such a reduction is therefore loose as $\ell$ grows and cannot be seen as efficient for large $\ell$’s. We leave this question fully open.
6 Security Relations for Probabilistic Hash-and-Sign Signatures

We now move on to the case of a probabilistic hash-and-sign signature scheme \( S = \langle F, \Sigma \rangle \) where similarly as in the deterministic case, our goal is to relate the security of \( S \) to the one of its hash component \( F \), the underlying signature scheme \( \Sigma \) being seen as a fixed parameter. \( \Sigma_1, \Sigma_2, \mathcal{Y}_1 \) and \( \mathcal{Y}_2 \) denote the inner functions of \( \Sigma \).

6.1 Attacking \( S = \langle F, \Sigma \rangle \) by Attacking \( F \)

Lemma 28. Let \( S = \langle F, \Sigma \rangle \) be a probabilistic hash-and-sign signature scheme as per Definition 21. Then for any integers \( n_1, n_2 > 0 \),

\[
\begin{align*}
\text{A-COL}^{n_1, n_2} [F] &\Rightarrow 1\text{-EF}^{n_1}\text{-CMA} [S], 1\text{-EF}^{n_2}\text{-CMA} [S] \quad (9) \\
\text{U-COL}^{n_1, n_2} [F] &\Rightarrow \text{ER}^{n_1, n_2} [S] \quad (10) \\
\text{A-SEC}^{n_2} [F] &\Rightarrow 1\text{-UF}_{n_1}\text{-CMA} [S] \quad (11) \\
\text{U-SEC}^{n_2} [F] &\Rightarrow 1\text{-EF}^{n_2}\text{-KMA}_{n_1} [S] \quad (12) \\
\text{U-SEC}^{n_2} [F] &\Rightarrow \text{UR}^{n_2} [S] \quad (13)
\end{align*}
\]

Proof. (A-COL\(^{n_1, n_2} [F] \Rightarrow 1\text{-EF}^{n_1}\text{-CMA} [S])\. Given a \((\tau_F, \varepsilon_F)\)-solver \( \mathcal{A}_F \) for a A-COL\(^{n_1, n_2} [F] \) we build an EF\(^{n_1}\)-CMA attacker \( \mathcal{A}_S \) breaking \( S \) with probability \( \varepsilon_S = \varepsilon_F \) and \( \tau_S = \tau_F + \text{Time} (S.\text{Sign}, n_2) \) in exactly one call to the signing oracle. Given a random public key \( \text{pk} \leftarrow S.\text{Gen}() \), our attacker \( \mathcal{A}_S \) runs \( \mathcal{A}_F \) to produce an \((n_1, n_2)\)-absolute-collision \((M_1, M_2)\), namely a pair \((M_1, M_2) \in \{0, 1\}^{n_1} \times \{0, 1\}^{n_2} \) such that \( M_1 \neq M_2 \) and \( F(M_1, r) = F(M_2, r) \) for any \( r \in \{0, 1\}^\tau \). Then \( \mathcal{A}_S \) requests a signature \( \sigma = \Sigma_2 (\text{sk}, F(M_2, r), r, \text{aux}) \) on \( M_2 \) where \((r, \text{aux}) = \Sigma_1 (\text{sk}, u) \) for some \( u \in \{0, 1\}^u \) and produces \( \sigma \) as a valid signature on \( M_1 \neq M_2 \). The same works with 1-EF\(^{n_2}\)-CMA as well by symmetry.

Proof. (U-COL\(^{n_1, n_2} [F] \Rightarrow \text{ER}^{n_1, n_2} [S])\. Given a \((\tau_F, \varepsilon_F)\)-existential-collision-finder \( \mathcal{A}_F \) we build a repudiator \( \mathcal{A}_S \) breaking \( \text{ER}^{n_1, n_2} [S] \) with probability \( \varepsilon_S = \varepsilon_F \) and time \( \tau_S = \tau_F + \text{Time} (S.\text{Sign}, n_1) \). Given a random key pair \((\text{pk}, \text{sk}) \leftarrow S.\text{Gen}() \), \( \mathcal{A}_S \) randomly picks \( u \leftarrow \{0, 1\}^u \) and computes \((r, \text{aux}) = \Sigma_1 (\text{sk}, u) \). Since \( u \) is taken uniformly at random, \( r \) is uniformly distributed over \( \{0, 1\}^\tau \). Now \( \mathcal{A}_S \) runs \( \mathcal{A}_F (r) \) to build an \((n_1, n_2)\)-collision for \( r \), i.e. \((M_1, M_2) \in \{0, 1\}^{n_1} \times \{0, 1\}^{n_2} \) such that \( F(M_2, r) = F(M_1, r) \). Then \( \mathcal{A}_S \) finishes to sign \( M_1 \) by computing \( \sigma = \Sigma_2 (\text{sk}, F(M_1, r), r, \text{aux}) \) and outputs the tuple \((M_1, M_2, \sigma)\).

Proof. (A-SEC\(^{n_2} [F] \Rightarrow 1\text{-UF}_{n_1}\text{-CMA} [S])\. Assume that \( \mathcal{A}_F (\tau_F, \varepsilon_F)\)-solves A-SEC\(^{n_2} [F] \). We build a UF\(_{n_1}\)-CMA attacker \( \mathcal{A}_S \) which \((\tau_S, \varepsilon_S)\)-breaks \( S \) in one call to the signing oracle with \( \varepsilon_S = \varepsilon_F \) and \( \tau_S = \tau_F + \text{Time} (S.\text{Sig}, n_2) \). Given a public key \( \text{pk} \leftarrow S.\text{Gen}() \) and \( M_1 \leftarrow \{0, 1\}^{n_1} \), \( \mathcal{A}_S \) must produce a valid signature on \( M_1 \). \( \mathcal{A}_S \) runs \( \mathcal{A}_F (M_1) \) to build an \((n_1, n_2)\)-absolute-collision \((M_1, M_2)\). \( \mathcal{A}_S \) then requests a signature on \( M_2 \) and is given \( \sigma = \Sigma_2 (\text{sk}, F(M_2, r), r, \text{aux}) \) where \((r, \text{aux}) = \Sigma_1 (\text{sk}, u) \) for some \( u \in \{0, 1\}^u \). \( \mathcal{A}_S \) then returns \((M_1, \sigma)\).

Proof. (U-SEC\(^{n_2} [F] \Rightarrow 1\text{-EF}^{n_2}\text{-KMA}_{n_1} [S])\. Assuming \( \mathcal{A}_F (\tau_F, \varepsilon_F)\)-solves SEC\(^{n_2} [F] \), we build an attacker \( \mathcal{A}_S \) which \((\tau_S, \varepsilon_S)\)-solves 1-EF\(^{n_2}\)-KMA\(_{n_1} [S] \) with \( \varepsilon_S = \varepsilon_F \) and \( \tau_S = \tau_F \). \( \mathcal{A}_S \) is given a random \( \text{pk} \leftarrow S.\text{Gen}() \) and a single random message-signature-pair \((M_1, \sigma)\) where \( M_1 \leftarrow \{0, 1\}^{n_1} \) and \( \sigma = \Sigma_2 (\text{sk}, F(M_1, r), r, \text{aux}) \) with \((r, \text{aux}) = \Sigma_1 (\text{sk}, u) \) for \( u \leftarrow \{0, 1\}^u \). Since \( u \) is uniform, note that \( r \) is uniform over \( \{0, 1\}^\tau \). \( \mathcal{A}_S \) runs \( \mathcal{A}_F (M_1, r) \) to construct an \( n_2 \)-bit string \( M_2 \neq M_1 \) with \( F(M_2, r) = F(M_1, r) \). \( \mathcal{A}_S \) then returns the existential forgery \((M_2, \sigma)\).

Proof. (U-SEC\(^{n_2} [F] \Rightarrow \text{UR}^{n_2} [S])\. Assuming \( \mathcal{A}_F (\tau_F, \varepsilon_F)\)-solves U-SEC\(^{n_2} [F], \mathcal{A}_S \) which on input a random key pair \((\text{pk}, \text{sk}) \leftarrow S.\text{Gen}() \) and a random message \( M_1 \in \{0, 1\}^{n_1} \), returns a message \( M_2 \neq M_1 \). \( M_2 \neq M_1 \). \( \mathcal{A}_S \) then computes \( \sigma = \Sigma_2 (\text{sk}, F(M_1, r), r, \text{aux}) \) and outputs \( (M_2, \sigma) \).

Lemma 29 (The case of primitive signatures). Let \( S = \langle F, \Sigma \rangle \) be a probabilistic hash-and-sign signature scheme and assume that \( \Sigma \) is primitive. Then

\[
\forall n > 0, \quad \text{U-PRE}^n [F] \Rightarrow \text{EF}^n\text{-KOA} [S]
\]
6.2 Proving $S = \langle F, \Sigma \rangle$ Secure Assuming $F$ Secure

We are now looking for positive security relations between $S$ and $F$. As in the deterministic case, it seems unlikely (although we do not disprove it) that security proof exist which relate the unforgeability of $S$ to $F$. Assuming $S$ injective, however, is enough to show that non-repudiation can be guaranteed under security assumptions on $F$.

**Lemma 30 (The case of injective signatures).** Let $S$ be a probabilistic hash-and-sign signature scheme and assume $S$ is injective. Then for any $n_1, n_2 > 0$,

$$\text{E-COL}^{n_1, n_2} [F] \leq \text{ER}^{n_1, n_2} [S]$$

$$\text{E-SEC}^{n_2} [F] \leq \text{UR}^{n_2} [S].$$

**Proof** $\text{E-COL}^{n_1, n_2} [F] \leq \text{ER}^{n_1, n_2} [S]$. Let us assume an adversary $A_S$ which $(\tau_S, \varepsilon_S)$-breaks $\text{ER}^{n_1, n_2} [S]$. We build an existential collision-finder $A_F$ which $(\tau_F, \varepsilon_F)$-breaks $\text{E-COL}^{n_1, n_2} [F]$ with $\varepsilon_F = \varepsilon_S$ and $\tau_F = \tau_S + \text{Time}(S.\text{Gen}) + \text{Time}(S.\text{Ver})$. $A_S$ is given a random key $pk \leftarrow S.\text{Gen}()$. Since $S$ is primitive, $A_S$ can generate a random pair $(m, \sigma = S.\text{Sign}(sk, m, u))$ by running $S.\text{Prim}(pk)$. Note that $m$ is uniformly distributed over $\{0, 1\}^m$ and $u$ is uniform over $\{0, 1\}^n$, thereby making $r = \tau_1(pk, \sigma)$ uniform over $\{0, 1\}^t$. Now $A_S$ runs $A_F(m, r)$ to construct $M \in \{0, 1\}^n$ such that $F(M, r) = m$. $A_S$ then outputs $(M, \sigma)$.

Discussion. As opposed to deterministic hash-an-sign signatures, the probabilistic hash-and-sign paradigm inherently offers better security guarantees. For instance, breaking $S$ in the EF-CMA sense is easy if $H$ is not collision-resistant in the deterministic case $S = \langle H, \Sigma \rangle$. Breaking the same security level for $S = \langle F, \Sigma \rangle$, however, seems to require the generation of absolute collisions for $F$, a much stronger result. Overall, given a fixed-size signature scheme $\Sigma$, it seems largely preferable to domain-extend it in the probabilistic way for which security is obtained under much weaker assumptions on the security of the inner hash component.

7 Merkle-Damgård-based Instantiations of $F$

In previous sections, we introduced and made use of new security notions for a hash function family $F : \{0, 1\}^* \times \{0, 1\}^t \rightarrow \{0, 1\}^m$. To exemplify our investigation on the impacts of $F$ on a signature scheme $S = \langle F, \Sigma \rangle$, we analyze three practical instantiations of $F$ using a Merkle-Damgård (MD for short) hash function. This is motivated by the fact that SHA-1, MD5 and most of other hash functions we use in practice are obtained by applying some variant of the Merkle-Damgård construction to an underlying compression function $f : \{0, 1\}^m \times \{0, 1\}^b \rightarrow \{0, 1\}^m$. First, we relate attackers against $F$ and attackers against the Merkle-Damgård hash function. This allows to reformulate the security results of the previous sections which involve $F$ into more specific security statements involving the MD hash function directly. Based on these reductions, we display the effective workload of the best known attacks against $F$ in every instantiation.
7.1 Instantiating $F$ with any hash function $H$

Let $H$ be a hash function. There are many ways one may attempt to construct a hash function family $F$ using $H$. We consider later on the two most “natural” constructions, namely $F(M, r) = H(M || r)$ and $F(M, r) = H(r || M)$. For these two operating modes (we actually consider a wider class $F(M, r) = H([M, r])$ of operating modes), we relate the security of $F$ – as defined in Section 3.2, to the one of $H$ – as defined in Section 3.1. Combining these new security relations with the results of Section 6, it is easy to reformulate attacks or security assumptions on hash-and-sign signatures in terms of attacks or security assumptions on $H$.

**Lemma 31.** Let $H : \{0,1\}^* \to \{0,1\}^m$ be a hash function and let $[M, r]$ denote an $(\{M\} + r)$-bit (one-to-one) encoding of the pair $(M, r) \in \{0,1\}^* \times \{0,1\}^r$. Let now $F$ be the hash function family defined on $M \in \{0,1\}^*$ and $r \in \{0,1\}^r$ (r being arbitrary) as $F(M, r) = H([M, r])$. Then for any $n_1, n_2 > 0$,

$$\text{PRE}^{n_2+r} [H] \iff \text{E-PRE}^{n_2} [F] \iff \text{U-PRE}^{n_2} [F]$$

where non-underlined relations are already known by Theorem 17.

**Proof** $\text{COL}^{n_1+r, n_2+r} [H] \iff \text{E-COL}^{n_1, n_2} [F]$.
Given an adversary $A_F$ which $(\tau_F, \varepsilon_F)$-breaks $\text{E-COL}^{n_1, n_2} [F]$, we construct a reduction $A_H$ which $(\tau_H, \varepsilon_H)$-breaks $\text{COL}^{n_1+r, n_2+r} [H]$ with $\tau_H = \tau_F$ and $\varepsilon_H = \varepsilon_F + \text{Time (encode , \cdot , n_1, r)} + \text{Time (encode , \cdot , n_2, r)}$. $A_H$ runs $A_F$ to construct a tuple $(M_1, M_2, r)$ such that $F(M_1, r) = F(M_2, r)$, that is, $H([M_1, r]) = H([M_2, r])$. $A_H$ then outputs the $(n_1 + r, n_2 + r)$-collision $([M_1, r], [M_2, r])$ on $H$.

**Proof** $\text{SEC}^{n_2+r} [H] \iff \text{U-SEC}^{n_2} [F]$.
Given an adversary $A_F$ which $(\tau_F, \varepsilon_F)$-breaks $\text{U-SEC}^{n_2} [F]$, one builds a reduction $A_H$ which $(\tau_H, \varepsilon_H)$-breaks $\text{SEC}^{n_2+r} [H]$ with $\tau_H = \tau_F + \text{Time (parse , \cdot , n_2, r)} + \text{Time (encode , \cdot , n_2, r)}$ and $\varepsilon_H = \varepsilon_F$. Given a random $M_1 \in \{0,1\}^{n_1+r}$, $A_H$ parses $M_1$ as $M_1 = [M_1, r]$ where $|M_1| = n_1$ and $|r| = r$. Then $A_H$ runs $A_F(M_1, r)$ to obtain an $n_2$-bit string $M_2$ such that $F(M_2, r) = F(M_1, r)$ i.e. $H(M_2) = H(M_1)$. $A_H$ then outputs $M_2 = [M_2, r]$.

**Proof** $\text{PRE}^{n_2+r} [H] \iff \text{E-PRE}^{n_2} [F]$.
Given an adversary $A_F$ which $(\tau_F, \varepsilon_F)$-breaks $\text{E-PRE}^{n_2} [F]$, one builds a reduction $A_H$ which $(\tau_H, \varepsilon_H)$-breaks $\text{PRE}^{n_2+r} [H]$ with $\varepsilon_H = \varepsilon_F$ and $\tau_H = \tau_F + \text{Time (encode , \cdot , n_2, r)}$. Given a random $m \in \{0,1\}^{n_2}$, $A_H$ runs $A_F(m)$ to construct a pair $(M, r) \in \{0,1\}^{n_2} \times \{0,1\}^r$ such that $F(M, r) = m$. Then $A_H$ outputs $M = [M, r]$. Obviously $H(M) = m$.

**Proof** $\text{PRE}^{n_2+r} [H] \Rightarrow \text{E-PRE}^{n_2} [F]$.
Given an adversary $A_H$ which $(\tau_H, \varepsilon_H)$-breaks $\text{PRE}^{n_2+r} [H]$, one builds a reduction $A_F$ which $(\tau_F, \varepsilon_F)$-breaks $\text{PRE}^{n_2} [F]$ with $\varepsilon_F = \varepsilon_H$ and $\tau_F = \tau_H + \text{Time (parse , \cdot , n_2, r)}$. Given a random $m \in \{0,1\}^{n_2}$, $A_F$ runs $A_H(m)$ to construct an $(n_2 + r)$-bit string $M$ such that $H(M) = m$. Then $A_F$ parses $M$ as $M = [M, r]$ where $|M| = n_2$ and $|r| = r$ and outputs $(M, r)$. Obviously $F(M, r) = m$.

7.2 Merkle-Damgård hash functions

Let $f : \{0,1\}^m \times \{0,1\}^b \to \{0,1\}^m$ be a compression function, $g : \{0,1\}^* \to (\{0,1\}^b)^*$ an arbitrary function (called message padding or padding) and $IV_0 \in \{0,1\}^m$. We recall that a hash function $H$ can be obtained from these components by using iterated hashing as described in Definition 3. We write $H = \text{ITER} [f, g, IV_0]$.

**Definition 32** (Collision-propagating paddings). Let $k_1, k_2 > 0$. A padding function $g : \{0,1\}^* \to (\{0,1\}^b)^*$ is said to be $(k_1, k_2)$-collision-propagating when for any compression function $f : \{0,1\}^m \times \{0,1\}^b \to \{0,1\}^m$ and any $IV_0 \in \{0,1\}^m$, the hash function $H = \text{ITER} [f, g, IV_0]$ is such that for any $k_1$-block string $M_1 \in \{0,1\}^{k_1 b}$ and $k_2$-block string $M_2 \in \{0,1\}^{k_2 b}$, if $H(M_1) = H(M_2)$ then $H(M_1 || M) = H(M_2 || M)$ for any $M \in \{0,1\}^*$. 

14
The basic and strengthened Merkle-Damgård constructions are a specific case of iterated hashing using specific functions $g = g_0$ or $g = g_s$ defined as follows. For any $M \in \{0,1\}^*$, $g_0(M)$ is obtained by appending to $M$ as many 0-bits as necessary to yield a $|M|/b$-block string. Let $a \leq b$ be a size parameter. Given $M \in \{0,1\}^*$, $g_s(M)$ is formed by appending a single 1-bit to $M$ and as many 0-bits as required so that the length of the so-obtained string is congruent to $b - a$ modulo $b$. Then the bitlength of $M$, seen as an $a$-bit string, is appended:

$$g_0(M) = M \parallel 0^t \quad \text{with} \quad t = -|M| \mod b,$$
$$g_s(M) = M \parallel 1 \parallel 0^t \parallel |M|_a \quad \text{with} \quad t = -(|M| + a + 1) \mod b.$$

Appending a logical length-block prior to hashing is called MD-strengthening [7,8,17]. This technique aims at preventing collision and pseudo-collision attacks which find colliding messages of different length, including trivial collisions for random IV’s, long message attacks and fixed point attacks [22,26]. The protection it seems to offer justifies the practical use of MD-strengthening in common hash functions.

Let us now fix $f : \{0,1\}^m \times \{0,1\}^b \rightarrow \{0,1\}^m$ and $IV_0 \in \{0,1\}^m$. The related Merkle-Damgård hash function without MD-strengthening is the hash function $H_0 = \mathrm{ITER}[f, g_0, IV_0]$. Its counterpart with MD-strengthening is the hash function $H_s = \mathrm{ITER}[f, g_s, IV_0]$. We may sometimes use $H_{x,IV} = \mathrm{ITER}[f, g_x, IV]$ for $x = 0$ or $s$.

**Proposition 33.** a) $g_0$ is a $(k_1, k_2)$-collision-propagating padding for any $k_1, k_2 > 0$. b) $g_s$ is a $(k, k)$-collision-propagating padding for any $k > 0$.

Hence, it is rather obvious that MD-strengthening is not enough to thwart attacks based on propagating collisions in a way or another. In fact, all known second preimage or collision-finding attacks against Merkle-Damgård hash functions with MD-strengthening generate messages with the same block length so that the padding $g_s$, which only depends on the length of the input message, is the same for generated messages and therefore does not interfere with the attack. In fact, it is easily seen that for any $n = k \cdot b > 0$, $\mathrm{PRE}^n [H_0] \nleftrightarrow \mathrm{PRE}^n [H_s]$, $\mathrm{SEC}^n [H_0] \nleftrightarrow \mathrm{SEC}^n [H_s]$ and $\mathrm{COL}^{n,n} [H_s] \nleftrightarrow \mathrm{COL}^{n,n} [H_0]$. We therefore restrict ourselves to instantiations of $F$ based on $H_s = \mathrm{ITER}[f, g_s, IV_0]$, knowing in advance that all recent SEC and COL attacks against $H_0$, which retrieve same-length colliding messages, equally apply to the real-life setting $H_s = \mathrm{ITER}[f, g_s, IV_0]$.

### 7.3 MD-based hash function families

**Lemma 34.** Let $H_s = \mathrm{ITER}[f, g_s, IV_0]$ be an MD hash function with MD strengthening and $F$ defined on $\{0,1\}^* \times \{0,1\}^\ast$ as $F(M, r) = H_s(M\parallel r)$. Then for any $n = k \cdot b > 0$,

$$\mathrm{A-SEC}^n [F] \nleftrightarrow \mathrm{SEC}^n [H_s] \nleftrightarrow \mathrm{SEC}^n [H_0]$$

Proof ($\mathrm{A-COL}^{n_1,n_2} [F] \nleftrightarrow \mathrm{COL}^{n_1,n_2} [H_s]$ for any $n_1, n_2 > 0$). Given an adversary $A_{H_s}$ which $(\tau_{H_s}, \varepsilon_{H_s})$-breaks $\mathrm{COL}^{n_1,n_2} [H_s]$, one builds a reduction $A_F$ which $(\tau_F, \varepsilon_F)$-breaks $\mathrm{A-COL}^{n_1,n_2} [F]$ with $\varepsilon_F = \varepsilon_{H_s}$ and $\tau_F = \tau_{H_s}$. $A_F$ runs $A_{H_s}$ to build an $(n_1, n_2)$-collision $(M_1, M_2)$ on $H$ and outputs it. Since $g$ is collision-propagating, one has $H(M_1\parallel r) = H(M_2\parallel r)$ for any $r \in \{0,1\}^\ast \subseteq \{0,1\}^\ast$, meaning that $(M_1, M_2)$ is an $(n_1, n_2)$-absolute collision on $F$.

Proof ($\mathrm{A-SEC}^{n_1}_s [F] \nleftrightarrow \mathrm{SEC}^{n_1}_s [H_s]$ for any $n_1, n_2 > 0$). Given an adversary $A_{H_s}$ $(\tau_{H_s}, \varepsilon_{H_s})$-breaking $\mathrm{SEC}^{n_1}_s [H_s]$, one builds a reduction $A_F$ which $(\tau_F, \varepsilon_F)$-breaks $\mathrm{A-SEC}^{n_1}_s [F]$ with $\varepsilon_F = \varepsilon_{H_s}$ and $\tau_F = \tau_{H_s}$. Given a random $M_1 \leftarrow \{0,1\}^{n_1}$, $A_F$ runs $A_{H_s}$($M_1$) to build an $(n_1, n_2)$-collision $(M_1, M_2)$ on $H$ and outputs $M_2$. Since $g$ is collision-propagating, one has $H(M_1\parallel r) = H(M_2\parallel r)$ for any $r \in \{0,1\}^\ast \subseteq \{0,1\}^\ast$, meaning that $(M_1, M_2)$ is an $(n_1, n_2)$-absolute collision on $F$.

**Lemma 35.** For $H_s = \mathrm{ITER}[f, g_s, IV_0]$ as above, let now consider $F$ defined as $F(M, r) = H_s(r\parallel M)$. Then for any $n = k \cdot b > 0$, $\mathrm{A-SEC}^n [F]$ and $\mathrm{A-COL}^{n,n} [F]$ are perfectly secure unless $f$ and $IV_0$ are voluntarily constructed otherwise.

This is even true when $F(M, r) = H_0(r\parallel M)$. To see this, assume for instance that $r$ is a multiple of the block size $b$ and pose $S = \{IV | IV = H_0(r), r \in \{0,1\}^\ast\}$. Denote $H_{0,IV} = \mathrm{ITER}[f, g_0, IV]$. Then finding an $(n, n)$ or $(n_1, n_2)$-absolute collision for $IV$ implies the existence of $M_1, M_2$ such that $\forall IV \in S, H_{0,IV}(M_1) = H_{0,IV}(M_2)$. As the set $S$ has to be large (otherwise finding collisions on $H_0 = H_{0,IV_0}$ is easy), this property is statistically unlikely for a practical hash function.
7.4 Comparing operating modes \( F(M, r) = H_s(M||r) \) and \( F(M, r) = H_s(r||M) \)

It is easily seen that a probabilistic signature scheme \( \mathcal{S} = (F, \Sigma) \) based on the operating mode \( F(M, r) = H_s(M||r) \) yields in reality deterministic hash-and-sign signatures on message subspaces \( M \in \{0,1\}^k \). Indeed posing \( H_{s,IV} = ITER[f, g_s, IV] \), it is obvious that if \( |M| = k \cdot b \) then \( H_s(M||r) = H_{s,m}(r) \) where \( m = H_0.IV_0(M) \). So on these subspaces, \( \sigma = S.Sign(sk, M, u) = \Sigma_2(sk, F(M, r), r, aux) \) where \( (r, aux) = \Sigma_1(sk, u) \) can be reformulated as \( \sigma = \Sigma_2(sk, H_{s,m}(r), r, aux) \) where \( m = H_0(M) \). This drives us to the following observation.

**Lemma 36.** Let \( \mathcal{S} = (F, \Sigma) \). If \( F(M, r) = H_s(M||r) \) then \( \mathcal{S} = (H_0, \Sigma') \) on message subspaces \( \{0,1\}^kb \) for \( k > 0 \) for some signature scheme \( \Sigma' \).

So in this case \( \mathcal{S} \) can be viewed as a deterministic signature scheme. The security benefits inherent to using the probabilistic hash-and-sign paradigm are then completely lost:

**Corollary 37 (Operating mode** \( F(M, r) = H_s(M||r) \) **). For any \( n = k \cdot b > 0 \), all security levels \( EF_n.CMA[S], UF_n.CMA[S], EF_n.KMA[S], ER_n[S] \) and \( UR_n[S] \) fall as soon as \( COL^n.n[H_0] \) or \( SEC^n[H_0] \) are broken (Lemma 24).

In the primitive signature case, even \( EF^n.KOA[S] \) is easy to break if one finds an efficient preimage attack against \( PRE^n[H_0] \).

By opposition, the operating mode \( F(M, r) = H_s(r||M) \) provides (unless the compression function \( f \) is “unnatural”) perfect security for \( A.COL^n.n[F] \) and \( A.SEC^n[F] \), thus leaving no visible breach in the security of \( \mathcal{S} \):

**Corollary 38 (Operating mode** \( F(M, r) = H_s(r||M) \) **). Combining the security reductions of Lemmas 35 and 31 and all security statements of Section 6, there is no known way to break \( \mathcal{S} \) in any sense even if \( COL^n.n, SEC^n, PRE^n \) are totally broken for \( H_0 \).

Clearly, this strongly suggests to prefer the second operating mode over the first one in practical signature implementations.

7.5 Concrete security figures for three instantiations of \( F(M, r) \)

We display concrete security workloads for effectively attacking \( F(m, r) \) under current knowledge for the three instantiations \( F(m, r) = H_s(M||r), F(m, r) = H_s(r||M) \) and \( F(m, r) = H_{s,r}(M) \) and \( H_s = MD4, MD5, SHA-0 \) and \( SHA-1 \). What we plot is the expected running time \( \tau_{H_s} \) of the attack algorithm when imposing a success probability \( \varepsilon_{H_s} \) heuristically close to one.

|                | \( H_s(M||r) \)     | \( H_s(r||M) \)     | \( H_{s,r}(M) \)     |
|----------------|---------------------|---------------------|---------------------|
| **MD4**        | ![Fig. 3](image)    | ![Fig. 3](image)    | ![Fig. 3](image)    |
| **MD5**        | ![Fig. 3](image)    | ![Fig. 3](image)    | ![Fig. 3](image)    |
| **SHA-0**      | ![Fig. 3](image)    | ![Fig. 3](image)    | ![Fig. 3](image)    |
| **SHA-1**      | ![Fig. 3](image)    | ![Fig. 3](image)    | ![Fig. 3](image)    |
| **U-PRE \([F]\)** | ![Fig. 3](image)    | ![Fig. 3](image)    | ![Fig. 3](image)    |
| **E-PRE \([F]\)** | ![Fig. 3](image)    | ![Fig. 3](image)    | ![Fig. 3](image)    |
| **A-SEC \([F]\)** | ![Fig. 3](image)    | ![Fig. 3](image)    | ![Fig. 3](image)    |
| **U-SEC \([F]\)** | ![Fig. 3](image)    | ![Fig. 3](image)    | ![Fig. 3](image)    |
| **E-SEC \([F]\)** | ![Fig. 3](image)    | ![Fig. 3](image)    | ![Fig. 3](image)    |
| **A-COL \([F]\)** | ![Fig. 3](image)    | ![Fig. 3](image)    | ![Fig. 3](image)    |
| **U-COL \([F]\)** | ![Fig. 3](image)    | ![Fig. 3](image)    | ![Fig. 3](image)    |
| **E-COL \([F]\)** | ![Fig. 3](image)    | ![Fig. 3](image)    | ![Fig. 3](image)    |

Breaking \( E-COL[F] \) is just finding collisions for the Merkle-Damgård hash function \( H_0 \) in the three cases since the value of \( r \) is unconstrained. The best attacks known are Sasaki’s new message difference for MD4 \( [35] \), Klima’s tunnels for MD5 \( [23] \), Wang’s attack on SHA-0 \( [40] \), and the latest improvements over Wang’s SHA-1 attack \( [39] \). When the input \( r \) is treated after all message blocks, these are also \( A-COL \) attacks as previously discussed. Conversely if \( r \) is processed in the beginning of the hash function, we assume that absolute collisions do not exist. Concerning preimages, there are some results that can be used. Against MD4, Yu et al. gave a differential path that allow to find preimages for a class of weak messages; in some cases \( r \) can be used to randomize the IV or the message until we have a weak message:
– with $F(M, r) = H_{0, r}(M)$, we just try as many IV’s as needed;
– with $F(M, r) = H_{0, IV_0}(r||M)$ we can randomize the message if $58 < |r| \leq 128$ (if $r$ is too short we will not find a weak one, and if it is too long the two messages will be the same).

The target collision attack against MD5 by Stevens, Lenstra and de Weger [37] can be used to generate solutions for $E_{SEC}[F]$ if $r$ is long enough (at least 4192 bits). A similar attack can certainly be done on MD4. $\infty$ indicates perfect security. Empty cells denote that we are not aware of an attack more efficient than generic attacks.

8 Conclusion

In this paper, we have investigated the impact of recent attacks on hash functions on signature schemes using them. We have suggested categories of signature schemes to tell them apart in terms of deterministic or probabilistic hash-and-sign mechanism, inventivity and primitiveness. For each of these categories (and notably for signature schemes such as FDH, PSS and Schnorr), we have shown how their security relates to the one of their inner hash component.

We also focused on the case of hash functions based on the Merkle-Damgård paradigm (which is the case of MDx and SHAx functions) by enlightening the security properties of popular operating modes, the ones one would try first to construct a hash function family. We have identified $F(M, r) = H_s(M||r)$ as the less secure one by far. A concrete conclusion of our work is the suggestion of using $F(M, r) = H_s(r||M)$ or more intricate operating modes. No security breach on probabilistic hash-and-sign signatures using $F$ is known in this case, even if $H_0$ is totally broken.

We leave as an open problem to study the impact of hash functions on encryption schemes.

References


A Short Definition of Some Classic Signature Schemes

**Schnorr.** This discrete log-based signature scheme was introduced in [36]. Let $G$ be an abelian group of prime order $q$ and generator $g$.

**Key generation** uniformly select the secret key $x \leftarrow \mathbb{Z}_q$ and deduce public key $y = g^x$.

**Signature**

- $\Sigma_1$: given $M \in \{0, 1\}^*$, select $k \leftarrow \mathbb{Z}_q$, set $r = g^k$ and output $(r, k)$.
- $\Sigma_2$: compute $m = F(M, r)$.

**Verification** compute $m = F(M, r)$ and check that $g^s y^{-m} = r$.

The signature scheme PrimSchnorr consists in simply defining the signature on $m$ as $\sigma = (s, r)$ where $r = g^k$ and $s = k + mx \mod q$ for a randomly selected $k \leftarrow \mathbb{Z}_q$. PrimSchnorr is existentially (even universally) forgeable since for any pair $(m, s) \in \mathbb{Z}_q^2$, $\sigma = (s, r)$ where $r = g^s y^{-m}$ is a valid signature on $m$. It is easily seen that Schnorr signatures are also injective.
**Full Domain Hash (FDH).** This signature scheme was proposed by Bellare and Rogaway in [2]. The scheme relies on a hash function \( H : \{0,1\}^* \rightarrow \{0,1\}^k \) and an RSA key generator \( \text{Gen}(1^k) \) returning random \( k \)-bit RSA keys.

**Key generation** randomly select \((n,e,d) \leftarrow \text{Gen}(1^k)\) and deduce public key \((n,e)\).

**Signature**
- \((H)\) compute \( m = H(M) \)
- \((\Sigma)\) given \( m \), compute \( s = m^d \mod n \) and return \( \sigma = s \).

**Verification** compute \( m = H(M) \) and check that \( s^e = m \mod n \).

FDH is obviously a deterministic hash-and-sign signature scheme. Since RSA is a trapdoor permutation, FDH is also primitive and injective.

**Probabilistic Full Domain Hash (PFDH).** PFDH is a probabilistic extension of FDH suggested in [12]. We make use of a family of hash functions \( F : \{0,1\}^* \times \{0,1\}^\prime \rightarrow \{0,1\}^k \) and an RSA key generator \( \text{Gen}(1^k) \) returning random \( k \)-bit RSA keys.

**Key generation** randomly select \((n,e,d) \leftarrow \text{Gen}(1^k)\) and deduce public key \((n,e)\).

**Signature**
- \((\Sigma_1)\) select \( r \leftarrow \{0,1\}^\prime \) and return \((r,\emptyset)\)
- \((F)\) compute \( m = F(M,r) \)
- \((\Sigma_2)\) given \( m \) and \((r,\emptyset)\), compute \( s = m^d \mod n \) and return \( \sigma = (s,r) \).

**Verification** compute \( m = F(M,r) \) and check that \( s^e = m \mod n \).

PFDH is easily seen to be a probabilistic hash-and-sign signature scheme. Given \( m \in \{0,1\}^k \), \( \text{PrimPFDH} \) simply consists in generating \( \sigma = (s,r) \) with \( s = m^d \mod n \) and \( r \leftarrow \{0,1\}^\prime \) and is therefore existentially forgeable. It is also the case that PFDH is injective.

**(Simplified) PSS.** PSS was originally conceived by Bellare and Rogaway and proposed to the IEEE P1363 working group [3]. We focus on a simplified version of PSS here. Let \( \text{Gen} \) be an RSA key generator as above and \( F : \{0,1\}^* \times \{0,1\}^\prime \rightarrow \{0,1\}^m \) and \( G : \{0,1\}^m \rightarrow \{0,1\}^\prime \) be two functions such that \( m + r + t = k \) where \( t \) is a small constant.

**Key generation** randomly select \((n,e,d) \leftarrow \text{Gen}(1^k)\) and deduce public key \((n,e)\).

**Signature**
- \((\Sigma_1)\) select \( r \leftarrow \{0,1\}^\prime \) and return \((r,\emptyset)\)
- \((F)\) compute \( m = F(M,r) \)
- \((\Sigma_2)\) given \( m \) and \((r,\emptyset)\), compute \( \text{pad}(m,r) = 0^t \| m || (r \oplus G(m)) \),
  \[ s = \text{pad}(m,r)^d \mod n \] and return \( \sigma = s \).

**Verification** letting \( \tilde{m} = s^e \mod n \), parse \( \tilde{m} \) as \( 0^t \| m || (r \oplus G(m)) \) to recover \((m,r)\). If this fails return 0. Otherwise check that \( m = F(M,r) \).

Given \( m \in \{0,1\}^m \), a signature on \( m \) with respect to \( \text{PrimPSS} \) is \( s = \text{pad}(m,r)^d \mod n \) for some randomly selected \( r \leftarrow \{0,1\}^\prime \). To build an existential forgery, randomly select \( s \leftarrow \mathbb{Z}_n \) such that \( s^e \mod n \) can be parsed as \( 0^t \| m || \rho \). This remains polynomial in \( k \) since \( t \) is a small constant. Now \( \rho = r \oplus G(m) \) for some \( r \in \{0,1\}^\prime \) meaning that \( s \) is a valid signature on \( m \). It is straightforward that any value of \( s \) admits at most one pair \((m,r)\) such that \( s = \text{pad}(m,r)^d \mod n \), so PSS is injective.

**EMSA-PSS.** PSS was originally conceived by Bellare and Rogaway and proposed to the IEEE P1363 working group [3]. We focus on a simplified version of PSS here. Let \( \text{Gen} \) be an RSA key generator as above and \( F : \{0,1\}^* \times \{0,1\}^\prime \rightarrow \{0,1\}^m \) and \( G : \{0,1\}^m \rightarrow \{0,1\}^\prime \) be two functions such that \( m + r + t = k \) where \( t \) is a small constant.
Key generation randomly select \( (n, e, d) \leftarrow \text{Gen}(1^k) \) and deduce public key \((n, e)\).

**Signature**
- \((\Sigma_1)\) select \( r \leftarrow \{0, 1\}^* \) and return \((r, \emptyset)\)
- \((F)\) compute \( m = F(M, r) \)
- \((\Sigma_2)\) given \( m \) and \((r, \emptyset)\), compute \(\text{pad}(m, r) = 0^d\|m\|(r \oplus G(m))\), \( s = \text{pad}(m, r)^d \mod n \) and return \( \sigma = s \).

**Verification**
- letting \( \tilde{m} = s^e \mod n \), parse \( \tilde{m} \) as \(0^d\|m\|(r \oplus G(m))\) to recover \((m, r)\).
- If this fails return 0. Otherwise check that \( m = F(M, r) \).

Given \( m \in \{0, 1\}^m \), a signature on \( m \) with respect to \( \text{PrimPSS} \) is \( s = \text{pad}(m, r)^d \mod n \) for some randomly selected \( r \leftarrow \{0, 1\}^* \). To build an existential forgery, randomly select \( s \leftarrow \mathbb{Z}_n \) such that \( s^e \mod n \) can be parsed as \(0^d\|m\|\rho\).

This remains polynomial in \( k \) since \( t \) is a small constant. Now \( \rho = r \oplus G(m) \) for some \( r \in \{0, 1\}^* \) meaning that \( s \) is a valid signature on \( m \). It is straightforward that any value of \( s \) admits at most one pair \((m, r)\) such that \( s = \text{pad}(m, r)^d \mod n \), so PSS is injective.

**Boneh-Lynn-Shacham (BLS).** The concept of this scheme was instantiated in [6] using bilinear maps. Let \( \mathcal{G} \) be a pairing-friendly group of prime order \( q \) and generator \( g \), and let \( \langle \cdot, \cdot \rangle : \mathcal{G}^2 \rightarrow \mathcal{G} \) denote a cryptographic bilinear map over \( \mathcal{G} \). The scheme relies on a hash function \( H : \{0, 1\}^* \rightarrow \{0, 1\}^m \) and a reversible function \( T : \{0, 1\}^m \rightarrow \mathcal{G} \) mapping \( m \)-bit strings to group elements.

**Key generation**
- uniformly select the secret key \( x \leftarrow \mathbb{Z}_q \) and deduce public key \( y = g^x \).

**Signature**
- \((H)\) compute \( m = H(M) \)
- \((\Sigma)\) given \( m \), compute \( s = T(m)^x \) and return \( \sigma = s \).

**Verification**
- compute \( m = H(M) \) and check that \( \langle s, g \rangle = \langle T(m), y \rangle \).

A signature on \( m \) with respect to \( \text{PrimBLS} \) is \( s = T(m)^x \). Therefore one can raise \( y = g^x \) to a random power \( r \) until \( g^r = T(m) \) can be reversed to yield \( m \in \{0, 1\}^m \). Then \( s = y^r = T(m)^x \) is a proper signature meaning that \( \text{PrimBLS} \) is existentially forgeable. Since \( T \) is reversible, BLS signatures are injective.

**Generic DSA.** This case captures DSA [31], ECDSA [20] and numerous variants. Let \( \mathcal{G} \) be a finite abelian group of prime order \( q \) and generator \( g \). Let \( H \) be a hash function and \( T : \mathcal{G} \rightarrow \mathbb{Z}_q \) be an arbitrary function.

**Key generation**
- uniformly select the secret key \( x \leftarrow \mathbb{Z}_q \) and deduce public key \( y = g^x \).

**Signature**
- \((H)\) compute \( m = H(M) \)
- \((\Sigma)\) given \( m \), randomly select \( k \leftarrow \mathbb{Z}_q \), set \( r = T(g^k) \), compute \( s = (m + rx)k^{-1} \mod q \) and return \( \sigma = (r, s) \).

**Verification**
- compute \( m = H(M) \) and check that \( T(g^\frac{r}{x}) = r \).

Generic DSA signatures are deterministic hash-and-sign signatures. It is easily seen that they are also injective. However, they do not seem to be primitive in general.

**Gennaro-Halevi-Rabin (GHR).** GHR was suggested in [19]. Its security is based on the strong RSA assumption and the existence of division-intractable hash functions. We refer the reader to [19] for security proofs and definitions in further detail. Following the ideas of [13], a simple way to realize this property consists in mapping bitstrings to prime numbers in a collision-resistant fashion. Let \( H : \{0, 1\}^* \rightarrow \{0, 1\}^m \cap \text{Primes} \) be such a function where \( m + t = k \) and \( t \geq 2 \) is a small constant.

**Key generation**
- randomly select a \( k \)-bit safe RSA modulus \( n = (2p' + 1)(2q' + 1) \) as well as \( u \leftarrow \mathbb{Z}_n^* \) of maximal order \( 2p'q' \). Deduce public key \((n, u)\).

**Signature**
- \((H)\) compute \( m = H(M) \)
- \((\Sigma)\) given \( m \), compute \( s = u^{m-1} \mod 2p'q' \mod n \) and define the signature as \( \sigma = s \).

**Verification**
- compute \( m = H(M) \) and check that \( s^m = u \mod n \).
GHR is a deterministic hash-and-sign scheme and is also injective since for any given \( s \in \mathbb{Z}_n \), there must exist at most one \( m \in \{0,1\}^m \cap \text{Primes} \) such that \( s^m \equiv u \mod n \). Indeed, if \( s^{m_1} = s^{m_2} = u \mod n \) with \( m_1 < m_2 \) then \( 2p'q' \) must divide \( m_2 - m_1 < 2^{k-t} \). This implies \( 2p'q' < 2^{k-t} \) which contradicts \( t \geq 2 \). However, GHR is not primitive.

**Tight-GHR signature scheme.** The security reduction of the above scheme to the flexible RSA problem is quite bad [19,11], and so the GHR signature scheme has next been modified by using a chameleon hash function [24], thus obtaining a tight scheme.

More precisely, an attacker against the obtained Tight-GHR can be used to solved either the discrete logarithm or the flexible RSA problems, with roughly the same success probability and running time. Let \( P \) and \( Q \) be large prime numbers so that \( Q \) divides \( P - 1 \), and let \( \langle g \rangle \) denote the cyclic subgroup generated by an element \( g \in \mathbb{Z}_p^* \) of order \( Q \). We then define \( H : \langle g \rangle \rightarrow \{0,1\}^h \).

**Key generation** randomly compute a safe RSA modulus \( n = (2p' + 1)(2q' + 1) \), a random element \( u \) in \( \mathbb{Z}_n^* \) and a random element \( y \) in \( \langle g \rangle \subseteq \mathbb{Z}_p^* \). Publish \( (n, u, g, y, P) \).

**Signature** given \( M \in \mathbb{Z}_n \), select \( r \leftarrow \mathbb{Z}_q \), compute \( m = H(g^M y^r \mod P) \), compute \( s = u^m \mod 2p'q' \) and define the signature as \( \sigma = (r, s) \).

**Verification** compute \( m = H(g^M y^r \mod P) \) and check that \( s^m \equiv u \mod n \).

**Twin-GHR signature scheme.** The twinning paradigm was introduced by Naccache, Pointcheval and Stern in [30]. It consists in signing two related messages with a potentially weaker signature scheme. It particularly fits to GHR signatures. The security of the resulting signature scheme can be reduced to the flexible RSA problem. Let \( P \) be an injective function that maps the set \( \{0,1\}^{2l_M} \) into the prime numbers.

**Key generation** picks two safe RSA moduli \( n = (2p' + 1)(2q' + 1) \) and \( N = (2P' + 1)(2Q' + 1) \), select two random elements \( u_1 \in \mathbb{Z}_n^* \) and \( u_2 \in \mathbb{Z}_N^* \). Publish \( (n, N, u_1, u_2) \).

**Signature** given \( M \in \{0,1\}^{l_M} \), select \( \mu_1 \leftarrow \{0,1\}^{2l_M} \) at random, compute \( \mu_2 = (\langle M \| M \rangle ) \oplus \mu_1 \), compute \( s_1 = u_1^{\mu_1} \mod 2p'q' \mod n \), compute \( s_2 = u_2^{\mu_2} \mod 2p'Q' \mod N \), define \( \sigma = (\mu_1, s_1, s_2) \) as the signature on \( M \).

**Verification** check that \( s_1^\mu_1 \equiv u_1 \mod n \) and \( s_2^{\mu_2} \equiv u_2 \mod N \).

**Cramer-Shoup signature scheme.** The Cramer-Shoup signature scheme (CS) is another hash-and-sign signature scheme secure under the strong RSA assumption. One of the advantages of the Cramer-Shoup scheme, compared to the GHR scheme, is that the hash function only needs to be collision-resistant. Its disadvantage is that the reduction is loose. Let \( H : \{0,1\}^* \rightarrow \{0,1\}^h \) be a collision-resistant hash function.

**Key generation** randomly compute a safe RSA modulus \( n = (2p' + 1)(2q' + 1) \), \( e \) an \( (\ell_h + 1) \)-bit prime and \( x, h \) two random elements in \( QR(n) \). Publish \( (n, e, x, h) \).

**Signature** given \( M \in \{0,1\}^{l_M} \),

\[(H)\quad \text{compute } m = H(M)\]

\[(\Sigma)\quad \text{given } m, \text{ choose } c \text{ as a random } (\ell_h + 1)\text{-bit prime, choose } u \text{ at random in } QR(n), \text{ compute } w = u^e h^{-m} \mod n, \text{ compute } v = (x h^{H(w)})^{e^{-1}} \mod 2p'q' \mod n, \text{ define } \sigma = (c, u, v) \text{ as the signature on } M.\]

**Verification** check that \( c \) is an odd \( (\ell_h + 1)\)-bit integer and \( v^c h^{-H(w')} \equiv x \mod n \) with \( w' = u^e h^{-H(m)} \mod n \).