Proceedings of the 8th ACM Conference on Computer and Communications Security. Pages ??–??.
(november 5 – 8, 2001, Philadelphia, Pennsylvania, USA)

Twin Signatures: an Alternative to the Hash-and-Sign Paradigm

David Naccache[1], David Pointcheval[2], and Jacques Stern[2]

[1] Gemplus Card International – 34, rue Guynemer – F–92447 Issy-les-Moulineaux, France
http://www.gemplus.com/smart – david.naccache@gemplus.com
[2] École Normale Supérieure – 45, rue d'Ulm – F–75230 Paris cedex 05, France
http://www.di.ens.fr/~{pointche,stern} – {david.pointcheval,jacques.stern}@ens.fr

Abstract. This paper introduces a simple alternative to the hash-and-sign paradigm, from the security point of view but for signing short messages, called *twinning*. A twin signature is obtained by signing twice a short message by a signature scheme. Analysis of the concept in different settings yields the following results:

– We prove that no generic algorithm can efficiently forge a twin DSA signature. Although generic algorithms offer a less stringent form of security than computational reductions in the standard model, such successful proofs still produce positive evidence in favor of the correctness of the new paradigm.
– We prove in standard model an equivalence between the hardness of producing existential forgeries (even under adaptively chosen message attacks) of a twin version of a signature scheme proposed by Gennaro, Halevi and Rabin and the Flexible RSA Problem.

We consequently regard twinning as an interesting alternative to hash functions for eradicating existential forgery in signature schemes.

Keywords: Digital Signatures, Provable Security, Discrete Logarithm, Generic Model, Flexible RSA Problem, Standard Model.

1 Introduction

The well-known *hash and sign* paradigm has two distinct goals: increasing *performance* by reducing the size of the signed message and improving *security* by preventing existential forgeries. As a corollary, hashing remains mandatory even for short messages.

From the conceptual standpoint, the use of hash functions comes at the cost of extra assumptions such as the conjecture that for all practical purposes, concrete functions can be identified with ideal black boxes [3] or that under certain circumstances (black box groups [15, 21]) a new group element must necessarily come from the addition of two *already known* elements. In some settings [11] both models are even used simultaneously.

This paper investigates a simple substitute to hashing that we call *twinning*. A twin signature is obtained by signing twice the same (short) raw message by a probabilistic signature scheme, or two probabilistically related messages.

We believe that this simple paradigm is powerful enough to eradicate existential forgery in a variety of contexts. To support this claim, we show that no generic algorithm can efficiently forge a twin DSA signature and prove that for a twin variant of a signature scheme proposed by Gennaro, Halevi and Rabin [8] (hereafter GHR) existential forgery, even under an adaptively chosen-message attack, is equivalent to the Flexible RSA Problem [5] in the standard model.

2 Digital Signature Schemes

Let us begin with a quick review of definitions and security notions for digital signatures. Digital signature schemes are the electronic version of handwritten signatures for digital documents: a user's signature on a message m is a string which depends on m, on public and secret data specific to the user and–possibly–on randomly chosen data, in such a way that anyone can check the validity of the signature by using public data only. The user's public data are called the *public key*, whereas his secret data are called the *secret key*. The intuitive security notion would be the impossibility to forge user's signatures without the knowledge of his secret key. In this section, we give a more precise definition of signature schemes and of the possible attacks against them (most of those definitions are based on [9]).

2.1 Definitions

A signature scheme is defined by the three following algorithms:

- The *key generation algorithm* G. On input 1^k, where k is the security parameter, the algorithm G produces a pair $(\mathsf{k_p}, \mathsf{k_s})$ of matching public and secret keys. Algorithm G is probabilistic.
- The *signing algorithm* Σ. Given a message m and a pair of matching public and secret keys $(\mathsf{k_p}, \mathsf{k_s})$, Σ produces a signature σ. The signing algorithm might be probabilistic.
- The *verification algorithm* V. Given a signature σ, a message m and a public key $\mathsf{k_p}$, V tests whether σ is a valid signature of m with respect to $\mathsf{k_p}$. In general, the verification algorithm need not be probabilistic.

2.2 Forgeries and Attacks

In this subsection, we formalize some security notions which capture the main practical situations. On the one hand, the **goals** of the adversary may be various:

- Disclosing the secret key of the signer. It is the most serious attack. This attack is termed *total break*.
- Constructing an efficient algorithm which is able to sign messages with good probability of success. This is called *universal forgery*.
- Providing a new message-signature pair. This is called *existential forgery*.

In many cases this latter forgery, the *existential forgery*, is not dangerous, because the output message is likely to be meaningless. Nevertheless, a signature scheme which is not existentially unforgeable (and thus that admits existential forgeries) does not guarantee by itself the identity of the signer. For example, it cannot be used to certify randomly looking elements, such as keys. Furthermore, it cannot formally guarantee the non-repudiation property, since anyone may be able to produce a message with a valid signature.

On the other hand, various **means** can be made available to the adversary, helping her into her forgery. We focus on two specific kinds of attacks against signature schemes: the *no-message attacks* and the *known-message attacks*. In the first scenario, the attacker only knows the public key of the signer. In the second one, the attacker has access to a list of valid message-signature pairs. According to the way this list was created, we usually distinguish many subclasses, but the strongest is the *adaptively* *chosen-message attack*, where the attacker can ask the signer to sign any message of her choice. She can therefore adapt her queries according to previous answers.

When one designs a signature scheme, one wants to computationally rule out existential forgeries even under adaptively chosen-message attacks, which is the strongest security level for a signature scheme.

3 Generic Algorithms

Before we proceed, let us stress that although the generic model in which we analyze DSA offers a somehow weaker form of security than the reductions that we apply to GHR in the standard model, it still provides *evidence* that twinning may indeed have a beneficial effect on security.

Generic algorithms [15, 21], as introduced by Nechaev and Shoup, encompass group algorithms that do not exploit any special property of the encodings of group elements other than the property that each group element is encoded by a unique string. Typically, algorithms like Pollard's ρ algorithm [18] fall under the scope of this formalism while index-calculus methods do not.

3.1 The Framework

Recall that any Abelian finite group Γ is isomorphic to a product of cyclic groups of the form $(\mathbb{Z}_{p^k}, +)$, where p is a prime. Such groups will be called standard Abelian groups. An encoding of a standard group Γ is an injective map from Γ into a set of bit-strings S.

We give some examples: consider the multiplicative group of invertible elements modulo some prime q. This group is cyclic and isomorphic to the standard additive group $\Gamma = \mathbb{Z}_{q-1}$. Given a generator g, an encoding σ is obtained by computing the binary representation $\sigma(x)$ of $g^x \bmod q$. The same construction applies when one considers a multiplicative subgroup of prime order r. Similarly, let E be the group of points of some non-singular elliptic curve over a finite field $\mathbb{F}$, then E is either isomorphic to a (standard) cyclic group Γ or else is isomorphic to a product of two cyclic groups $\mathbb{Z}_{d_1} \times \mathbb{Z}_{d_2}$. In the first case, given a generator G of E, an encoding is obtained by computing $\sigma(x) = x.G$, where $x.G$ denotes the scalar multiplication of G by the integer x and providing coordinates for $\sigma(x)$. The same construction applies when E is replaced by one of its subgroups of prime order r. Note that the encoding set appears much larger than the group size, but compact encodings using only one coordinate and a sign bit ± 1 exist and for such encodings, the image of σ is included in the binary expansions of integers $< tr$ for some small integer t, provided that r is close enough to the size of the underlying field $\mathbb{F}$. This is exactly what is recommended for cryptographic applications [10].

A *generic* algorithm $\mathcal{A}$ over a standard Abelian group Γ is a probabilistic algorithm that takes as input an *encoding list* $\{\sigma(x_1), \cdots, \sigma(x_k)\}$, where each x_i is in Γ. While it executes, the algorithm may consult an oracle for further encodings. Oracle calls consist of triples $\{i, j, \epsilon\}$, where i and j are indices of the encoding list and ϵ is $\pm$. The oracle returns the string $\sigma(x_i \pm x_j)$, according to the value of ϵ and this bit-string is appended to the list, unless it was already present. In other words, $\mathcal{A}$ cannot access an element of Γ directly but only through its name $\sigma(x)$ and the oracle provides names for the sum or difference of two elements addressed by their respective names. Note however that $\mathcal{A}$ may access the list at any time. In many cases, $\mathcal{A}$ takes as input a pair $\{\sigma(1), \sigma(x)\}$. Probabilities related to such algorithms are computed with respect to the internal coin tosses of $\mathcal{A}$ as well as the random choices of σ and x.

The following theorem appears in [21]:

Theorem 1. *Let Γ be a standard cyclic group of order N and let p be the largest prime divisor of N. Let $\mathcal{A}$ be a generic algorithm over Γ that makes at most n queries to the oracle. If $x \in \Gamma$ and an encoding σ are chosen at random, then the probability that $\mathcal{A}$ returns x on input $\{\sigma(1), \sigma(x)\}$ is $\mathcal{O}(n^2/p)$.*

Proof. We refer to [21] for a proof. However, we will need, as an ingredient for our own proofs, the probabilistic model used by Shoup. We develop the model in the special case where N is a prime number r, which is of interest to us. Alternatively, we could work in a subgroup of prime order r.

Basically, we would like to identify the probabilistic space consisting of σ and x with the space $S^{n+2} \times \Gamma$, where S is the set of bit-string encodings. Given a tuple $\{z_1, \cdots, z_{n+2}, y\}$ in this space, z_1 and z_2 are used as $\sigma(1)$ and $\sigma(x)$, the successive z_i are used in sequence to answer the oracle queries and the unique value y from Γ serves as x. However, this interpretation may yield inconsistencies as it does not take care of possible collisions between oracle queries. To overcome the difficulty, Shoup defines, along with the execution of $\mathcal{A}$, a sequence of linear polynomials $F_i(X)$, with coefficients modulo r. Polynomials F_1 and F_2 are respectively set to $F_1 = 1$ and $F_2 = X$ and the definition of polynomial F_ℓ is related to the ℓ-th query $\{i, j, \epsilon\}$: $F_\ell = F_i \pm F_j$, where the sign $\pm$ is chosen according to ϵ. If F_ℓ is already listed as a previous polynomial F_h, then F_ℓ is marked and $\mathcal{A}$ is fed with the answer of the oracle at the h-th query. Otherwise, z_ℓ is returned by the oracle. Once $\mathcal{A}$ has come to a stop, the value of x is set to y.

It is easy to check that the behavior of the algorithm which plays with the polynomials F_i is exactly similar to the behavior of the regular algorithm, if we require that y is not a root of any polynomial $F_i - F_j$, where i, j range over indices of unmarked polynomials. A sequence $\{z_1, \cdots, z_{n+2}, y\}$ for which this requirement is met is called a *safe* sequence. Shoup shows that, for any $\{z_1, \cdots, z_{n+2}\}$, the set of y such that $\{z_1, \cdots, z_{n+2}, y\}$ is not safe has probability $\mathcal{O}(n^2/r)$. From a safe sequence, one can define x as y and σ as any encoding which satisfies $\sigma(F_i(y)) = z_i$, for all unmarked F_i. This correspondence preserves probabilities. However, it does not completely cover the sample space $\{\sigma, x\}$ since executions such that $F_i(x) = F_j(x)$, for some indices i, j, such that F_i and F_j are not identical are omitted. To conclude the proof of the above theorem in the special case where N is a prime number r, we simply note that the output of a computation corresponding to a safe sequence $\{z_1, \cdots, z_{n+2}, y\}$ does not depend on y. Hence it is equal to y with only minute probability.

3.2 Digital Signatures over Generic Groups

We now explain how generic algorithms can deal with attacks against DSA-like signature schemes [6, 20, 16, 10]. We do this by defining a generic version of DSA that we call GDSA. Parameters for the signature include a standard cyclic group of prime order r together with an encoding σ. The signer also uses as a secret key/public key pair $\{x, \sigma(x)\}$. Note that we have chosen to describe signature generation as a regular rather than generic algorithm, using a full description of σ. To sign a message m, $1 < m < r$ the algorithm executes the following steps:

1. Generate a random number u, $1 \leq u < r$.

2. Compute $c \leftarrow \sigma(u) \bmod r$. If $c = 0$ go to step 1.
3. Compute $d \leftarrow u^{-1}(m + xc) \bmod r$. If $d = 0$ go to step 1.
4. Output the pair $\{c, d\}$ as the signature of m.

The verifier, on the other hand, is generic:

1. If $c \notin [1, r-1]$ or $d \notin [1, r-1]$, output invalid and stop.
2. Compute $h \leftarrow d^{-1} \bmod r$, $h_1 \leftarrow hm \bmod r$ and $h_2 \leftarrow hc \bmod r$.
3. Obtain $\sigma(h_1 + h_2 x)$ from the oracle and compute $c' \leftarrow \sigma(h_1 + h_2 x) \bmod r$.
4. If $c \neq c'$ output invalid and stop otherwise output valid and stop.

The reader may wonder how the verifier obtains the value of σ requested at step 3. This is simply achieved by mimicking the usual double-and-add algorithm and asking the appropriate queries to the oracle. This yields $\sigma(h_1)$ and $\sigma(h_2 x)$. A final call to the oracle completes the task.

A generic algorithm $\mathcal{A}$ can also perform forgery attacks against a signature scheme. This is defined by the ability of $\mathcal{A}$ to return on input $\{\sigma(1), \sigma(x)\}$ a triple $\{m, c, d\} \in \Gamma^3$ for which the verifier outputs valid. Here we assume that both algorithms are performed at a stretch, keeping the same encoding list.

To deal with adaptive attacks one endows $\mathcal{A}$ with another oracle, called the signing oracle. To query this oracle, the algorithm provides an element $m \in \Gamma$. The signing oracle returns a valid signature $\{c, d\}$ of m. Success of $\mathcal{A}$ is defined by its ability to produce a valid triple $\{\tilde{m}, \tilde{c}, \tilde{d}\}$, such that $\tilde{m}$ has not been queried during the attack.

Such a forgery can be easily performed against this GDSA scheme, even with just a passive attack: the adversary chooses random numbers h_1 and h_2, $1 \leq h_1, h_2 < r$ and computes $c \leftarrow \sigma(h_1 + h_2 x) \bmod r$. Then it defines $d = c h_2^{-1} \bmod r$, $h = d^{-1} \bmod r$, and eventually $m = d h_1 \bmod r$. The triple $\{m, c, d\} \in \Gamma^3$ is therefore a valid one, unless $c = 0$, which is very unlikely.

4 The Security of Twin GDSA

4.1 A Theoretical Result

The above definitions extend to the case of twin signatures, by requesting the attacker $\mathcal{A}$ to output an m and two distinct pairs $\{c, d\} \in \Gamma^2$, $\{c', d'\} \in \Gamma^2$. Success is granted as soon as the verifying algorithm outputs valid for both triples[1]. We prove the following:

Theorem 2. *Let Γ be a standard cyclic group of prime order r. Let S be a set of bit-string encodings of cardinality at least r, included in the set of binary representations of integers $< tr$, for some t. Let $\mathcal{A}$ be a generic algorithm over Γ that makes at most n queries to the oracle. If $x \in \Gamma$ and an encoding σ are chosen at random, then the probability that $\mathcal{A}$ returns a message m together with two distinct GDSA signatures of m on input $\{\sigma(1), \sigma(x)\}$ is $\mathcal{O}(tn^2/r)$.*

Proof. We cover the non adaptive case and tackle the more general case after the proof. We use the probabilistic model developed in section 3.1. Let $\mathcal{A}$ be a generic attacker able to forge some m and two distinct signatures $\{c, d\}$ and $\{c', d'\}$. We assume that, once these outputs have been produced, $\mathcal{A}$ goes on checking both signatures; we estimate the probability that both are valid.

[1] using [14] the simultaneous square-and-multiply generation or verification of two DSA signatures is only 17% slower than the generation or verification of a single signature.

We restrict our attention to behaviors of the full algorithm corresponding to safe sequences $\{z_1, \cdots, z_{n+2}, y\}$. By this, we discard a set of executions of probability $\mathcal{O}(n^2/r)$. We let P be the polynomial $(md^{-1}) + (cd^{-1})X$ and Q be the polynomial $(md'^{-1}) + (c'd'^{-1})X$.

- We first consider the case where either P or Q does not appear in the F_i list before the signatures are produced. If this happens for P, then P is included in the F_i list at signature verification and the corresponding answer of the oracle is a random number z_i. Unless $z_i = c \bmod r$, which is true with probability at most t/r, the signature is invalid. A similar bound holds for Q.
- We now assume that both P and Q appear in the F_i list before $\mathcal{A}$ outputs its signatures. We let i denote the first index such that $F_i = P$ and j the first index such that $F_j = Q$. Note that both F_i and F_j are unmarked (as defined in section 3.1). If $i = j$, then we obtain that $md^{-1} = md'^{-1}$ and $cd^{-1} = c'd'^{-1}$. From this, it follows that $c = c'$, $d = d'$ and the signatures are not distinct.
- We are left with the case where $i \neq j$. We let $\Omega_{i,j}$, $i < j$, be the set of safe sequences producing two signatures such that the polynomials P, Q, defined as above appear for the first time before the algorithm outputs the signatures, as F_i and F_j. We consider a fixed value w for $\{z_1, \cdots, z_{j-1}\}$ and let $\hat{w}$ be the set of safe sequences extending w. We note that F_i and F_j are defined from w and we write $F_i = a + bX$, $F_j = a' + b'X$. We claim that $\Omega_{i,j} \cap \hat{w}$ has probability $\leq t/r$. To show this, observe that one of the signatures that the algorithm outputs is necessarily of the form $\{c, d\}$, with $c = z_i \bmod r$, $c = db \bmod r$ and $m = da \bmod r$. Now, the other signature is $\{c', d'\}$ and since m is already defined we get $d' = ma'^{-1} \bmod r$ and $c' = b'd' \bmod r$. This in turn defines $z_j \bmod r$ within a subset of at most t elements. From this, the required bound follows and, from the bound, we infer that the probability of $\Omega_{i,j}$ is at most t/r.

Summing up, we have bounded the probability that a safe sequence produces an execution of $\mathcal{A}$ outputting two valid signatures by $\mathcal{O}(tn^2/r)$. This finishes the proof.

In the proof, we considered the case of an attacker forging a message-signature pair from scratch. A more elaborate scenario corresponds to an attacker who can adaptively request twin signatures corresponding to messages of his choice. In other words, the attacker interacts with the legitimate signer by submitting messages selected by its program.

We show how to modify the security proof that was just given to cover the adaptive case. We assume that each time it requests a signature the attacker $\mathcal{A}$ immediately verifies the received signature. We also assume that the verification algorithm is normalized in such a way that, when verifying a signature $\{c, d\}$ of a message m, it asks for $\sigma((md^{-1}) + (cd^{-1})x)$ after a fixed number of queries, say q. We now explain how to simulate signature generation: as before, we restrict our attention to behaviors of the algorithm corresponding to safe sequences $\{z_1, \cdots, z_{n+2}, y\}$. When the (twin) signature of m is requested at a time of the computation when the encoding list contains i elements, one picks z_{i+q} and z_{i+2q} and manufactures the two signatures as follows:

1. Let $c \leftarrow z_{i+q} \bmod r$, pick d at random.
2. Let $c' \leftarrow z_{i+2q} \bmod r$, pick d' at random.
3. Output $\{c, d\}$ and $\{c', d'\}$ as the first and second signatures.

While verifying both signatures, $\mathcal{A}$ will receive the elements z_{i+q} and z_{i+2q}, as

$$\sigma((md^{-1}) + (cd^{-1})x) \text{ and } \sigma((md'^{-1}) + (c'd'^{-1})x)$$

respectively, unless F_{i+q} or F_{i+2q} appears earlier in the F_i list. Due to the randomness of d and d', this happens with very small probability bounded by n/r. Altogether, the simulation is spotted with probability $\mathcal{O}(n^2/r)$ which does not affect the $\mathcal{O}(tn^2/r)$ bound for the probability of successful forgery.

4.2 Practical Meaning of the Result

We have shown that, in the setting of generic algorithms, existential forgery against twin GDSA has a minute success probability. Of course this does not tell anything on the security of actual twin DSA. Still, we believe that our proof has some *practical* meaning. The analogy with hash functions and the random oracle model [3] is inspiring: researchers and practitioners are aware that proofs in the random oracle model are not proofs but a mean to spot design flaws and validate schemes that are supported by such proofs. Still, all standard signature schemes that have been proposed use specific functions which are not random by definition; our proofs seem to indicate that if existential forgery against twin DSA is possible, it will require to dig into structural properties of the encoding function. This is of some help for the design of actual schemes: for example, the twin DSA described in Appendix A allows signature with message recovery without hashing and without any form of redundancy, while keeping some form of provable security. This might be considered a more attractive approach than [17] or [1], the former being based on redundancy and the latter on random oracles. We believe that twin DSA is even more convincing in the setting of elliptic curves, where there are no known ways of taking any advantage of the encoding function.

5 An RSA-based Twinning in the Standard Model

The twin signature scheme described in this section belongs to the (very) short list of efficient schemes provably secure in the standard model: in the sequel, we show that producing existential forgeries even under an adaptively chosen-message attack is equivalent to solving the Flexible RSA Problem [5].

Security in the standard model implies no ideal assumptions; in other words we directly reduce the Flexible RSA Problem to a forgery. As a corollary, we present an efficient and provably secure signature scheme that does not require any hash function.

Furthermore, the symmetry provided by twinning is much simpler to analyze than Cramer-Shoup's proposal [5] which achieves a similar security level, and similar efficiency, with a rather intricate proof.

5.1 Gennaro-Halevi-Rabin Signatures

In [8] Gennaro, Halevi and Rabin present the following signature scheme: Let n be an ℓ-bit RSA modulus [19], H a hash-function and $y \in \mathbb{Z}_n^\star$. The pair $\{n, y\}$ is the signer's public key, whose secret key is the factorization of n.

- To sign m, the signer hashes $e \leftarrow H(m)$ (which is very likely to be co-prime with $\varphi(n)$) and computes the e-th root of y modulo n using the factorization of n:

$$s \leftarrow y^{1/e} \bmod n$$

– To verify a given $\{m, s\}$, the verifier checks that

$$s^{H(m)} \bmod n \stackrel{?}{=} y.$$

Security relies on the Strong RSA Assumption. Indeed, if H outputs elements that contain at least a new prime factor, existential forgery is impossible. Accordingly, Gennaro *et al.* define a new property that H must satisfy to yield secure signatures: *division intractability*. Division intractability means that it is computationally impossible to find $a_1, \ldots, a_k$ and b such that $H(b)$ divides the product of all the $H(a_i)$. In [8], it is conjectured that such functions exist and heuristic conversions from collision-resistant into division-intractable functions are shown (see also [4]).

Still, security against adaptively chosen-message attacks requires the hash function H to either behave like a random oracle model or achieve the chameleon property [12]. This latter property, for a hash function, provides a trapdoor which helps to find second preimages, even with some fixed part. Indeed, some signatures can be pre-computed, but with specific exponents before outputting y: $y = x^{\prod_i e_i} \bmod n$ for random primes $e_i = H(m_i, r_i)$.

Using the chameleon property, for the i-th query m to the signing oracle, the simulator who knows the trapdoor can get an r such that $H(m_i, r_i) = H(m, r) = e_i$. In the random oracle model, one simply defines $H(m, r) \leftarrow e_i$.

Then $s = x^{\prod_{j \neq i} e_j} = y^{1/e_i} \bmod n$ and the signature therefore consists of the triple $\{m, r, s\}$ satisfying

$$s^{H(m,r)} = y \bmod n.$$

Cramer and Shoup [5] also proposed a scheme based on the Strong RSA Assumption, the first practical signature scheme to be secure in the standard model, but with universal one-way hash functions; our twin scheme will be similar but with a nice symmetry in the description (which helps for the security analysis) and no hash-functions, unless one wants to sign a long message.

5.2 Preliminaries

We build our scheme in two steps. The first scheme resists existential forgeries when subjected to no-message attacks. Twinning will immune it against adaptively chosen-message attacks.

Injective function into the prime integers. Before any description, we will assume the existence of a function p with the following properties: given a security parameter k (which will be the size of the signed messages), p maps any string from $\{0, 1\}^k$ into the set of the prime integers, p is also designed to be easy to compute and injective. A candidate is proposed and analyzed in Appendix B.

The Flexible RSA Problem and the Strong RSA Assumption. Let us also recall the *Flexible RSA Problem* [5]. Given an RSA modulus n and an element $y \in \mathbb{Z}_n^\star$, find any exponent $e > 1$, together with an element x such that $x^e = y \bmod n$.

The *Strong RSA Assumption* is the conjecture that this problem is intractable for large moduli. This was independently introduced by [2, 7], and then used in many further security analyses (*e.g.* [5, 8]).

5.3 A First GHR Variant

The first scheme is very similar to GHR without random oracles but with function p instead:

- To sign $m \in \{0,1\}^k$, the signer computes $e \leftarrow p(m)$ and the e-th root of y modulo n using the factorization of n

$$s \leftarrow y^{1/e} \bmod n$$

- To verify a given $\{m, s\}$, the verifier checks that

$$s^{p(m)} \bmod n \stackrel{?}{=} y.$$

Since p provides a new prime for each new message (injectivity), existential forgery contradicts the Strong RSA Assumption. However, how can we deal with adaptively chosen-message attacks without any control over the output of the function p, which is a publicly defined non-random oracle and not a trapdoor function either?

5.4 The Twin Version

The final scheme is quite simple since it consists in duplicating the previous one: the signer uses two ℓ-bit RSA moduli n_1, n_2 and two elements y_1, y_2 in $\mathbb{Z}_{n_1}^\star$ and $\mathbb{Z}_{n_2}^\star$ respectively. Secret keys are the prime factors of the n_i.

- To sign a message m, the signer probabilistically derives two messages $\mu_1, \mu_2 \in \{0,1\}^k$, (from m and a random tape ω), computes $e_i \leftarrow p(\mu_i)$ and then the e_i-th root of y_i modulo n_i, for $i = 1, 2$, using the factorization of the moduli:

$$\{s_1 \leftarrow y_1^{1/e_1} \bmod n_1, s_2 \leftarrow y_2^{1/e_2} \bmod n_2\}$$

- To verify a given $\{m, \omega, s_1, s_2\}$, the verifier computes μ_1 and μ_2, then checks that $s_i^{p(\mu_i)} \bmod n_i \stackrel{?}{=} y_i$, for $i = 1, 2$.

To prevent forgeries, a new message must involve a new exponent, either e_1 or e_2, which never occurred in the signatures provided by the signing oracle. Therefore, a first requirement is that μ_1 and μ_2 define at most one message m, but only if they have been correctly constructed. Thus, some redundancy is furthermore required.

We thus suggest the following derivation, to get μ_1 and μ_2 from $m \in \{0,1\}^{k/2}$ (we assume k to be even): one chooses two random elements $a, b \in \{0,1\}^{k/2}$, then $\mu_1 = (m \oplus a)||(m \oplus b)$ and $\mu_2 = a||b$.

Clearly, given μ_1 and μ_2, one gets back $M = \mu_1 \oplus \mu_2$, which provides a valid message if and only if the redundancy holds: $\overline{M} = \underline{M}$, where $\overline{S}$ and $\underline{S}$ denote the two $k/2$-bit halves of a k-bit string S, the most significant and the least significant parts respectively.

5.5 Existential Forgeries

Let us show that existential forgery of the twin scheme, with above derivation process, leads to a new solution of the Flexible RSA Problem:

Lemma 3. *After q queries to the signing oracle, the probability that there exist a new message m and values a, b, which lead to $\mu_1 = (m \oplus a)||(m \oplus b)$ and $\mu_2 = a||b$, such that both $e_1 = p(\mu_1)$ and $e_2 = p(\mu_2)$ already occurred in the signatures provided by the signing oracle is less than $q^2/2^{k/2}$.*

Proof. Let $\{m_i, a_i, b_i, s_{1,i}, s_{2,i}\}$ denote the answers of the signing oracle. Using the injectivity of p, the existence of such m, a and b means that there exist indices i and j for which

$$(m \oplus a)||(m \oplus b) = \mu_1 = \mu_{1,i} = (m_i \oplus a_i)||(m_i \oplus b_i)$$
$$a||b = \mu_2 = \mu_{2,j} = a_j||b_j.$$

Then
$$a \oplus b = (m \oplus a) \oplus (m \oplus b) = (m_i \oplus a_i) \oplus (m_i \oplus b_i) = a_i \oplus b_i,$$
and
$$a \oplus b = a_j \oplus b_j.$$
Therefore, for a $j > i$ (the case $i > j$ is similar), the new random elements a_j, b_j must satisfy $a_j \oplus b_j = a_i \oplus b_i$. Since it is randomly chosen by the signer, the probability that this occurs for some $i < j$ is less than $(j-1)/2^{k/2}$.

Altogether, the probability that for some j there exists some $i < j$ which satisfies the above equality is less that $q^2/2 \times 2^{-k/2}$. By symmetry, we obtain the same result if we exchange i and j.

The probability that both exponents already appeared is consequently smaller than $q^2/2^{k/2}$.

To prevent adaptively chosen-message attacks, we need no trapdoor property for p, nor random oracle assumption either. We simply give the factorization of one modulus to the simulator, which can use any pre-computed exponentiation with any new message, as when chameleon functions are used [8].

5.6 Adaptively Chosen-Message Attacks

Indeed, to prevent adaptively chosen-message attacks, one just needs to describe a simulator; our simulator works as follows:

- The simulator is first given the moduli n_1, n_2 and the elements $y_1 \in \mathbb{Z}_{n_1}^\star$, $y_2 \in \mathbb{Z}_{n_2}^\star$, as well as the factorization of n_γ, where γ is randomly chosen in $\{1,2\}$. To simplify notations we assume that $\gamma = 1$. And the following works without loss of generality since the derivation of μ_1 and μ_2 is perfectly symmetric: they are randomly distributed, but satisfy $\mu_1 \oplus \mu_2 = m||m$ (it is a perfect secret sharing).
- The simulator randomly generates q values $e_{2,j} \leftarrow p(\mu_{2,j})$, with randomly chosen $\mu_{2,j} \in_R \{0,1\}^k$ for $j = 1, \ldots, q$ and computes
$$z \leftarrow y_2^{\prod_{j=1,\ldots,q} e_{2,j}} \bmod n_2.$$
The new public key for the signature scheme is the following: the moduli n_1, n_2 with the elements y_1, z in $\mathbb{Z}_{n_1}^\star$ and $\mathbb{Z}_{n_2}^\star$ respectively.
- For the j-th signed message m, the simulator first gets $(a||b) \leftarrow (m||m) \oplus \mu_{2,j}$. It therefore computes $\mu_1 \leftarrow a||b$, and thus $\mu_2 \leftarrow \mu_{2,j} = (m \oplus a)||(m \oplus b)$.
Then, it knows $s_2 = y_2^{\prod_{i \neq j} e_{2,i}} \bmod n_2$, and computes s_1 using the factorization of n_1.

Such a simulator can simulate up to q signatures, which leads to the following theorem.

Theorem 4. *Let us consider an adversary against the twin-GHR scheme who succeeds in producing an existential forgery, with probability greater than ε, after q adaptive queries to the signing oracle in time t, then the Flexible RSA Problem can be solved with probability greater than ε' within a time bound t', where*

$$\varepsilon' = \frac{1}{2}\left(\varepsilon - \frac{q^2}{2^{k/2}}\right) \quad \textit{and} \quad t' = t + \mathcal{O}(q \times \ell^2 \times k).$$

Proof. Note that the above bounds are almost optimal since $\varepsilon' \cong \varepsilon/2$ and $t' \cong 2t$. Indeed, the time needed to produce an existential forgery after q signature queries is already in $\mathcal{O}(q \times (|n_1|^2 + |n_2|^2)k)$. To evaluate the success probability, q is less than say 2^{40}, but k may be taken greater than 160 bits (and even much more).

To conclude the proof, one just needs to address the random choice of γ. As we have seen in Lemma 3, with probability greater than $\varepsilon - q^2/2^{k/2}$, one of the exponents in the forgery never appeared before. Since γ is randomly chosen and the view of the simulation is perfectly independent of this choice, with probability of one half, $e = e_{\bar{\gamma}}$ is new. Let us follow our assumption that $\gamma = 1$, then

$$s^e = s_2^e = z = y_2^\pi \bmod n_2,$$

where $\pi = \prod_{j=1,\ldots,q} e_{2,j}$. Since e is new, it is relatively prime with π, and therefore, there exist u and v such that $ue + v\pi = 1$: let us define $x = y_2^u s^v \bmod n_2$,

$$x^e = (y_2^u s^v)^e = y_2^{1-v\pi} s^{ev} = y_2 (y_2^\pi)^{-v} (s^e)^v = y_2 \bmod n_2.$$

We thus obtain an e-th root of the given y_2 modulo n_2, for a new prime e.

5.7 More Signatures

One may remark that the length of the messages we can sign with above construction is limited to $k/2$ bits, because of the required redundancy. But one can increase the size, by signing three derived messages: in order to sign $m \in \{0,1\}^k$, one chooses two random elements $a, b \in \{0,1\}^{k/2}$ (we still assume k to be even), and signs with different moduli

$$\begin{aligned} \mu_1 &= m \oplus (a||b) \\ \mu_2 &= a||b \\ \mu_3 &= m \oplus (b||a). \end{aligned}$$

6 Conclusion and Further Research

We proposed an alternative to the well-known hash-and-sign paradigm, based on the simple idea of signing twice (or more) identical or related short messages. We believe that our first investigations show that this is a promising strategy, deserving further study.

A number of interesting questions remain open. First, from the efficiency point of view, which is a frequent concern, we are aware that the current proposals do not deal with either the computational cost, or the communication load, in an efficient way.

Thus, for example, can the number of fields in a twin DSA be reduced from four ($\{c, d\}$ and $\{c', d'\}$) to three or less? Can we also suppress some fields in the twin-GHR, or sign k-bit long messages with only two signatures?

Finally, can an increase in the number of signatures (e.g. three instead of two) yield better security bounds?

References

1. M. Abe and T. Okamoto. A Signature Scheme with Message Recovery as Secure as Discrete Logarithm. In *Asiacrypt '99*, LNCS 1716. Springer-Verlag, Berlin, 1999.
2. N. Barić and B. Pfitzmann. Collision-Free Accumulators and Fail-Stop Signature Schemes without Trees. In *Eurocrypt '97*, LNCS 1233, pages 480–484. Springer-Verlag, Berlin, 1997.
3. M. Bellare and P. Rogaway. Random Oracles Are Practical: a Paradigm for Designing Efficient Protocols. In *Proc. of the 1st CCS*, pages 62–73. ACM Press, New York, 1993.
4. J.-S. Coron and D. Naccache. Security Analysis of the Gennaro-Halevi-Rabin Signature Scheme. In *Eurocrypt '99*, LNCS 1592, pages 91–101. Springer-Verlag, Berlin, 1999.
5. R. Cramer and V. Shoup. Signature Scheme based on the Strong RSA Assumption. In *Proc. of the 6th CCS*, pages 46–51. ACM Press, New York, 1999.
6. T. El Gamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory*, IT–31(4):469–472, July 1985.
7. E. Fujisaki and T. Okamoto. Statistical Zero Knowledge Protocols to Prove Modular Polynomial Relations. In *Crypto '97*, LNCS 1294, pages 16–30. Springer-Verlag, Berlin, 1997.
8. R. Gennaro, S. Halevi, and T. Rabin. Secure Hash-and-Sign Signature Without the Random Oracle. In *Eurocrypt '99*, LNCS 1592, pages 123–139. Springer-Verlag, Berlin, 1999.
9. S. Goldwasser, S. Micali, and R. Rivest. A Digital Signature Scheme Secure Against Adaptative Chosen-Message Attacks. *SIAM Journal of Computing*, 17(2):281–308, April 1988.
10. IEEE P1363. Standard Specifications for Public Key Cryptography. Available from `http://grouper.ieee.org/groups/1363`, August 1998.
11. M. Jakobsson and C. P. Schnorr. Security of Discrete Logarithm Cryptosystems in the Random Oracle Model and Generic Model. Available from `http://www.bell-labs.com/~markusj`, 1998.
12. H. Krawczyk and T. Rabin. Chameleon Hashing and Signatures. In *Proc. of NDSS '2000*. Internet Society, 2000.
13. A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996. Available from `http://www.cacr.math.uwaterloo.ca/hac/`.
14. D. M'Raïhi and D. Naccache. Batch Exponentiation – A Fast DLP-based Signature Generation Strategy. In *Proc. of the 3rd CCS*, pages 58–61. ACM Press, New York, 1996.
15. V. I. Nechaev. Complexity of a Determinate Algorithm for the Discrete Logarithm. *Mathematical Notes*, 55(2):165–172, 1994.
16. NIST. Digital Signature Standard (DSS). Federal Information Processing Standards PUBlication 186, November 1994.
17. K. Nyberg and R. A. Rueppel. Message Recovery for Signature Schemes Based on the Discrete Logarithm Problem. In *Eurocrypt '94*, LNCS 950, pages 182–193. Springer-Verlag, Berlin, 1995.
18. J. M. Pollard. Monte Carlo Methods for Index Computation (mod p). *Mathematics of Computation*, 32(143):918–924, July 1978.
19. R. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.
20. C. P. Schnorr. Efficient Signature Generation by Smart Cards. *Journal of Cryptology*, 4(3):161–174, 1991.
21. V. Shoup. Lower Bounds for Discrete Logarithms and Related Problems. In *Eurocrypt '97*, LNCS 1233, pages 256–266. Springer-Verlag, Berlin, 1997.

A Twin Signatures with Message Recovery

In this appendix, we describe a twin version of the Nyberg-Rueppel scheme [17] which provides message recovery. Keeping the notations of section 4.1:

1. Generate a random number u, $1 \leq u < r$.

2. Compute $c \leftarrow \sigma(u) + m \bmod r$. If $c = 0$ go to step 1.
3. Compute an integer $d \leftarrow u - cx \bmod r$.
4. Output the pair $\{c, d\}$ as the signature.

In the above, f is what is called in [10] a *message with appendix*. It simply means that it has an adequate redundancy. The corresponding verification is performed by the following (generic) steps:

1. If $c \notin [1, r-1]$ or $d \notin [0, r-1]$, output invalid and stop.
2. Obtain $\sigma(d + cx)$ from the oracle and compute $\gamma \leftarrow \sigma(d + cx) \bmod r$.
3. Check the redundancy of $m \leftarrow c - \gamma \bmod r$. If incorrect output invalid and stop; otherwise output the reconstructed message m, output valid and stop.

In the twin setting, signature generation is alike but is performed twice, so as to output two distinct signatures. However, no redundancy is needed. The verifier simply checks that the signatures are distinct and outputs two successive versions of the message, say m and m'. It returns valid if $m \stackrel{?}{=} m'$ and invalid otherwise. The security proof is sketched here, we leave the discussion of adaptive attacks to the reader.

We keep the notations and assumptions of section 4 and let $\mathcal{A}$ be a generic attacker over Γ which outputs, on input $\{\sigma(1), \sigma(x)\}$, two signature pairs $\{c, d\}$, $\{c', d'\}$ and runs the verifying algorithm that produces from these signatures two messages m, m' and checks whether they are equal. We wish to show that, if $x \in \Gamma$ and an encoding σ are chosen at random, then the probability that $m = m'$ is $\mathcal{O}(tn^2/r)$.

As before, we restrict our attention to behaviors of the full algorithm corresponding to safe sequences $\{z_1, \cdots, z_n, y\}$. We let P, Q be the polynomials $d + cX$ and $d' + c'X$. We first consider the case where either P or Q does not appear in the F_i list before the signatures are produced. If this happens for P, then, P is included in the F_i list at signature verification and the corresponding answer of the oracle is a random number z_i. Since m is computed as $c - z_i \bmod r$, the probability that $m = m'$ is bounded by t/r. A similar bound holds for Q.

We now assume that both P and Q appear in the F_i list before $\mathcal{A}$ outputs its signatures. We let i denote the first index such that $F_i = P$ and j the first index such that $F_j = Q$. Note that both F_i and F_j are unmarked (as defined in section 3.1). If $i = j$, then we obtain that $c = c'$ and $d = d'$. From this, it follows that the signatures are not distinct.

As in section 4, we are left with the case where $i \neq j$ and we define $\Omega_{i,j}$, $i < j$, to be the set of safe sequences producing two signatures such that the polynomials P, Q, defined as above appear for the first time before the algorithm outputs the signatures, as F_i and F_j. We show that, for any fixed value $w = \{z_1, \cdots, z_{j-1}\}$, $\Omega_{i,j} \cap \hat{w}$ has probability $\leq t/r$, where $\hat{w}$ is defined as above. Since we have $m = c - z_i \bmod r$ and $m' = c' - z_j \bmod r$, we obtain $z_j = c' - c + z_i \bmod r$, from which the upper bound follows. From this bound, we obtain that the probability of $\Omega_{i,j}$ is at most t/r and, taking the union of the various $\Omega_{i,j}$s, we conclude that the probability to obtain a valid twin signature is at most $\mathcal{O}(tn^2/r)$.

B The Choice of Function p

B.1 A Candidate

The following is a natural candidate:

$$\begin{aligned} p : \{0,1\}^k &\to \mathcal{P} \\ m &\mapsto \texttt{nextprime}(m \times 2^\tau) \end{aligned}$$

where τ is suitably chosen to guarantee the existence of a prime in any set $[m \times 2^\tau, (m+1) \times 2^\tau[$, for $m < 2^k$.

Note that the deterministic property of `nextprime` is not mandatory, one just needs it to be injective. But then, the preimage must be easily recoverable from the prime: the exponent is sent as the signature, from which one checks the primality and extracts the message (message-recovery).

B.2 Analysis

It is clear that any generator of random primes, using m as a seed, can be considered as a candidate for p. The function proposed above is derived from a technique for accelerating prime generation called *incremental search* (e.g. [13], page 148).

1. Input: an odd k-bit number n_0 (derived from m)
2. Test the s numbers n_0, $n_0 + 2$, ..., $n_0 + 2(s-1)$ for primality

Under reasonable number-theoretic assumptions, if $s = c \cdot \ln 2^k$, the probability of failure of this technique is smaller than $2e^{-2c}$, for large k.

Using our notations, in such a way that there exists at least a prime in any set $[m \times 2^\tau, (m+1) \times 2^\tau[$, but with probability smaller than 2^{-80}, we obtain from above formulae that $c \cong 40$, and $2^\tau \geq 40 \ln 2^{k+\tau+1}$. Therefore, a suitable candidate is $\tau \cong 5 \log_2 k$, and less than $20k$ primality tests have to be performed.

B.3 Extensions

Collision-resistance: To sign large messages (at the cost of extra assumptions), one can of course use any collision-resistant hash-function h before signing (using the classical hash-and-sign technique). Clearly, the new function $m \mapsto p(h(m))$ is not mathematically injective, but just computationally injective (which is equivalent to collision-resistance), which is enough for the proof.

Division intractability: If one wants to improve efficiency, using the division-intractability conjecture proposed in [8], any function that outputs k-bit strings can be used instead of p. More precisely:

Definition 5 (Division Intractability). A function H is said (n, ν, τ)-division intractable if any adversary which runs in time τ cannot find, with probability greater than ν, a set of elements a_1, ..., a_n and b such that $H(b)$ divides the product of all the $H(a_i)$.

As above, that function p would not be injective, but just collision-resistant, which is enough to prove the following:

Theorem 6. *Let us consider the twin-GHR scheme where p is any (q, ε, t)-division-intractable hash function. Let us assume that an adversary $\mathcal{A}$ succeeds in producing an existential forgery under an adaptively chosen-message attack within time t and with probability greater than ε, after q queries to the signing oracle. Then one can either contradict the division-intractability assumption or solve the Flexible RSA Problem with probability greater than ε' within a time bound t', where*

$$\varepsilon' = \frac{1}{2}\left(\varepsilon - \frac{q^2}{2^{k/2}}\right) \quad \textit{and} \quad t' = t + \mathcal{O}(q \times \ell^2 \times k).$$