Forgery-Resilience for Digital Signature Schemes

Atefeh Mashatan
LASEC - EPFL
Ecole Polytechnique Fédérale de Lausanne
CH-1015 Lausanne, Switzerland
Atefeh.Mashatan@epfl.ch

Khaled Ouafi
LASEC - EPFL
Ecole Polytechnique Fédérale de Lausanne
CH-1015 Lausanne, Switzerland
Khaled.Ouafi@epfl.ch

ABSTRACT

We introduce the notion of forgery-resilience for digital signature schemes, a new paradigm for digital signature schemes exhibiting desirable legislative properties. It evolves around the idea that, for any message, there can only be a unique *valid* signature, and exponentially many *acceptable* signatures, all but one of them being *spurious*.

This primitive enables a judge to verify whether an alleged forged signature is indeed a forgery. In particular, the scheme considers an adversary who has access to a signing oracle and an oracle that solves a "hard" problem, and who tries to produce a signature that appears to be acceptable from a verifier's point of view. However, a judge can tell apart such a spurious signature from a signature that is produced by an honest signer. This property is referred to as *validatibility*. Moreover, the scheme provides *undeniability* against malicious signers who try to fabricate spurious signatures and deny them later by showing that they are not valid. Last but not least, *trustability* refers to the inability of a malicious judge trying to forge a valid signature.

This notion for signature schemes improves upon the notion of fail-stop signatures in different ways. For example, it is possible to sign more than one messages with forgery-resilient signatures and once a forgery is found, the credibility of a previously signed signature is not under question.

A concrete instance of a forgery-resilient signature scheme is constructed based on the hardness of extracting roots of higher residues, which we show to be equivalent to the factoring assumption. In particular, using collision-free accumulators, we present a tight reduction from malicious signers to adversaries against the factoring problem. Meanwhile, a secure pseudorandom function ensures that no *polynomially-bounded cheating verifier*, who can still solve hard problems, is able to forge valid signatures. Security against malicious judges is based on the RSA assumption.

Keywords

forgery-resilient signatures, digital signatures, fail-stop signatures, higher residuosity, post-quantum cryptography.

1. INTRODUCTION

Digital signature schemes are designed to be the digital equivalent of handwritten signatures, i.e., to authenticate a message or data by binding it with an attachment, called the signature. A signature is thus a proof that the message has been produced by the signer. This concept appeared as early as the invention of public-key cryptography, by Diffie and Hellman [10]. Later, Goldwasser, Micali, and Rivest [13] formally defined a cryptographic digital signature scheme is.

Nowadays, digital signatures have become one of the most important primitives in cryptography and are being deployed in many security protocols. Their significant became even more evident as the volume of online transactions and electronic documents seem to be getting larger than ever. While they were developing, lawmakers have expressed underlying concerns about the legal consequences of these signatures. One of the fundamental legal issues raised is that in case of a successful forgery, how can an innocent signer prove that he or she was not involved in producing the forged signature.

For many cryptographers, this was not a concern because the security of a signature scheme is based on some intractability assumption stating that solving a particular computational problem is hard, for instance. Consequently, it is infeasible to carry out a "successful forgery" by means of cryptanalytic techniques under that particular intractability assumption. This is the case for classical digital signatures where the security is guaranteed by the hardness of factoring [3, 7], discrete logarithm [12, 20], or lattice-based problems. Note that although non-repudiation is offered by classical digital signature schemes, it only deals with the case where a possibly malicious signer has actually signed a signature, but later on refutes it and claims that he or she never produced that signature. Indeed, it does not stop a successful forgery carried out by an adversary.

From a legislative point of view, forgeries do exist, although they are very hard to produce! Hence, dealing with the consequences of possible, but barely occurring, forgeries remains a big concern for legislative authorities and has fostered the felt need for accommodating more powerful adversaries, who can solve a hard problem, such as factoring. Moreover, our widely deployed intractability assumptions are mostly considered based on our current computational capabilities. Whereas lawmakers would ideally want to build their legislative infrastructure more robustly that would last for a much longer period of time and not based on the computational power of computers, something that appears to be changing every decade.

In the case of a successful forgery caused by solving an instance of a hard problem, one could naively say that all previous signatures are still valid but we should stop from signing from now on.

Unfortunately, this approach will not work unless there is a time-stamp attached to the signatures. Otherwise, a signer with an RSA public key published will have no means of disavowing the future forged signatures and will be held responsible for the contents of the messages as the signatures do not contain any information about the date of creation. Another too easy way out of the problem is to discredit *all* the signatures produced, in the past or future, under the intractability assumption of the solved "hard" problem! In this senario, dishonest signers will be given the opportunity to deny the authenticity of messages they have signed, freeing themselves from the potential responsibility their content induces.

Waidner and Pfitzman took a better approach and proposed the notion of fail-stop signatures [21, 22, 32]. A fail-stop signature protects the signers from the possibility that an adversary may be able to forge their signature by means of cryptanalytic techniques. The motivation behind the naming is that a signer is able to detect a forgery and prove it has happened and, as a result, is able to 'stop' the scheme from 'failing', that is, being vulnerable to forgeries. However, fail-stop signatures inherit some limitations that one would ideally want to alleviate. For instance, with a pair of private and public key, a signer can only sign one message and not more. Furthermore, once a forgery is found, the credibility of previous signatures goes under question.

We propose the notion of *forgery-resilient signature schemes* which enable a judge to verify whether an alleged forged signature is indeed a forgery or a valid signature. We then instantiate the first forgery-resilient signature scheme.

1.1 Fail-stop signatures and their limitations

To the best our knowledge, fail-stop signatures, introduced by Waidner and Pfitzman [32] and followed by other researchers [2, 9, 26, 27, 29, 30, 31], account for the only attempt to address the legal issue of digital signature schemes when considering the possibility of forgeries resulting from cryptanalysis. They can be considered as variants of the one-time signature scheme [15] where, given a key, only one message can be signed, although some compression techniques allow us to sign more messages with one pair of keys [2, 21]. In case of a forgery, the adversary has found a solution to a supposedly hard problem. Given this forgery, the legitimate signer can combine his or her signature with the forged signature and produce a solution to the same instance of the hard problem. Hence, the scheme allows a legitimate, polynomially bounded, signer to prove that a forgery has taken place. We note that security in fail-stop signature schemes is information-theoretic, i.e., forgers are assumed to be computationally unbounded.

Fail-stop signatures are instantiated using the notion of bundling homomorphism: a one-way function with the particularity that for any value from its range, there are exponentially many and equally likely elements from its domain that map to this value. To guarantee that the adversary does not output the signer's signature, the set of possible signatures computable by the signer should be small compared to the set of all possible signatures computed by the adversary. On the other hand, a dishonest signer should not be able to produce signatures which can later be proved to be forgeries. Note that unconditional security against both the signer and the verifier cannot be achieved for the same reason that one cannot design a commitment scheme that is both statistically hiding and statistically binding without requiring further interaction between parties. Therefore, the security of the verifiers in fail-stop signature schemes against a dishonest signer can only be based on a computational assumption. Concretely, it assumes that finding collisions on the underlying bundling homomorphism leads to the solution of a hard computational problem.

As was eluded to earlier, fail-stop signatures suffer from a number of limitations. The first limitation concerns the credibility of valid signatures. Although each signer is able to detect and prove forgeries, the credibility of *all* previous signatures comes under question once a forgery has taken place. Moreover, you can no longer sign a message once there is a forgery, even if you have never used your key to sign a message. That is, the system "collapses" once a forgery is realized. Furthermore, the signer has to generate one pair of private key and public key for each message she wants to sign. We refer the interested reader to a complete reference on fail-stop signatures [21, 22].

1.2 Our Contribution

We propose *forgery-resilient signature schemes* that live up to their name and are resilient against forgeries, that is, not only forgeries are detected and stopped (as in fail-stop signature schemes), but also one can continue using the signature scheme, so the credibility of previous signatures is not questioned. Moreover, we do not limit the number of signatures to one per key. Furthermore, once a forgery has occurred, all valid signatures produced by an honest signer are still recognized as such, even if they were produced after the forgery.

Then, we provide a concrete instance of a forgery-resilient signature scheme whose security is based on the assumption that factoring an RSA modulus is hard when considering strong primes. Our instantiation evolves around the idea that, for a message, there can only be a unique *valid* signature, whereas there are exponentially many *acceptable* signatures, all but one of them being *spurious* signatures.

Forgery-resilient signature schemes consider an adversary who has access to two oracles, a signing oracle and an oracle that solves a "hard" problem, and who tries to produce a signature that appears to be acceptable from a verifier's point of view. However, a judge can tell apart such a spurious signature from a signature that is produced by an honest signer. We refer to this ability of the judge as *validatibility* of a signature scheme. Moreover, the scheme considers malicious signers who try to fabricate spurious signatures and refute them later by showing that they are not valid and are results of a forgery. In a forgery-resilient signature scheme, a *polynomially bounded* signer cannot do so, even with the knowledge of spurious signatures produced by a third-party, and we refer to this property as *undeniability* against a malicious signer.

We shift the ability to prove forgeries from the signer to a judge, as in the end the matter has to be settled by a judge. Note that this change of responsibility is the core reason why we can keep on signing even after a forgery. Moreover, previous honestly computed signatures are not questioned even after a forgery is discovered. Hence, considering a judge can be thought of as the price to pay to overcome the limitations of fail-stop signatures. In order to be able to carry out his task, the judge is given a *validation key* for each signer. Although judges, as legal authorities, can be assumed to be honest, we still consider the unlikely scenario that a judge tries to make use of the verification key to forge a valid signature. *Trustability* refers to the notion where a malicious judge is unable to misuse his or her power to forge valid signatures.

It should be noted that malicious judges are not given access to the solver oracle, i.e., we assume no collusion between the judge and the adversary for undeniability. That is because, as it will be discussed later, the verification and validation keys together (almost) uniquely determine the secret key. Therefore, preventing judges from computing that key requires the secret key to be "hidden" by another computational assumption, which can always be broken if the judge accesses the solver oracle.

The idea behind the proof is to allow, for every message, an exponential number of acceptable signatures, i.e., signatures that are correct with respect to the public key. However, we require that only one of those is valid with respect to both the public key and the validation key of the judge. In order to determine whether a forgery has taken place, the judge not only runs the usual validation using the public key of the signer, but also runs a subroutine using the validation key associated to the signer's private key. This later subroutine either asserts the validity of the signature or shows that it is indeed a spurious signature resulted from cryptanalysis.

In fail-stop signatures, the credibility of all signatures is questioned after a forgery because the signer's private key is not bound to any value that correctly maps to the public key. Forgery-resilient signature schemes bind the signer to a single possible private key that will map to the validation key and to the public key. As a result, even after a forgery, the judge can distinguish between a signature produced honestly by the signer from a spurious one.

Fail-stop signature schemes consider unbounded adversaries to be able to deal with forgeries and, consequently, can only sign one message. It is clear that this is not an acceptable restriction when the scheme has to deployed in real systems. We take a different approach and consider adversaries who have access to an oracle that solves instances of hard problems, such as factoring. Indeed, the motivation behind this line of research was to answer the question of what happens if the underlying security assumption, factoring or discrete logarithm for instance, of a digital signature scheme is broken. Hence, we consider adversaries who are polynomially bounded but can solve instances of the factoring problem, for instance. Albeit this setting is close to considering unbounded algorithms in practice, it still allows us to use primitives that stand secure against powerful adversaries when some limitations like on the number of queries that are made [1].

1.3 Overview of our Construction

For our scheme, we assign two functions for the signer that associate the private key to the public key and to the validation key, respectively. Consider the Rabin function [24], $f(x) = x^2 \mod n$, whose invertibility is known to be equivalent to the problem of factoring the modulus n. As a consequence of the Chinese Remainder Thm (CRT), this function maps four integers to a single one. We use a generalization of the Rabin function, $f(x) = x^c \mod n$, for a c that divides the order of the group $\mathbb{Z}_n^\star$. Then, using a result from number theory due to Frobenius [11], we prove that this function actually maps c integers to a single value. Note that although factoring gives the ability to compute c-th roots, the number of them is exponential when the size of c is polynomial. Therefore, even with knowledge of the factorization, a polynomial-time algorithm cannot compute them all.

Our construction uses two instances of this function. In particular, two different factors of the order are employed, one will be used to compute the public key and the other will serve for the computation of the validation key. Since that function maps an exponential number of elements to a single one, recovering the secret key from the public key, even when the factorization of the modulus is known, is highly unlikely. Conceptually, acceptable signatures are constructed from any of those candidate secret keys while the valid one can only be produced using the key that is held by the signer. Nevertheless, in order to distinguish that secret key from the others, we show that only one key can be valid with respect to both the public and validation keys.

One drawback of our scheme is that the verification key's size grows linearly with the number of messages the signer is allowed to sign. This is not a concern in practice since those keys only serve in case of forgery.

We also introduce a stronger security property for accumulator schemes, that is still satisfied by some existing proposals [2], that we name strong collision-freeness. We use such an accumulator scheme to stop malicious signers from disavowing the public randomness included in each signature. Furthermore, we make the signer generate this randomness by means of a pseudorandom function.

1.4 Outline

The rest of the paper is structured as follows. Section 2 provides the necessary background and the factoring intractability assumption. Whereas, Section 3 is devoted to formally defining the notion of forgery-resilient digital signature schemes. We then dedicate Section 4 to our concrete instantiation of the first forgery-resilient digital signature scheme. In Section 5, we focus on analyzing the security of the scheme, prove its correctness, soundness, undeniability, and validatibility. We leave the definition of strongly collision-free accumulators, our construction's proof of trustability, and the efficiency analysis to the Appendices.

2. PRELIMINARIES

This section is devoted to setting the necessary background for this paper. We start by a formal definition of classical digital signatures. We then continue with the definition of pseudorandom functions and finally finish with listing the number theoretic results and assumptions.

Throughout this paper, a function $f(s)$ is said to be polynomial in s if there exists a constant $c \in \mathbb{N}$ such that $f(s)$ is $O(s^c)$. Similarly, $f(s)$ is said to be negligible if, for every $c \in \mathbb{N}$, $f(s)$ is $O(s^{-c})$. For simplicity, we use the notation $f(s) = \mathsf{negl}(s)$ to express that f is negligible in s.

An algorithm is said to be PPT, PPT for short, if its running time is polynomial in the size of its inputs. When the output of an algorithm is fully determined by its inputs, we say that it is deterministic, otherwise, we say that it is probabilistic. Generic PPT algorithms are denoted by $\mathcal{A}_p$ while computationally unbounded adversaries are denoted by $\mathcal{A}_\infty$. If an algorithm $\mathcal{A}$ can query an external $\mathcal{O}$, it will be denoted $\mathcal{A}^{\mathcal{O}}$. An algorithm may also have access to a special oracle, referred to as $\mathcal{O}_{\mathsf{hard}}$, that takes as input an instance of a problem $P \in \mathsf{FNP}$[1], which is assumed to be hard, and returns a solution to this problem.

2.1 Number Theoretic Background

We call a prime number p a strong prime if $p = 2ap' + 1$ and $p' \gg 2a$ is also a prime number. We then consider a PPT algorithm Gen that, on input a security parameter 1^k and two integers a and b of bit-size α, generates two random strong primes $p = 2ap' + 1$ and $q = 2bq' + 1$, such that $\lfloor \log_2 p' \rfloor = \lfloor \log_2 q' \rfloor = \ell_{\mathsf{F}}(k)/2$, where, for any security parameter $k \in \mathbb{N}$, $\ell_{\mathsf{F}}(k)$ denotes a function that represents the recommended (bit-)size of the RSA modulus $n = pq$. Gen outputs $n = pq$ along with p and q. A popular choice to match 80-bit security against factorization is to take $\ell_{\mathsf{F}}(80) = 1024$. However, we take the equations of Lenstra and Verheul as a reference [16] for selecting parameter and key sizes and use $\ell_{\mathsf{F}}(80) = 1184$.

When n is an ℓ-bit long integer, multiplication in $\mathbb{Z}_n^\star$ is performed with time complexity $O(\ell^2)$ while exponentiation modulo

[1]FNP is the class of function problems of the following form: Given an input x and a polynomial-time predicate $P(x, y)$, if there exists a y satisfying $F(x, y)$, then output any such y, otherwise output $\perp$. Also, note that, without loss of generality, we could restrict our oracle to solve problems in NP.

n is performed in time $O(\ell^3)$. $\varphi(\cdot)$ denotes the Euler totient function.

When implied by the context, we omit writing explicitly $\bmod n$ for calculations modulo n.

DEFINITION 1. *Let us consider a PPT algorithm $\mathcal{A}_{\mathsf{FACT}}$ who takes as input $a, b, n = pq$, such that p, q, ${}^{(p-1)}/_{2a}$, and ${}^{(p-1)}/_{2b}$ are prime numbers, and outputs p and q. The factoring assumption states that for any such $\mathcal{A}_{\mathsf{FACT}}$, we have*

$$\forall a, b \in \mathbb{N}^\star : \quad \Pr\Big[(p,q) \leftarrow \mathcal{A}_{\mathsf{FACT}}(1^k, n, a, b) \\ \Big| (n,p,q) \leftarrow \mathsf{Gen}(1^k, a, b)\Big] = \mathsf{negl}(k).$$

The probability is taken over the random coins of Gen *and* $\mathcal{A}_{\mathsf{FACT}}$.

DEFINITION 2. *Let $\mathcal{A}_{\mathsf{RSA}}$ be a polynomial-time algorithm who takes as input $a, b, n = pq$, such that p, q, ${}^{(p-1)}/_{2a}$, and ${}^{(p-1)}/_{2b}$ are prime numbers, and outputs an element from $\mathbb{Z}_N^\star$. We further let e be an integer co-prime with $\varphi(n)$. We say that the RSA assumption holds in $\mathbb{Z}_n^\star$ if*

$$\forall a, b \in \mathbb{N}^\star : \quad \Pr\Big[g \leftarrow \mathcal{A}_{\mathsf{RSA}}(1^k, n, a, b, e, h = g^e) \\ \Big| \begin{array}{c} (n,p,q) \leftarrow \mathsf{Gen}(1^k, a, b) \\ e \in_R \mathbb{Z}_{\varphi(n)}^\star, g \in_R \mathbb{Z}_n^\star \end{array}\Big] = \mathsf{negl}(k)$$

The probability is taken over the random coins of Gen *and* $\mathcal{A}_{\mathsf{FACT}}$ *as well as the random choice of e and g.*

Such RSA moduli were used by Naccache and Stern to construct a public-key cryptosystem [17]. We also refer to the work of Groth [14] that makes a more detailed treatment of the factorization of such numbers by Pollard's rho method [23] and other generic factoring algorithms such as the general number field sieve. Note that the subsequent attack of Coron et al. [8] restricts the set of parameters for which the assumption holds.

In $\mathbb{Z}_n^\star$, every element has a unique a-th root if and only if we have $\mathsf{gcd}(a, \varphi(n)) = 1$. In fact, this ensures the correctness of the RSA cryptosystem. However, when a and $\varphi(n)$ are not coprime, there exists many a-th roots as stated by the following theorem.

THEOREM 1. *If a divides the order of a group, then the number of elements in the group whose order divides a is a multiple of a. If the group is cyclic, then this number is exactly a.*

Note that $\mathbb{Z}_n^\star$ is not cyclic when n is an RSA modulus. However, the next theorem proves that the number of a-th roots of any element in $\mathbb{Z}_n^\star$ is *exactly* equal to a in the class of RSA moduli we consider in this paper, i.e., product of two strong primes.

THEOREM 2. *Let $n = pq$ be an RSA modulus such that p and q are strong prime numbers, i.e., $p = 2ap' + 1$, $q = 2bq' + 1$. If $\mathsf{gcd}(a, q-1) = \mathsf{gcd}(b, p-1) = 1$, then for any $y \in \mathbb{Z}_n^\star$, the equation $g^a = y \bmod n$ has exactly a solutions.*

PROOF. Let $\mathsf{CRT} : \mathbb{Z}_n^\star \to \mathbb{Z}_p^\star \times \mathbb{Z}_q^\star$ denote the isomorphism induced by the CRT. By applying CRT, if g is a solution to the equation $g^a = y \bmod n$, then $(g_p, g_q) = (g \bmod p, g \bmod q)$ is a solution to the two equations

$$(g_p)^a = y \mod p, \tag{1}$$

$$(g_q)^a = y \mod q. \tag{2}$$

On one hand, recalling that $\mathbb{Z}_p^\star$ is cyclic, we can apply Thm 1 to deduce that Equation (1) has exactly a solutions (remember that $\varphi(p) = ap'$). On the other hand, we have that $\mathsf{gcd}(a, \varphi(q)) = 1$ as a is coprime with 2, b, and q' by assumption. Consequently, Equation (2) accepts only one solution. As there exist a tuples of the form (g_p, g_q) satisfies Equations (1) and (2), by applying CRT^{-1}, we deduce that the number of a-th roots of y in $\mathbb{Z}_n^\star$ is exactly a. □

As it is the case for the classical Rabin function, we can rather easily prove that extracting an a-th root of an a-th residue in $\mathbb{Z}_n^\star$ is equivalent to factoring with an even lower reduction gap (for the case of Rabin it is equal to $^1/_2$). Assume an adversary $\mathcal{A}_{\mathsf{root}}$ who, on input an a-th residue $y \in \mathbb{Z}_n^\star$, outputs an x s.t. $x^a = y \pmod n$. We now construct an adversary against factoring $\mathcal{A}_{\mathsf{FACT}}$ that has access to $\mathcal{A}_{\mathsf{root}}$. $\mathcal{A}_{\mathsf{FACT}}$ picks a random $x_1 \in \mathbb{Z}_n^\star$ and computes $y = x_1^a \mod n$. It then calls upon $\mathcal{A}_{\mathsf{root}}$ on input y and gets x_2. When $x_1 \neq x_2$, $\mathcal{A}_{\mathsf{FACT}}$ computes $\mathsf{gcd}(x_1 - x_2, N)$, a non-trivial factor of N. Since the equality $x_1 = x_2$ holds with probability 2^{-a}, we obtain

$$\Pr[\mathcal{A}_{\mathsf{FACT}} \text{ wins}] = (1 - 2^{-a}) \Pr[\mathcal{A}_{\mathsf{root}} \text{ wins}]. \tag{3}$$

2.2 Pseudorandom Functions

Informally, a pseudorandom function (PRF) is a function which cannot be distinguished from a uniformly chosen random function by any algorithm that is allowed a certain number of evaluations of that function.

DEFINITION 3. *Let $F : \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ be a family of functions from $\mathcal{D}$ to $\mathcal{R}$ indexed by keys taken from the set $\mathcal{K}$. Let $R : \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ be a family of random functions from $\mathcal{D}$ to $\mathcal{R}$ indexed by keys taken from the set $\mathcal{K}$. We say that F is a family of pseudorandom functions if it satisfies the following properties.*

1. *There exists a PPT (in λ) sampling algorithm that outputs a random element of $\mathcal{K}$.*

2. *For every $K \in \mathcal{K}$ and every $x \in \mathcal{D}$, $F_K(x)$ is computable by a polynomial-time algorithm (in λ) .*

3. *For every PPT distinguisher $\mathcal{D}$, interacting with a black-box implementing either F_K or R_K, for a randomly chosen K, and outputing a bit b, we have*

$$\left|Pr[\mathcal{D}^{F_K}(1^\lambda) \to 1] - Pr[\mathcal{D}^{R_K}(1^\lambda) \to 1]\right| = \mathsf{negl}(\lambda).$$

The probabilty is taken over the random choice of $K \in \mathcal{K}$ and the random tape of $\mathcal{D}$.

In the sequel, we will restrict ourselves to fixed-input-length PRFs: the number of queries the distinguisher is allowed to make is bounded by a value N. Such a primitive has the advantage of being implementable by an encryption scheme that is provably indistinguishable from the ideal cipher with respect to adversaries limited to N queries to the encryption oracle. For instance, the block cipher KFC [1] is a good candidate for our PRF.

2.3 Classical Digital Signature Schemes

The message space is denoted by $\mathcal{M}$, the signature space by $\mathcal{S}$, the public key space by $\mathcal{PK}$, and the private key space $\mathcal{SK}$. A digital signature scheme is composed of the three following algorithms.

$\mathsf{Keygen}(1^\lambda) \to (pk, sk)$.

A PPT algorithm for generating the pair of public and private keys (pk, sk). It takes as input a security parameter $\lambda \in \mathbb{N}$ written in unary form.

$\mathsf{Sign}(sk, m) \to \sigma$.
A PPT algorithm that takes as input a message $m \in \mathcal{M}$ and a private key sk, and outputs a signature $\sigma \in S$. Note that this algorithm may be either probabilistic or deterministic.

$\mathsf{Verify}(pk, m, \sigma) \to b$.
A deterministic polynomial-time algorithm that outputs 1 when σ is a correct signature of a message m with respect to the public key pk. Otherwise, it outputs 0.

The standard security model for such a scheme is the *existential unforgeability under the adaptive chosen message attack* model introduced by Goldwasser et al. [13]. In summary, it considers an adversary fed with the public key pk who is assumed to have access to a signing Oracle $\mathcal{O}_{\mathsf{Sign}}$ that she can query as many times as the signer can sign. This oracle has the ability to compute the signature of any message chosen by the adversary whose final goal is to produce a valid signature (i.e., one that passes the validation) for a message he did not submit to $\mathcal{O}_{\mathsf{Sign}}$. If no polynomially bounded adversary is able to produce that signature except with probability negligible in λ, then the signature scheme is secure.

3. DEFINITION OF FORGERY-RESILIENT DIGITAL SIGNATURE SCHEMES

Like classical digital signature schemes, a forgery-resilient signature scheme is composed of the classical key generation algorithm, run by an authority (acting as a trusted third party TTP), the signing algorithm, run by the signer, and the acceptance algorithm, run by verifiers. Additionally, in order to show that a forgery has occurred, a judge is given a validation key which is used to assert the validity of a signature. A corresponding validation algorithm has thus to be defined. Considering practical scenarios, one can consider the judge as a TTP.

DEFINITION 4. *A forgery-resilient digital signature scheme is composed of the following five algorithms.*

1. $\mathsf{Setup}(1^\alpha, 1^\lambda) \to \mathsf{param}$*: generates the parameters of the scheme and a trapdoor information that is discarded. This algorithm runs in polynomial-time in its inputs and is executed by the authority.*

2. $\mathsf{KeyGen}(\mathsf{param}, 1^N) \to (sk, pk, vk)$*: is a PPT protocol run by the signer and the judge that generates three keys which will be used to produce N signatures. At the end of the protocol, both parties obtain a public key pk that is later published and the signer obtains a private key sk while the judge obtains the validation key vk. We also assume the existence of two predicates computable in polynomial-time Ψ and Ψ' such that $\Psi(sk, pk, vk)$, resp. $\Psi'(pk, vk)$, returns 1 if and only if the triplet (sk, pk, vk), resp. the pair (pk, vk), is a valid output, resp. partial output, of* KeyGen. *Anytime the predicate Ψ' is not satisfied, the judge outputs $\perp$, meaning that the key generation failed, and no public key is published.*

3. $\mathsf{Sign}(sk, m) \to \sigma$*: computes the signature σ of the message $m \in \mathcal{M}$, the message space in polynomial-time (in λ). When this operation runs correctly, σ is said to be a valid signature of the message m.*

4. $\mathsf{Accept}(pk, m, \sigma) \to b \in \{0, 1\}$*: is an algorithm that yields 1 if the signature σ of the message m is acceptable with respect to pk. Otherwise, it returns 0. A pair (m, σ) that passes the acceptance test is said to be acceptable.*

5. $\mathsf{Validate}(vk, pk, m, \sigma) \to b \in \{0, 1\}$*: returns 0 if the signature is spurious and 1 if it is valid. This algorithm is used by the judge in case of a dispute between parties to determine whether the alleged forgery (m, σ) is a spurious signature or a valid one.*

We require such a scheme to be correct and sound (Properties 1 and 2, resp.). Moreover, forgery resilience induces that the algorithms should satisfy undeniability and validatability (Properties 3 and 4, resp.).

1. ***Completness.*** *The scheme is said to be complete if for every α, λ we have*

$$\Pr\left[\begin{array}{c}\mathsf{Accept}(pk, m, \sigma) = 1,\\ \mathsf{Validate}(vk, pk, m, \sigma) = 1\end{array} \middle| \begin{array}{c}\mathsf{param} \leftarrow \mathsf{Setup}(1^\alpha, 1^\lambda)\\ (sk, pk, vk) \leftarrow \mathsf{KeyGen}(\mathsf{param}, 1^N)\\ m \in_R \mathcal{M}\\ \sigma \leftarrow \mathsf{Sign}(sk, m)\end{array}\right] = 1,$$

where the probability is taken over the random coins of Setup, KeyGen, *and* Sign *(when it is probabilistic) along with the choice of m.*

2. ***Soundness.*** *This property states that every valid signature is acceptable. So, if Σ denotes the space of signatures, we require that for every α, λ*

$$\Pr\left[\begin{array}{c}\mathsf{Accept}(pk, m, \sigma) = 0\\ \wedge\\ \mathsf{Validate}(vk, pk, m, \sigma) = 1\end{array} \middle| \begin{array}{c}\mathsf{param} \leftarrow \mathsf{Setup}(1^\alpha, 1^\lambda)\\ (sk, pk, vk) \leftarrow \mathsf{KeyGen}(\mathsf{param}, 1^N)\\ m \in_R \mathcal{M}, \quad \sigma \in_R \Sigma\end{array}\right] = 0,$$

3. ***Undeniability.*** *We require that no polynomially bounded signer can produce an acceptable signature that is not valid even when she has an oracle access that, on query, returns a pair composed of a random message and a spurious signature for it. Concretely, we denote by $\mathcal{O}_{\mathsf{Forge}}$ an oracle initialized with the public key pk to which the adversary asks for spurious signatures of adaptively chosen messages. Naturally, the adversary is not allowed to output a signature for a message that was submitted to $\mathcal{O}_{\mathsf{Forge}}$. In other words, for all naturals α, λ and every polynomial-time algorithm $\mathcal{A}_p$, we have*

$$\Pr\left[\begin{array}{c}\mathsf{Accept}(pk, m^\star, \sigma^\star) = 1,\\ \mathsf{Validate}(vk, pk, m^\star, \sigma^\star) = 0\end{array} \middle| \begin{array}{c}\mathsf{param} \leftarrow \mathsf{Setup}(1^\alpha, 1^\lambda)\\ (sk, pk, vk) \leftarrow \mathsf{KeyGen}(\mathsf{param}, 1^N)\\ (m^\star, \sigma^\star) \leftarrow \mathcal{A}_p^{\mathcal{O}_{\mathsf{Forge}}}(sk, pk)\end{array}\right] = \mathsf{negl}(\lambda),$$

with the probability taken over the randomness of Setup, KeyGen, *and $\mathcal{A}_p$.*

4. ***Validatability.*** *We say that the scheme is validatable against adversaries able to solve instances of hard problems if for any polynomially bounded adversary who has access to an oracle $\mathcal{O}_{\mathsf{hard}}$ that can solve problems lying in* FNP *(which may be equivalent to an* NP*-Complete problem), and to a signing oracle $\mathcal{O}_{\mathsf{Sign}}$ to which she can submit N adaptively chosen messages, it holds that for all naturals α, λ and every*

PPT algorithm $\mathcal{A}^{\mathcal{O}_{\mathsf{Sign}},\mathcal{O}_{\mathsf{hard}}}$,

$$\left|\Pr\left[\begin{array}{c}\mathsf{Accept}(pk,m^\star,\sigma^\star)=1\\ \mathsf{Validate}(vk,pk,m^\star,\sigma^\star)=1\end{array}\right.\right.$$
$$\left.\left|\begin{array}{c}\mathsf{param}\leftarrow\mathsf{Setup}(1^\alpha,1^\lambda)\\(sk,pk,vk)\leftarrow\mathsf{KeyGen}(\mathsf{param},1^N)\\(m^\star,\sigma^\star)\leftarrow\mathcal{A}^{\mathcal{O}_{\mathsf{Sign}},\mathcal{O}_{\mathsf{hard}}}(pk)\end{array}\right]-2^{-\alpha}\right|$$

is negligible, where the probability is taken over the randomness of all the probabilistic algorithms.

5. ***Trustability.*** *We say that the scheme is trustable if no malicious judge, modeled as a polynomially bounded adversary who has knowledge of both public and validation keys and has access to a signing oracle* $\mathcal{O}_{\mathsf{Sign}}$*, to which she can submit* N *adaptively chosen messages, can output a valid signature on a message not submitted to* $\mathcal{O}_{\mathsf{Sign}}$*. In other words, we require that*

$$\Pr\left[\begin{array}{c}\mathsf{Accept}(pk,m^\star,\sigma^\star)=1\\ \mathsf{Validate}(vk,pk,m^\star,\sigma^\star)=1\end{array}\right.$$
$$\left|\begin{array}{c}\mathsf{param}\leftarrow\mathsf{Setup}(1^\alpha,1^\lambda)\\(sk,pk,vk)\leftarrow\mathsf{KeyGen}(\mathsf{param},1^N)\\(m^\star,\sigma^\star)\leftarrow\mathcal{A}^{\mathcal{O}_{\mathsf{Sign}}}(pk,vk)\end{array}\right]$$

is negligible, where the probability is taken over the randomness of Setup, KeyGen, $\mathcal{O}_{\mathsf{Sign}}$ *(if* Sign *is probabilistic), and* $\mathcal{A}$*.*

In contrast to fail-stop signatures, once a forgery occurs, it does not put honestly produced signatures under question. This is a consequence of the correctness requirement as the judge is the one deciding the validity of the signatures. However, in fail-stop signatures, the proof of forgery was computed by the signer. Moreover, we require stronger properties here as the signer should not be able to produce spurious signatures once a forgery happens. We do not explicitly specify that the signer does not have the validation key vk. Indeed, this is not a requirement for our construction and the signer may be able to compute vk from sk.

Moreover, we could have given the signer the ability to obtain valid forgeries. However, as the undeniability property will only be considered in conjonction with validatibility, the probability that a forger manages to produce a valid signature is negligible. Therefore, when considering secure forgery-resilient schemes, the probability that the signer obtains such a signature is negligible. Moreover, we show in Appendix D that giving the adversary the access to an oracle implementing the Validate algorithm does not affect the security of the scheme.

4. INSTANTIATING A FORGERY-RESILIENT DIGITAL SIGNATURE SCHEME

In this section, we present our concrete proposal of a forgery-resilient digital signature scheme. Throughout this section and the following one, we assume the existence of a strongly collision-free accumulator scheme. In short, an accumulator scheme is said to be strongly collision-free if it remains collision-free even if the adversary obtains the witness of an element that was not in the accumulated set (see Appendix B for a proper definition and an explicit construction.).

$\mathbf{Setup}(1^\alpha,1^\lambda)$.

The setup algorithm first picks an α-bit odd prime number a and an α-bit odd positive integer b such that $\mathsf{gcd}(a,b)=1$. Then, it generates an RSA modulus $n=pq$ by choosing two random prime numbers p' and q' such that $p=2ap'+1$ and $q=2bq'+1$ are also prime numbers. We further require that $\varphi(n)>N$. It also picks an integer $e<\varphi(n)$ such that $\mathsf{gcd}(e,\varphi(n))=1$. In parallel, the algorithm sets the accumulator key for a collision-free accumulator scheme by calling $\mathsf{AccKey}(1^\lambda)\to(K_{\mathsf{acc}},\cdot)$. For the sake of simplicity, we let $a'=ae$ and $b'=be$.

The message space of the resulting scheme is $\mathcal{M}=\mathbb{Z}_{a'}$. The param $(n,a,b,e,K_{\mathsf{acc}})$ is published.

$\mathbf{KeyGen}(\mathsf{param},1^N)$.

The protocol runs as follow:

- The signer picks a random index K for a PRF $F_K:\mathbb{Z}_{\varphi(n)}\to\mathbb{Z}_n^\star$.
- The signer computes $g_i=F_K(i)$, for $i=1\dots N$, and then $y_i=g_i^{a'}$ and $v_i=g_i^{b'}$. He then sends the list of v_i's to the judge. After that, the signer accumulates the set of y_i's, i.e., he calls upon $\mathsf{Acc}_{K_{\mathsf{acc}}}(y_1,\dots,y_N)\to(z,\mathsf{aux})$.
- The signer picks $g\in\mathbb{Z}_n^\star$ at random, computes $y=g^{a'}$ and $v=g^{b'}$, and transmits the last value to the judge.
- The predicate Ψ' for the judge's verification consists of checking whether $y^b=v^a$. If the last equality does not hold, then he outputs $\perp$.

The scheme's keys are set to $sk=(g,K,\mathsf{aux})$, $pk=(a,b,e,n,y,z)$, and $vk=(v,v_1,\dots,v_N)$. At the end of key generation, the signer sets a state counter ctr to 0.

$\mathbf{Sign}(sk,m)$.

The signer first starts with incrementing its state counter by one: $\mathsf{ctr}=\mathsf{ctr}+1$. He then recomputes $g_{\mathsf{ctr}}=F_K(\mathsf{ctr})$ and let $y_{\mathsf{ctr}}=g_{\mathsf{ctr}}^{a'}$. After that, it computes the witness of y_{ctr} by calling $w_{K_{\mathsf{acc}}}(y_{\mathsf{ctr}},z,\mathsf{aux})\to w_{\mathsf{ctr}}$. At last, given the message $m\in\mathcal{M}$, he computes $g_m=g^m\cdot g_{\mathsf{ctr}}^{b+m}$. He releases the pair $(g_m,y_{\mathsf{ctr}},w_{\mathsf{ctr}})$ as his signature.

$\mathbf{Accept}(pk,m,\sigma)$.

Given a signature $\sigma=(g_m,y_{\mathsf{ctr}},w_{\mathsf{ctr}})$ on a message m, the accept algorithm checks whether $\mathsf{EvalAuth}_{K_{\mathsf{acc}}}(y_{\mathsf{ctr}},w_{\mathsf{ctr}},z_{\mathsf{acc}})=1$ and $g_m^{a'}=y^m y_{\mathsf{ctr}}^{b+m}$.

$\mathbf{Validate}(vk,pk,m_\star,\sigma_\star=(g_{m_\star},y_\star))$.

The validation algorithm first calls $\mathsf{Accept}(pk,m_\star,\sigma_\star)$. If the signature is not acceptable, it yields 0, meaning "not a forgery".

Once the signature has been proven to be acceptable, it checks whether there exists an i, $0<i\leq N$, for which the following equality holds $y_\star^b=v_i^a$. If no index is found, then the algorithm stops and outputs 1, which means "forgery". After that, Validate tests the equality $g_m^{b'}=v_i^{b+m}v^m$. If it holds, the algorithm outputs 1, which means "not a forgery", and that the signature is valid. Otherwise, it outputs 0 in which case the signature is spurious.

4.1 Reducing the size of the Validation Key

While we assume that the judge can behave maliciously, we might consider a scenario in which he behaves honestly given its position as a legislative authority. In this settings, we can adapt our construction and sacrifice trustability to obtain a validation key of constant size. That is, we proceed by replacing the list of v_i's by K and updating the Validate algorithm accordingly. It is clear that other properties that trustability are still valid in this context as they do not depend on the validation key.

4.2 Online Judges

Alternatively, we could make the rather constraining assumption that the judge is online and reachable whenever the signer wishes to compute a signature. In this scenario, the construction simplifies in a way that instead of generating the v_i's from a PRF, they could be purely random. Then at each signature, the signer sends the signature along with the corresponding v_i to the judge (Note that the channel is not necessarily secure). The later alerts in case of a detected forgery. A more detailled treatment of this variant will appear in the full version of the paper.

5. SECURITY ANALYSIS

This section is devoted to showing the correctness of the scheme we propose in Section 4 and its security. Due to lack of space, we leave the proof of trustability to Appendix C.

5.1 Correctness and Soundness

THEOREM 3. *The scheme defined in Section 4 is correct and sound. Furthermore, given y_{ctr} there exists exactly one valid signature (g_m, y_{ctr}) for every message m.*

PROOF. First, note that the first step of Validate is to check whether the signature it receives is acceptable and outputs 0 if it is not the case. Hence, every non-acceptable signature is rejected by Validate and the scheme is sound.

We also have to show that any signature produced by the Sign algorithm passes the Accept test. Clearly, from the signature algorithm we have$g_m = g^m \cdot g_{\mathsf{ctr}}^{b+m}$. By raising both sides to the power of $a' = ae$, we obtain

$$g_m^{a'} = \left(g^{a'}\right)^m \cdot \left(g_{\mathsf{ctr}}^{a'}\right)^{b+m} = y^m \cdot y_{\mathsf{ctr}}^{b+m}, \tag{4}$$

which corresponds to the equation of the Accept algorithm.

Now, recall that a valid signature passes the Validate algorithm, i.e., such a signature verifies

$$g_m^{b'} = \left(g^{b'}\right)^m \cdot \left(g_{\mathsf{ctr}}^{b'}\right)^b = v^m \cdot v_{\mathsf{ctr}}^b. \tag{5}$$

We now show that this signature is unique with respect to y_{ctr}. Assume that there exists two distinct valid signatures (g_m, y_{ctr}) and (g'_m, y_{ctr}) for a message m ($g_m \neq g'_m$). From Equations (4) and (5), we deduce that $g_m^a = g_m'^a$ and $g_m^b = g_m'^b$. Let $x = g_m \cdot g_m'^{-1}$. Note that since $g_m \neq g'_m$, we should have $x \neq 1$, $x^{ae} = 1$, and $x^{be} = 1$. This implies that the order of x divides both ae and be. As $\gcd(a, b) = 1$, it must be that $x^e = 1$. Since $\gcd(e, \varphi(n)) = 1$, we deduce $x = 1$ This contradicts the assumption, so there can only exist one g_m. □

5.2 Undeniability: Security against Malicious Signers

The following theorem formalizes the fact that in order to produce a spurious signature, the signer needs to compute the factorization of the RSA modulus.

THEOREM 4. *If the factorization assumption holds (Definition 1) and the accumulator scheme is strongly collision-free, then the scheme described in Section 4 is undeniable.*

PROOF. Let $\mathcal{A}_p$ be a malicious signer that queries $\mathcal{O}_{\mathsf{Forge}}$ to obtain $\Sigma = \left\{(m'_1, (g_{m'_1}, y'_1, w'_1)), \ldots, (m'_\ell, (g_{m_\ell}, y'_\ell, w'_\ell))\right\}$, a list of pairs message and spurious signature. Recall that $\mathcal{A}_p$ is successful when it outputs a spurious, i.e., acceptable and not valid, signature $(g^\star_{m^\star}, y^\star, w^\star)$ on a message $m^\star$ that was not submitted to $\mathcal{O}_{\mathsf{Forge}}$. We use the game methodology to separate different cases depending on the accumulated value contained in the adversary's spurious signature.

- **Game 0.** This is the original game played by $\mathcal{A}_p$
- **Game 1.** In this game, we deal with the event E_1 that an adversary wins by producing an acceptable signature such that $y^\star$ is neither in Σ nor in $\{y_1, \ldots, y_N\}$. That is, we are considering adversaries who, in order to produce an acceptable signature, defeat the security of the accumulator scheme. In other words, the adversary succeeds in producing a pair $(y^\star, w^\star)$ such that $\mathsf{EvalAuth}_{K_{\mathsf{acc}}}(y^\star, w^\star) = 1$.

  We construct an adversary $\mathcal{A}_{\mathsf{acc}}$ against the accumulator scheme as follows. The challenger performs AccKey to generate K_{acc} for the adversary $\mathcal{A}_{\mathsf{acc}}$ who generates N, a, b and executes KeyGen for $\mathcal{A}_p$, hence perfectly emulating $\mathcal{A}_p$'s environment. $\mathcal{A}_{\mathsf{acc}}$ also simulates O_{Forge} by picking a random $g^\star_{\mathsf{ctr}}$, submitting $y^\star_{\mathsf{ctr}} = g^{\star a}_{\mathsf{ctr}}$ to its own oracle for obtaining its witness $w^\star_{\mathsf{ctr}}$, and computing $g^\star_m = g^m \cdot g^{\star b+m}_{\mathsf{ctr}}$. In the end, it returns the pair $(g^\star_m, y^\star_{\mathsf{ctr}}, w^\star_{\mathsf{ctr}})$ as the forged signature. By construction, this pair is an acceptable signature if the witness $w^\star_{\mathsf{ctr}}$ satisfies EvalAuth. At the end of $\mathcal{A}_p$'s execution, $\mathcal{A}_{\mathsf{acc}}$ outputs $(y^\star, w^\star)$ which verifies with AuthEval when $\mathcal{A}_p$ wins, i.e., $\Pr[\mathcal{A}_{\mathsf{acc}} \text{ wins}] = \Pr[\mathcal{A}_p \text{ wins}|E_1]$. However, under the hypothesis that the accumulator is strongly collision-free, the probability above is negligible.
- **Game 2.** In a second step, we take care of the case in which the adversary is outputting a spurious signature for which $y \in \{y'_1, \ldots, y'_\ell\}$ and denote this event E_2. That is, we consider that the malicious signer is making use of the auxiliary information to produce the spurious pair. Let i be the index such that $(y'_i, w'_i) = (y^\star, w^\star)$. Using the equation of the Accept algorithm, we can write $g^{\star a'}_{m^\star} = y^{m^\star} y_i^{m^\star+b}$ and $g^{a'}_{m'_i} = y^{m'_i} y_i^{m'_i+b}$. So, we derive $(g_{m^\star} \cdot g^{-1}_{m'_i})^a = (y \cdot y_i)^{m^\star - m'_i} \pmod n$. Setting $x = g_{m^\star} \cdot g^{-1}_{m'_i}$, $c = y \cdot y_i$, and $\delta = m^\star - m'_i$, we obtain the equation $x^a = c^\delta \pmod n$. Since $\delta \in \mathbb{Z}^\star_{a'}$, $\gcd(a, \delta) = 1$. We can use the extended Euclidean algorithm to obtain two integers β and γ such that $\gamma a + \beta\delta = 1$. Then, we can construct an efficient adversary which is able to extract a-th roots of random elements in $\mathbb{Z}^\star_n$, using the equality $c = c^{\gamma a + \beta\delta} = \left(x^\beta \cdot c^\alpha\right)^a \pmod n$. By (3) we obtain

  $$\begin{aligned}\Pr[\mathcal{A}_p \text{ wins}|E_2] &= \Pr[\mathcal{A}_{\mathsf{root}} \text{ wins}] \\ &= \frac{1}{(1-2^{-a})} \Pr[\mathcal{A}_{\mathsf{FACT}} \text{ wins}].\end{aligned}$$

  Since 2^{-a} is negligible then, under the factorization assumption, the probability of $\mathcal{A}_p$ winning with E_2 occurring is negligible.
- **Game 3.** In this last game, we have $y^\star \in \{y_1, \ldots, y_N\}$. That is, the y contained in the spurious signatures matches one of the ones that are released for the honest signatures, let us assume that it matches y_i. In such a case, we construct an adversary against factoring. Following the factoring assumption, $\mathcal{A}_{\mathsf{FACT}}$, first calls Gen to obtain n to factor and uses KeyGen to generate the key material. It then calls $\mathcal{A}_p$

on input $sk = (g, K, \mathsf{aux})$ and $pk = (y, z)$. It gets the spurious pair $(m^\star, g^\star_{m^\star})$ such that $g^{\star a}_{m^\star} = y^{m^\star} y_i^b$. Then, $\mathcal{A}_{\mathsf{FACT}}$ simulates the Sign algorithm of $\mathcal{A}_p$ with state $\mathsf{ctr} = i$ and computes the signature of the message $m^\star$ under the key $(g, g_i = F_K(i))$. We denote this signature $(g_{m^\star}, y_i)$. As the scheme is correct, this signature is acceptable and valid, in contrast to the spurious pair, so $g^a_{m^\star} = y^{m^\star} y_i^b$. Hence, we obtain $g^a_{m^\star} = g^{\star a}_{m^\star} \pmod n$.

Since only one of the two signatures is valid, we must have $g_{m^\star} \neq g^\star_{m^\star}$. As $\mathsf{gcd}(a, q-1) = 1$, we deduce that $g_{m^\star} = g^\star_{m^\star} \pmod q$. Hence, $\mathcal{A}_{\mathsf{FACT}}$ recovers the factorization of n by computing $q = \mathsf{gcd}(g_{m^\star} - g^\star_{m^\star}, n)$ so we have

$$\Pr[\mathcal{A}_p \text{ wins}] = \Pr[\mathcal{A}_{\mathsf{FACT}} \text{ wins}],$$

which is negligible under the factorization assumption. □

5.3 Validatibility: Security against Powerful Forgers

We now formalize the security against a polynomially-bounded forger who can break any hardness assumption.

The idea is the following. We consider a variation of the scheme in which the signer generates each g_i randomly and independently from the others. We then show that in this scheme no adversary can produce a signature that turns to be valid except with probability less than or equal to $2^{-\alpha}$. Then, we come back to our original scheme in which the g_i's are generated pseudorandomly. If there exists an adversary who is able to produce a valid signature with non-negligible probability then she can be transformed into a distinguisher for the PRF. The following lemma formalizes the fact that if the values $g_1, \ldots, g_n$ of the scheme described in Section 4 are chosen randomly, then any (even computationally unbounded) adversary is not able to produce a valid signature.

LEMMA 1. *Assume that the* KeyGen *algorithm of Section 4 is modified as follows. The values* $g_1, \ldots, g_N$ *are chosen following a uniform distribution and independently from each other. Then, no adversary, even computationally unbounded, can produce a valid signature, except with probability* $2^{-\alpha}$.

Note that this lemma is stated considering a computationally unbounded adversary. Obviously, it remains true when we consider a polynomially bounded adversary with access to problem solvers.

PROOF. Without loss of generality, we assume that the adversary makes N queries to the signing Oracle. First note that the adversary can recover the factorization of N_{acc} so that she can compute by herself a such that $x^{Prime(y)} = a \pmod N)_{\mathsf{acc}}$. Hence, the only useful information that the signer reveals about sk is pk and N valid signatures $((g_{m_1}, y_1), \ldots, (g_{m_N}, y_N))$ on the messages $m_1, \ldots m_N$. Using only one signature, say the i-th, the set of possible keys given this information is

$$\begin{aligned}\mathcal{SK}_i &= \left\{(g, g_i) \in \mathbb{Z}_n^{\star 2} : g^a = y, g_i^a = y_i, g_{m_i} = g^m \cdot g_i^b\right\} \\ &= \left\{(g, g_i) \in \mathbb{Z}_n^{\star 2} : g^a = y, g_i^a = y_i, g_i^b = g_{m_i} \cdot g^{-m}\right\}\end{aligned}$$

We note that, by the last two equations of the set $\mathcal{SK}_i$, the second element of the tuple, g_i, is completely determined by the choice of g. Hence, the cardinality of $\mathcal{SK}_i$ is equal to the number of solutions to the equation $g^a = y$. As a consequence of Thm 2, we have that $|\mathcal{SK}_i| = a > 2^\alpha$. Note that the description of $\mathcal{SK}_i$ only depends on the choice of g so the adversary can discard all the other $\mathcal{SK}_j$'s for $j \neq i$.

We must now find out how many of these acceptable keys output a valid signature for a message $m_\star$. As y_i, $g_{m_\star}$, and K are fixed, the valid signature is unique (due to Thm 3). We thus need to analyze the number of pairs (g, g_i) that produce that valid signature. By definition of valid signatures, such a pair should satisfy the equalities $g_{m_\star} = g^{m_\star} \cdot g_i^b$, $g_i^a = y_i$, and $g_i^b = F_K(i)$. Note that the first equality can be rewritten as $g = \left(g_{m_\star} \cdot g_i^{-b}\right)^{m_\star^{-1}}$.

As $\mathsf{gcd}(a, b) = 1$, the last two equations imply that there is only one solution for g_i. Once this g_i is found, there exists only one possible g that satisfies the first equation. So, there exists only one key pair that produces valid signatures. Since there are at least 2^α acceptable keys and one valid key, uniformly distributed among the set of acceptable keys (g is randomly chosen by the signer), the probability that an adversary recovers the valid key is bounded by $2^{-\alpha}$. □

Now, we are ready to prove the main theorem regarding the validatibility property of our scheme.

THEOREM 5. *The construction proposed in Section 4 is validatable against polynomial-time adversaries who have access to an oracle to solve the factorization problem.*

PROOF. Let us consider the adversary $\mathcal{A}^{\mathcal{O}_{\mathsf{hard}}}$ who tries to produce a valid signature in one of the following two settings.

- Game_0. In the first setting, the adversary is interacting with the real scheme.
- Game_1. In this case, the PRF F_K of the signer is replaced by picking random elements from $\mathbb{Z}_n^\star$.

We consider an algorithm $\mathcal{D}$ who only outputs 1 when $\mathcal{A}_{\mathcal{O}}$ wins the game. Otherwise, it outputs 0. As the PRF is indistinguishable from a random function, we have

$$\left|\Pr[\mathcal{D}(\mathcal{A}_{\mathcal{O}}^{\mathsf{Game}_0}) \to 1] - \Pr[\mathcal{D}(\mathcal{A}_{\mathcal{O}}^{\mathsf{Game}_1}) \to 1]\right| = \mathsf{negl}(\lambda).$$

By Lemma 1, we know that $\Pr[\mathcal{A}_{\mathcal{O}}^{\mathsf{Game}_1}] \leq 2^{-\alpha}$. Therefore, it must be that $|\Pr[\mathcal{A}_{\mathcal{O}}^{\mathsf{Game}_0}] - 2^{-\alpha}| = \mathsf{negl}(\lambda)$. The scheme proposed in Section 4 is, thus, validatable. □

Acknowledgements

The work described in this paper has been supported, in part, by the European Commission through the ICT Programme under contract ICT-2007-216646 ECRYPT II, and by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center of the SNF under grant number 5005-67322.

We also would like to thank the anonymous reviewers for helping us to improve this paper.

6. REFERENCES

[1] Thomas Baignères and Matthieu Finiasz. KFC - the krazy feistel cipher. In *ASIACRYPT 2006*, pages 380–395, 2006.

[2] Niko Barić and Birgit Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In *EUROCRYPT 1997*, pages 480–494.

[3] Mihir Bellare and Phillip Rogaway. The exact security of digital signatures - how to sign with RSA and Rabin. In *EUROCRYPT 1996*, pages 399–416.

[4] Josh Cohen Benaloh and Michael de Mare. One-way accumulators: A decentralized alternative to digital sinatures (extended abstract). In *EUROCRYPT 1993*, pages 274–285.

[5] Jan Camenisch, Markulf Kohlweiss, and Claudio Soriente. An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In *PKC 2009*, pages 481–500, 2009.

[6] Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *CRYPTO 2002*, pages 61–76.

[7] Jean-Sébastien Coron. On the exact security of full domain hash. In *CRYPTO 2000*, pages 229–235.

[8] Jean-Sébastien Coron, Antoine Joux, Avradip Mandal, David Naccache, and Mehdi Tibouchi. Cryptanalysis of the rsa subgroup assumption from tcc 2005. In *PKC 2011. Proceedings*, volume 6571 of *Lecture Notes in Computer Science*, pages 147–155. Springer, 2011.

[9] Ivan Damgård, Torben P. Pedersen, and Birgit Pfitzmann. On the existence of statistically hiding bit commitment schemes and fail-stop signatures. In *CRYPTO 1993*, pages 250–265.

[10] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22:644–654, 1976.

[11] Georg Frobenius. *Über einen Fundamentalsatz der Gruppentheorie, II.* Sitzungsberichte der Preussischen Akademie Weissenstein, 1907.

[12] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *CRYPTO 1984*, pages 10–18.

[13] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.

[14] Jens Groth. Cryptography in subgroups of $\mathrm{z_n}$. In *TCC 2005*, pages 50–65.

[15] Leslie Lamport. Constructing digital signatures from a one-way function. Technical report, SRI International Computer Science Laboratory, 1979.

[16] Arjen K. Lenstra and Eric R. Verheul. Selecting cryptographic key sizes. *J. Cryptology*, 14(4):255–293, 2001.

[17] David Naccache and Jacques Stern. A new public key cryptosystem based on higher residues. In *ACM Conference on Computer and Communications Security*, pages 59–66, 1998.

[18] Lan Nguyen. Accumulators from bilinear pairings and applications. In *CT-RSA 2005*, volume 3376, pages 275–292, 2005.

[19] Kaisa Nyberg. Fast accumulated hashing. In *FSE 1996*, volume 1039, pages 83–87, 1996.

[20] National Institute of Standards and Technology. *FIPS PUB 186-2: Digital Signature Standard (DSS)*. NIST, 2000.

[21] Torben P. Pedersen and Birgit Pfitzmann. Fail-stop signatures. *SIAM J. Comput.*, 26(2):291–330, 1997.

[22] Birgit Pfitzmann. *Digital Signature Schemes, General Framework and Fail-Stop Signatures*, volume 1100. 1996.

[23] John M. Pollard. A monte carlo method for factorization. *BIT Numerical Mathematics*, 15(3):331–334, 1975.

[24] Michael O. Rabin. Digitalized signatures and public-key functions as intractable as factorization. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, 1979.

[25] Ron Rivest and Robert Silverman. Are 19strong' primes needed for RSA? Cryptology ePrint Archive, Report 2001/007, 2001. http://eprint.iacr.org/.

[26] Reihaneh Safavi-Naini, Willy Susilo, and Huaxiong Wang. An efficient construction for fail-stop signature for long messages. *Journal of Information Science and Engineering*, 17(6):879–898, 2001.

[27] Katja Schmidt-Samoa. Factorization-based fail-stop signatures revisited. In *ICICS 2004*, pages 118–131.

[28] Adi Shamir. On the generation of cryptographically strong pseudorandom sequences. *ACM Trans. Comput. Syst.*, 1(1):38–44, 1983.

[29] Willy Susilo, Reihaneh Safavi-Naini, and Josef Pieprzyk. RSA-based fail-stop signature schemes. In *ICPP Workshops*, pages 161–166, 1999.

[30] Eugène van Heijst, Torben P. Pedersen, and Birgit Pfitzmann. New constructions of fail-stop signatures and lower bounds (extended abstract). In *CRYPTO 1992*, pages 15–30.

[31] Eugène van Heyst and Torben P. Pedersen. How to make efficient fail-stop signatures. In *EUROCRYPT 1992*, pages 366–377.

[32] Michael Waidner and Birgit Pfitzmann. The dining cryptographers in the disco - underconditional sender and recipient untraceability with computationally secure serviceability (abstract). In *EUROCRYPT 1989*, page 690.

APPENDIX

A. EFFICIENCY ANALYSIS

Among the five algorithms of our scheme, the setup is certainly the most expensive operation to perform. Generating strong primes of the specified form mentioned earlier is done in time roughly $O\left(\ell_{\mathsf{F}}^4(\lambda) + (1+\alpha+\ell_{\mathsf{F}}(\lambda))^4\right)$. Note that this complexity is in fact very close to the key generation of RSA with strong primes [25] and is only done once.

Using some simple optimization techniques, we can drastically reduce the complexity of modular exponentiations to the power of a and b by choosing values for a and b with very low Hamming weight. The security proof of our scheme is based on the size of a and b and not their Hamming weight. Considering the today's factoring algorithms, assuming low Hamming weight for a and b does not help the adversary as, in the end, the size of p and q does not change. Assuming that this choice of a and b does not weaken the security, one can use the standard 'square-and-multiply' technique and reduce all exponentiations to a, resp. b, to $\alpha + \mathsf{Hwt}(a)$, resp. $\alpha + \mathsf{Hwt}(b)$, modular multiplications, where Hwt denotes the Hamming weight function.

Using this technique, we claim that signing or accepting can be performed using one exponentiation, g^m or y^m, resp., and a small number of multiplications. Namely, g_{ctr}^a and g_{ctr}^b can both be computed exclusively using modular multiplications during signing. The same applies to the Accept algorithm concerning the computations of g_m^a and g_m^b. As for the key generation, it essentially requires the signer to pick a key for the PRF, a random element, N invocations of the PRF, and $N+1$ modular exponentiations to the power of b. Using the same trick as above, all these modular exponentiations can be turned into modular multiplications.

B. CRYPTOGRAPHIC ACCUMULATORS

Accumulators were introduced by Benaloh and de Mare [4] as an alternative to digital signatures for authenticating a predetermined set of elements Y. Basically, an accumulator scheme is an algorithm to combine a large set of values into one, short accumulator z of constant size , such that there is a short witness that a given value was indeed incorporated into the accumulator. The initial proposal of Benaloh and de Mare was to use a quasi-commutative hash function, i.e., a one-way hash function h that satisfies $h(h(x, y_1), y_2) = h(h(x, y_2), y_1)$. Barić and Pfitzman [2] later generalized this definition to not require quasi-commutativity and provided a stronger notion of security called collision-freeness (Note that almost all accumulator schemes are still based on a quasi-commutative function). While in one-way accumulators, the verifier is required to be unable to produce a witness for an element that is not included in the accumulator's set Y correctly verifies with the accumulator, collision-free accumulators consider this impossibility from the prover point of view, i.e., the adversary is granted the ability to freely choose the set Y.

In this work, we will need an even stronger security notion for collision-free accumulators. That is, we consider the existence of an algorithm that given an eventual trapdoor information, the authenticator z, and an element y not in the set Y that generated z, produces a witness that makes y being accepted by the verification algorithm as being accumulated. In these settings, the adversary's goal, who is polynomial, is to produce a witness for a value y that is not in the set Y and has not been queried to the oracle that satisfies the verification algorithm. The scheme is secure if the winning probability of any such adversary is negligible and we call an accumulator satisfying this notion strongly collision-free. Note that since unconditionally collision-free accumulators such as [19] do not admit the existence of an (exponential) algorithm for producing a witness of a value that was not accumulated in a set, the oracle our adversary accesses for strong collision-freeness cannot exist. Hence, in this case collision-freeness and strong collision-freeness are equivalent. For this reason, the definition that follows only deals with computationally secure accumulators.

DEFINITION 5. *An accumulator scheme is defined through the first four algorithms. The fifth one is defined for strongly collision-free accumulator schemes.*

- $\mathsf{AccKey}(1^\lambda) \to (K_{\mathsf{acc}}, t)$. *A PPT algorithm that, on input a security parameter λ and outputs an accumulator key K_{acc} and a trapdoor information t. For the sake of simplicity, we assume that the accumulator key uniquely caracterizes a set Y.*
- $\mathsf{Acc}_{K_{\mathsf{acc}}}(Y_w = \{y_1, \dots, y_N\}) \to (z_{\mathsf{acc}}, \mathsf{aux})$. *The algorithm used to accumulate a set Y_w of N values from Y using the accumulator key K_{acc}. After its execution, the value z_{Acc} is made public. This algorithm is deterministic and runs in polynomial-time.*
- $w_{K_{\mathsf{acc}}}(y_i, z_{\mathsf{acc}}, \mathsf{aux}) \to w_i$. *This is the deterministic polynomial-time algorithm used to generate the witness w_i for y_i. It outputs $\perp$, when $y_i \notin Y_w$.*
- $\mathsf{EvalAuth}_{K_{\mathsf{acc}}}(y_i, w_i, z_{\mathsf{acc}}) \to \{0, 1\}$. $\mathsf{EvalAuth}$ *is a polynomial-time algorithm that asserts the authenticity of y_i with respect to z_{acc}. As such, it takes as input the accumulator key K_{acc} and the value to authenticate y_i along with its witness w_i and outputs a bit.*
- $\mathsf{Extract}_{t,K_{\mathsf{acc}}}(z_{\mathsf{acc}}, y^\star) \to w^\star$. *A special polynomial-time algorithm for producing a witness $w^\star$ for $y^\star \in Y$. That witness has to be correct in the sense that* $\mathsf{EvalAuth}_{K_{\mathsf{acc}}}(y^\star, w^\star, z_{\mathsf{acc}}) = 1$

A strongly collision-free accumulator has to satisfy the following two properties.

- ***Correctness.*** *This notion captures the requirement that every value that was accumulated in a set can be authenticated. More formally, we require that for every natural number λ and N,*

$$\Pr\left[\exists i : \mathsf{EvalAuth}_{K_{\mathsf{acc}}}(y_i, w_i, z_{\mathsf{acc}}) = 0 \left| \begin{array}{c} (K_{\mathsf{acc}}, \cdot) \leftarrow \mathsf{AccKey}(1^\lambda) \\ Y_w = \{y_1, \dots, y_N\} \subset_R Y \\ (z_{\mathsf{acc}}, \mathsf{aux}) \leftarrow \mathsf{Acc}_{K_{\mathsf{acc}}}(Y_w) \\ w_i \leftarrow w_{K_{\mathsf{acc}}}(y_i, z_{\mathsf{acc}}, \mathsf{aux}) \end{array}\right.\right] = 0,$$

  with the probability being taken over the choice of Y_w and the randomness of AccKey.
- ***Strong Collision-freeness.*** *We consider a polynomial-time algorithm $\mathcal{A}$ that has access to an oracle which implements* $\mathsf{Extract}_{t,K_{\mathsf{acc}}}(\cdot,\cdot)$ *and denote by Y_E the list of queries $\mathcal{A}$ submits to its oracle. The accumulator scheme is strongly collision-free if for every such algorithm $\mathcal{A}$, it holds that for every natural numbers λ and N*

$$\Pr\left[\begin{array}{c} \mathsf{EvalAuth}_{K_{\mathsf{acc}}}(y', w', z_{\mathsf{acc}}) = 1 \\ y' \notin Y_w \cup Y_E \end{array} \left| \begin{array}{c} (K_{\mathsf{acc}}, t) \leftarrow \mathsf{AccKey}(1^\lambda) \\ (Y_w, y', w') \leftarrow \mathcal{A}^{\mathsf{Extract}_{t,K_{\mathsf{acc}}}(\cdot,\cdot)}(K_{\mathsf{acc}}) \\ (z_{\mathsf{acc}}, \mathsf{aux}) \leftarrow \mathsf{Acc}_{K_{\mathsf{acc}}}(Y_w) \end{array}\right.\right] = \mathsf{negl}(\lambda).$$

  Here, the probability is taken over the random coins of $\mathcal{A}$ and AccKey.

Several constructions of collision-free accumulators were proposed and it turns out that most of them can be easily shown to be strongly collision-free. We mention in particular the collision-free accumulator based on the Strong RSA assumption due to Barić and Pfitzman [2] which can easily be shown to be strongly collision-free using a classical trick by Shamir [28] regarding the difficulty of computing the e-th root of an element from $\mathbb{Z}_n^\star$ given an e'-th root of the same element when e and e' are co-prime. After the introduction of dynamic accumulators by Camenisch and Lysyanskaya [6], several constructions of accumulators from bilinear pairings such as [18, 5] were proposed. Writing a proof that these commitments are also strongly collision-free is not very difficult (In [5], when unnecessary dynamic addition of elements is discarded, an adversary against the strong collision-freeness of the scheme reduces to an attacker against the underlying signature scheme). Due to lack of space, giving a full proof for the strong collision-freeness of these accumulators is left to the full version of the paper. We further recall that Nyberg's accumulator scheme [19] is strongly collision-free as it is unconditionally collision-free.

C. TRUSTABILITY AGAINST MALICIOUS JUDGES

The only property left to demonstrate relates to malicious judges whose goal is to produce a valid signature on a message that was not signed by the signer.

THEOREM 6. *If the accumulator is strongly collision-free, then the construction of Section 4 is validatable with respect to the judge under the RSA assumption.*

Due to space constrains, we omit to give a full formal proof of this result. It shall however appear in the final version of the paper.

PROOF. We let $\mathcal{A}$ be a polynomial-time algorithm that takes as input a public key pk and a validation key vk to output a valid signature $\sigma_\star = (g_{m_\star}, y_\star, w_\star)$ on a message $m_\star$. Since a valid signature is unique, as states Thm 3, it must be that

$$g_{m_\star} = g^{m_\star} y_\star^{b+m_\star} \tag{6}$$

We now consider an adversary $\mathcal{A}_{\mathsf{RSA}}$ against the RSA problem. As described in Definition 2, $\mathcal{A}_{\mathsf{RSA}}$ takes as input the modulus n along with the three integers a, b (that divide $\varphi(n)$), and e (that is co-prime with $\varphi(n)$) and an element $h \in \mathbb{Z}_n^\star$ which e-th root has to be recovered. Given the algorithm $\mathcal{A}$ against the forgery-resilient scheme, we construct $\mathcal{A}_{\mathsf{RSA}}$ as follow. First, $\mathcal{A}_{\mathsf{RSA}}$ computes $y = h^a$ and $v = h^b$. It then generates the necessary material for the accumulator scheme, i.e., it uses AccKey to get the accumulator key K_{Acc} and picks a random K for the PRF F. After that, it follows on the scheme's KeyGen algorithm and computes the values to accumulate. In the end, $\mathcal{A}_{\mathsf{RSA}}$ obtains a valid pair of public and verification keys that are given as input to $\mathcal{A}$. After $\mathcal{A}$'s execution, $\mathcal{A}_{\mathsf{RSA}}$ gets the signature $(g_{m_\star}, y_\star, w_\star)$ and the corresponding message $m_\star$.

As for the proof of Thm 5, we can use the strong collision-freeness of the accumulator scheme to rule out the case where $y_\star \notin \{y_1, \ldots, y_N\}$. Therefore, there must exist an index $i < N$ such that $y_\star = F_K^{a'}(i)$. Once $\mathcal{A}_{\mathsf{RSA}}$ recovers this index, it uses Equation (6) to retrieve the value of $g^{m_\star}$. Using the fact that $m_\star$ and e are coprime, $\mathcal{A}_{\mathsf{RSA}}$ computes g using the extended Eucledian algorithm. □

D. ON THE USE OF A VALIDATE ORACLE

One might wonder what happens if the powerful adversary against the validatibility property of a forgery-resilient signature scheme does have access to a $\mathsf{Validate}$ oracle, i.e., she can ask the judge to assert the validity of a signature. We prove that this does not give a significant advantage.

THEOREM 7 (VALIDATIBILITY WITH A $\mathsf{Validate}$ ORACLE). *Let a forgery-resilient scheme with security parameters α and λ. If $2^{-\alpha}$ is negligible in λ then no polynomial-time adversary, who, along with the signing and solver oracles, is augmented by an access to a* $\mathsf{Validate}$ *oracle, defeats the validatibility property of the scheme with a non-negligible probability.*

PROOF. *(Sketch)* Assume an adversary who has an access to a $\mathsf{Validate}$ oracle against the validatibility of the scheme. We construct a simulator for that oracle as follows. If the adversary submits a signature obtained from the signing oracle, then the simulator outputs 1. In all other cases, the simulator outputs 0. Due to the correctness of the scheme, $\mathsf{Validate}$ always outputs 1 when the simulator outputs 1. In the other case, i.e., when $\mathsf{Validate}$ outputs 1 and the simulator outputs 0, we obtain an adversary who successfully forges a valid signature. However, the validatibility of the scheme upper-bounds the probability that an adversary produces a valid signature by $2^{-\alpha} + \mathsf{negl}(\lambda)$, which is negligible. Hence, the probability that the answers from the Validate oracle and the simulator differ is negligible. Using a simple hybrid argument, we can show that every adversary with an access to a $\mathsf{Validate}$ oracle reduces to one that does not have such an oracle at her disposal. □

A similar result can be obtained for signers. Since they do not necessarily know the validation key, it is legitimate to study the eventual advantage a malicious signer acquires through the operation of querying the judge for the validity of a signature. Using a similar reasoning used for the case of verifiers, we derive the following theorem.

THEOREM 8. *Consider a forgery-resilient scheme with security parameters α and λ. If there exists a polynomial-time adversary (in λ), who, along with the secret and public keys, has the ability to query a* $\mathsf{Validate}$ *oracle, and defeats the undeniability property of the scheme with a non-negligible probability, then we can construct another polynomial-time adversary who is successful in defeating the undeniability property of the scheme with a non-negligible probability without having to query the* $\mathsf{Validate}$ *oracle.*

PROOF. *(Sketch)* In order to show that the signers get no advantage by accessing a $\mathsf{Validate}$ oracle, we can show that the result of the latter can be predicted by constructing a simulator for that oracle. The simulation is carried out as follows: when the signer submits a pair (m, σ) to the $\mathsf{Validate}$ oracle, the simulator returns whether the pair is acceptable (since the signer gets the public key as input, it is known to the simulator). That is, the simulator answers 1 when the pair is aceptable and 0 otherwise. For the proof, we need to consider the cases where the outputs of $\mathsf{Validate}$ and the simulator differ. For the first case, when the $\mathsf{Validate}$ oracle outputs 1, i.e., that the signature is valid, then, the simulator never outputs 0, i.e., the signature is not acceptable since every valid signature is acceptable due to the soundness property. For the other case, we use the undeniability property (with an empty list as auxiliary input) to limit the probability that the adversary outputs an acceptable, but not valid, signature to a negligible value in λ. Again, the simulator and the $\mathsf{Validate}$ oracle produce computationally indistinguishable distributions and as before, a hybrid argument can show that the adversary who behaves like the original one but uses the simulator instead of the $\mathsf{Validate}$ oracle produces an indistinguishable output. In other words, this adversary wins the undeniability experiment with a probability essentially unchanged, i.e., the difference between the two winning probabilities is negligible. □

D.1 On Existential Unforgeability of Forgery-Resilient Signatures

When defining the security properties of forgery-resilient signatures, we have not explicitly demanded that it must be difficult for a polynomially-bounded forger to construct acceptable signatures. Clearly, a signature scheme is useless if it is easy to make forgeries, even if they can be later repudiated. Actually, we can show that the proposed security definitions imply that forging a signature of any message, under a chosen message attack, is hard for polynomially bounded enemies. As a corollary, any forgery-resilient signature scheme can be transformed into a secure classical digital signature scheme.

THEOREM 9. *Assuming that $2^{-\alpha}$ is negligible in λ, any correct, undeniable, and validatable, forgery-resilient signature scheme can be transformed into an equally efficient classical digital signature scheme secure against existential forgery under chosen-message attacks in which the* Accept *algorithm plays the role of the classical* Verify *algorithm and the validation key vk is discarded.*

PROOF. We start from a successful existential forger $\mathcal{A}$ who takes the public-key as input and has access to a signing oracle $\mathcal{O}_{\mathsf{Sign}}$ for which we consider two winning cases. In the first one, the adversary wins by producing a valid, and hence acceptable, signature while in the second case the forgery is only an acceptable signature. We show that both winning probabilities are negligible.

For the first case, let us define the adversary $\mathcal{A}_p$ against the undeniability property of the forgery-resilient signature scheme as follows. On input the key pair (sk, pk), the adversary runs $\mathcal{A}$ with input pk and uses sk to build the signing oracle $\mathcal{O}_{\mathsf{Sign}}$ that answers $\mathcal{A}$'s signing queries. $\mathcal{A}_p$ finally outputs the same output as $\mathcal{A}$, namely $(m^\star, \sigma^\star)$. Clearly, the winning probability of $\mathcal{A}_p$ equals $\mathcal{A}$'s, which is negligible due to undeniability.

For the second case, we consider an adversary $\mathcal{A}^{\mathcal{O}}$ against the validatibility property of the forgery-resilient signature scheme. Recalling that $\mathcal{A}_{\mathcal{O}}$ has access to a signing oracle $\mathcal{O}_{\mathsf{Sign}}$, we define it as follows: On input pk, it runs $\mathcal{A}$ with input pk and forwards any of its signing query to $\mathcal{O}_{\mathsf{Sign}}$. As before, $\mathcal{A}_{\mathcal{O}}$ outputs the same output as $\mathcal{A}$ (Note that $\mathcal{A}_{\mathcal{O}}$ does not query $\mathcal{O}$ since $\mathcal{A}$ does not have access to it.). At last, validatibiliy ensures that the probability of $\mathcal{A}_{\mathcal{O}}$ winning is negligible (recall that $2^{-\alpha}$ is negligible).

We conclude that $p_{\mathcal{A}} = \mathsf{negl}(\lambda)$. Hence the scheme is secure against existential forgery with chosen message attack. $\square$