New Signature Schemes with Coupons and Tight Reduction

[Full version. Published in John Ioannidis, Angelos Keromytis, Moti Yung, Eds., *Applied Cryptography and Network Security 2005 – ACNS 2005*, vol. 3531 of *Lecture Notes in Computer Science*, pp. 513–528, Springer-Verlag, 2005.]

Benoît Chevallier-Mames[1,2]

[1] Gemplus, Card Security Group,
Av. du Jujubier, ZI Athélia IV, F-13705 La Ciotat Cedex, France

[2] École Normale Supérieure, Département d'Informatique,
45 rue d'Ulm, F-75230 Paris 05, France

`benoit.chevallier-mames@gemplus.com`

Abstract. Amongst provably secure signature schemes, two distinct classes are of particular interest: the ones with tight reduction (*e.g.*, RSA-PSS), and those which support the use of coupons (*e.g.*, Schnorr signature).

This paper introduces a new generic signature scheme based on any zero-knowledge identification protocol $\mathcal{Z}$ and signature scheme $\mathcal{S}$ verifying basic security properties. The so-obtained signature scheme features provable security with tight reduction under the same complexity assumptions as the ones under which the basic zero-knowledge identification protocol and signature scheme are secure. In addition to that, interestingly, the combined scheme supports coupons.

We propose an application of our generic conversion scheme based on RSA. We note however that any computational problem $\mathcal{P}$ could be turned into such a tight signature scheme supporting coupons for any zero-knowledge identification protocol and signature scheme based on $\mathcal{P}$. Interestingly, our design technique provides an alternative to the RSA-PSS signature standard, as it enjoys an *equivalently* tight security while enabling the use of coupons for increased performances.

1 Introduction

Signatures are certainly the most extensively used functionality in public key cryptography. Most popular signature schemes include RSA [25] or Schnorr [26] but a lot of other signature schemes have been proposed through the years. However, one had to wait until 1995 to find adequate security analysis of these thanks to Bellare and Rogaway's random oracle model which provided the source of the first security proofs for practical signature schemes [1, 2]. The use of this model typically allows to give the attacker access to oracles simulating hash functions and signatures, resulting in the computational transformation of a forged signature into the solution of a problem taken as a reference and assumed to be hard to solve (*e.g.*, integer factorization, e-th root or discrete logarithm extraction).

Even though a few signature schemes exist that are proven secure in the standard model [9, 5], proofs in the random oracle model are still widely used today as they lead to better reductions than any other proof technique.

Up to now, one of the most powerful reduction technique in the random oracle model is the Forking Lemma introduced by Pointcheval and Stern [20]. This technique is extremely general, and roughly consists in running the attacker repeatedly with different random oracles to get two distinct but related forged signatures. Most zero-knowledge signature schemes are such that, once applied, this technique allows one to recover the secret key.

The Forking Lemma is a powerful proof concept due to its generality: one only assumes the use of random oracles. Its major drawback, however, is that a loose reduction is obtained, *i.e.*, an attacker against the signature scheme can be used to break a supposedly intractable problem, but with a probability much smaller than the probability of a forgery. A consequence of that fact is that using a scheme proven secure with the Forking Lemma requires larger keys, resulting in a loss of efficiency. This constitutes a significant disadvantage in comparison with schemes such as RSA-PSS [2] for which there exists a proof turning a forger into a tight e-th root extractor [2, 4].

On the other hand, the Forking Lemma sometimes appears as the only way to come up with a proof that a given signature scheme is secure, particulary for those derived from zero-knowledge protocols via the Fiat-Shamir heuristic [8]. These last signature schemes have the appealing additional property that the signer can precompute a quantity of the signature independently from the message called a *coupon*, and use this precomputation later to generate the complete signature in a very fast way [8].

In [6], Goldreich and Micali proposed a method to convert signature schemes into schemes with coupons, with the restriction of using a one-time signature scheme. Later in [27], Shamir and Tauman proposed a different solution to achieve this goal relying on chameleon hash functions [15].

In this paper, we introduce a new technique to achieve the same goal, that combines a signature scheme with a zero knowledge identification protocol verifying properties discussed later in the paper. The obtained scheme can be simulated in the random oracle model in a very tight way; we show that the scheme is approximately as secure as the weakest of its constituents.

An attractive property of our scheme is that it is as fast as the third pass of the underlying zero knowledge identification protocol, as soon as the signer uses coupons while signing. We note that this is not possible with cryptosystems like RSA-PSS or RSA-FDH. Combining the properties of high computational performance and tight security reduction is especially desirable in constrained environments such as smart cards and shows the interest of our results for practitioners.

Our paper is divided into four parts. In the next section, we introduce zero-knowledge identification protocols, Σ-protocols and signature schemes. Section 3 describes our generic conversion and assesses its security in the random oracle model. We also compare our results with prior works [6, 27]. Finally, Section 4 provides a particularly interesting instantiation of our scheme based on the RSA problem.

2 Definitions and Related Work

2.1 Zero-Knowledge Identification Protocols

Zero-knowledge identification protocols were invented by Fiat and Shamir in [8] as an identification paradigm. They are often seen as a (usually three or four-pass) series of exchanges between a prover and a verifier. Zero-knowledge identification protocols are a way for the prover to convince that he knows a secret (thereby proving his identity) without revealing to the verifier any other information whatsoever about the secret itself. More precisely, a zero-knowledge identification protocol is referred to as a *proof of knowledge* that has in addition the zero-knowledge property captured by the notion of indistinguishable simulatability. We refer the reader to [17] for more about zero-knowledge protocols.

We consider in what follows a three-pass zero-knowledge identification protocol $\mathcal{Z}$ containing a key generator $\mathsf{G}en_{\mathcal{Z}}$ which generates public parameters $\alpha \in \Lambda$ and private parameters $s \in S$. The protocol is also defined by some public functions $\mathcal{U} : R \mapsto X$, $\mathcal{V} : S \times R \times G \mapsto Y$ and $\mathcal{W} : \Lambda \times G \times Y \mapsto X$, and runs as shown on the following picture:

- The prover picks a random $r \in R$, computes $x = \mathcal{U}(r)$ and sends x to the verifier.
- The verifier verifies that $x \in X$ and sends a random challenge $g \in G$ to the prover.
- The prover replies with $y = \mathcal{V}(s, r, g)$ and the verifier checks that $x = \mathcal{W}(\alpha, g, y)$ and $y \in Y$.

Fig. 1. A three-pass zero-knowledge identification protocol $\mathcal{Z}$.

2.2 Signature Schemes and Coupons

A signature scheme $\mathcal{S}$ is defined as a collection of probabilistic algorithms ($\mathsf{G}en_{\mathcal{S}}$, $\mathsf{S}ig, \mathsf{V}er$) used in the following way. During set up, a key pair is generated by running algorithm $\mathsf{G}en_{\mathcal{S}}$, and the private key d is kept secret by the legitime user while the public key β is published. Given a message $m \in M$, the signer computes a signature $\sigma = \mathsf{S}ig(d, m)$. A verifier can ascertain that a signature is valid by checking that $\mathsf{V}er(\beta, \sigma, m) = \textsc{True}$.

The well-known Fiat-Shamir heuristic allows to turn a zero-knowledge identification protocol into a signature scheme. Briefly, the Fiat-Shamir transform makes the protocol non interactive by replacing the verifier's challenge g by the result of hashing x and m with a secure hash function.

Signature schemes derived from identification protocols via the Fiat-Shamir transform are of particular interest as they allow the use of *coupons*: typically, the first step consisting in computing $x = \mathcal{U}(r)$ can be performed *before* receiving the message m. Later, the signature generation is completed with the computation of $g = \mathcal{G}(m, x)$ and $y = \mathcal{V}(s, r, g)$. This second, message-dependent computation stage happens to be much faster than the first step in most zero-knowledge identification protocols $\mathcal{Z}$.

2.3 Σ-protocols and Forking Lemma

Σ-protocols are zero-knowledge protocols featuring an additional property: given any couple of correct transcriptions (x, g_1, y_1) and (x, g_2, y_1) with $g_1 \neq g_2$, it is computationally easy to recover the prover's secret key s and consequently solve the computational problem $\mathcal{P}_{\mathcal{Z}}$ underlying the identification protocol. This directly implies that a coupon must be used only once.

This property is fulfilled by many protocols and is in fact the cornerstone of the *Forking Lemma* introduced by Pointcheval and Stern in [20] to prove security in the random oracle model. The intuition is that an attacker capable of forging a signature with some probability ε can be transformed, in the random oracle model, into an algorithm that finds two valid signatures (x, g_1, y_1) and (x, g_2, y_1) with $g_1 \neq g_2$ under probability $\mathcal{O}\left(\varepsilon^2\right)$.

Briefly, the reduction technique runs the attacker over random definitions of the oracle $\mathcal{G}$ until a forgery (x, g_1, y_1) is output by the attacker. Then, in the *replay phase*, the attacker is rerun over partially modified oracle definitions with the hope to get a second forgery (x, g_2, y_2). In this second forgery, the answer g_2 is different from g_1 with overwhelming probability.

The Forking Lemma, however, provides *loose* security reductions as an attacker breaking the security of a Σ-protocol with probability ε is turned into an algorithm solving $\mathcal{P}_{\mathcal{Z}}$ with a significantly smaller probability.

Contrarily to loose security, there exist signature schemes admitting *tight* security reductions, meaning that an attacker breaking $\mathcal{S}$ with a certain probability can be used to solve the underlying computational problem with similar probability.

2.4 A Generic Construction for Tight Security with Coupons

Very few signature schemes feature both a tight security and coupons. There exists however a construction by Shamir and Tauman that achieves this twofold goal using chameleon hash functions [15]. The basic idea is to use a chameleon hash function $\mathcal{H}$ to compute $\sigma = \mathsf{S}ig(d, \mathcal{H}(m', r'))$ for randomly chosen $m' \in \mathcal{M}$ and $r' \in R$. Given the message m, the signer simply has to compute r so that

$\mathcal{H}(m,r) = \mathcal{H}(m',r')$. The signature of $m \in \mathcal{M}$ is then (σ, r) and is easily verified by checking whether $\mathsf{V}er(\beta, \sigma, \mathcal{H}(m,r)) = \text{True}$.

The construction described in this paper is different, and we compare it to Shamir and Tauman's approach later in the paper.

2.5 Known-Message and Chosen-Message Attacks

Several security notions have been defined for signature schemes and properly formalized in the seminal work of Goldwasser, Micali and Rivest [12, 13]. To quantify the security of a signature scheme, one has to define the adversary's *goal* and *ressources*.

A typical goal resides in *existential forgery*: the adversary tries to create a valid message-signature pair for a freely chosen message. The corresponding security property that a signature scheme fulfills to resist such an attack is called *existential unforgeability* (EUF).

Beyond the verification key which is public and hence known to the adversary, more information about the secret key may also be available. The strongest access to side information is captured by the scenario of *adaptive chosen-message attacks* (CMA), where the attacker may request the signer to sign any message of its choice in an adaptive way. In this paper, we also need a weaker type of attack called *known-message attacks* (KMA), where the attacker receives a number q_s of message-signature pairs (m_i, σ_i) she has no control on.

We say that a signature scheme is secure against adaptive chosen-message attacks if it resists existential forgeries under adaptive chosen-message attacks (EUF-CMA). A signature scheme is said to be secure against known-message attacks if no existential forgery is computationally feasible under any known-message attack (EUF-KMA).

Obviously, signature schemes that are EUF-CMA secure also are EUF-KMA secure. It is also known that there exist signatures that are EUF-KMA secure without being EUF-CMA secure, thereby showing that these two notions are distinct.

3 The Proposed Scheme

In this section, we introduce a novel conversion scheme which provides a way to build new signature schemes and discuss its features and security properties.

3.1 Our Construction

The first ingredient of our construction is a Σ-protocol $\mathcal{Z}$ which security relies on a problem $\mathcal{P}_{\mathcal{Z}}$, defined as above by a key generator $\mathsf{G}en_{\mathcal{Z}}$ generating public and private parameters $\alpha \in \Lambda$ and $s \in S$, and by some public functions $\mathcal{U} : R \mapsto X$, $\mathcal{V} : S \times R \times G \mapsto Y$ and $\mathcal{W} : \Lambda \times G \times Y \mapsto X$. This is as described on Fig. 1. Examples of such Σ-protocols are numerous, including Feige-Fiat-Shamir [7], Guillou-Quisquater [14], Schnorr [26], Poupard-Stern [22], Girault-Poupard-Stern [10, 21].

Our scheme also uses a EUF-KMA-secure signature scheme $\mathcal{S} : H \mapsto X$ based on a problem $\mathcal{P}_{\mathcal{S}}$, defined as a triple $(\mathsf{G}en_{\mathcal{S}}, \mathsf{S}ig, \mathsf{V}er)$. All known signature schemes that are EUF-KMA-secure in the random oracle model are such that there exists a probabilistic simulator $\mathcal{S}im$ which outputs on demand the signature of q_s random messages in polynomial time $T_0 = \mathsf{poly}(q_s)$. We will make use of this simulator in our security proof. An example of such a simulator can be found in every known proof that FDH-RSA is EUF-KMA-secure. Finally, our scheme uses a collision-intractable hash function $\mathcal{G} : \mathcal{M} \times X \mapsto G$ and a full-domain-hash function $\mathcal{H} : X \mapsto H$.

Our signature scheme is as follows.

Key generation: A key is generated by running $\mathsf{G}en_{\mathcal{Z}}$ and $\mathsf{G}en_{\mathcal{S}}$. The private key of the scheme is (d, s) while the public key is (α, β).

Signature: To sign a message $m \in \mathcal{M}$, one randomly chooses $r \in R$ and computes $u = \mathcal{U}(r)$, $h = \mathcal{H}(u)$ and $x = \mathsf{S}ig(d, h)$. Upon receiving the message m, one computes $g = \mathcal{G}(m, x)$ and $y = \mathcal{V}(s, r, g)$. The signature on m is $\sigma = (x, y)$.

Verification: To verify a signature $\sigma = (x, y) \in X \times Y$, one computes $g' = \mathcal{G}(m, x)$, $u' = \mathcal{W}(\alpha, g', y)$ and $h' = \mathcal{H}(u')$. Finally, the signature σ is accepted iff $\mathsf{V}er(\beta, x, h') = \text{TRUE}$.

To simplify the description of the verification, we have supposed that $\mathcal{G}$ and $\mathcal{W}$ are only defined on their respective input sets, checking implicitly the fact that x and y are in the correct sets X and Y. When implementing these functions over larger sets, it is critical that these tests are added before computing $\mathcal{G}(m, x)$ and $\mathcal{W}(\alpha, g', y)$.

Note 1. Our scheme has basically three steps: computing a coupon of a zero-knowledge scheme, signing it with a signature scheme and, at the reception of the message, giving the response of the zero-knowledge scheme, corresponding to the hash of the message and the signature.

Signature size. The size of the signature is $|X| + |Y|$. This differs slightly from the size of the original signature scheme derived from $\mathcal{Z}$ via Fiat-Shamir, as in this scheme, a technique due to Schnorr reduces the signature size to $|G| + |Y|$.

Size of public and private parameters. As a combination of two schemes $\mathcal{Z}$ and $\mathcal{S}$, our general scheme has a lot of parameters, public or private. But for particular scheme instantiations, some parameters could be shared between the signature scheme and the zero-knowledge identification protocol, thus reducing the number or size of the parameters. A concrete example is given in Section 4, with an instantiation of our scheme based on RSA.

Performances of signature generation. Used in a classical way, the execution time of our proposed signature is roughly the addition of the execution times of $\mathsf{S}ig$, $\mathcal{U}$, $\mathcal{H}$, $\mathcal{G}$ and $\mathcal{V}$. Using coupons, however, the off-line part (*i.e.*, precomputing (x, r)) is carried out before the on-line part of the signature takes place. The on-line computation then requires computing a hash value and running $\mathcal{V}$ once. This, in most identification protocols, remains very fast. This is notably the case within Schnorr, Poupard-Stern and Girault-Poupard-Stern.

Ease of implementation. Our scheme relies on a few hash functions, an arbitrary EUF-KMA signature and a Σ-protocol that can be chosen among popular examples. Hence, the software development of new algorithms is unnecessary in order to implement our scheme, already existing software routines may be simply linked together as proposed. This is of particular interest for constrained devices such as smart cards, where the size of code memory is limited, and for which developments may take a long time. In this respect, clearly, the fact that our scheme reuses implemented and tested routines with *e.g.*, protections against side-channel and fault attacks is a strong advantage.

Furthermore, the management of the public key can be done within existing public key infrastructures (PKI) as soon as systems $\mathcal{S}$ and $\mathcal{Z}$ preexisted by themselves in the PKI.

Comparison with [6] and [27]. The approach of [6] remains faster than the present work, but it suffers from imposing too large signatures. In this respect, the construction given in [27] is actually closer to our work, even if based on a totally different design. The same security level is achieved: [27] is tightly based on the problem of finding collisions in the chameleon hash function and of forging a signature with the EUF-CMA-secure signature scheme, while, as shown in the sequel, our construction is tightly related to the problem of recovering the secret key of the zero-knowledge scheme $\mathcal{Z}$ and of forging a signature of the EUF-KMA secure signature scheme $\mathcal{S}$.

Security. Most interestingly, even against an EUF-CMA attacker, our scheme remains as secure as the weakest problem among $\mathcal{P}_{\mathcal{S}}$ and $\mathcal{P}_{\mathcal{Z}}$. This reduction is again tight, as shown in the next subsection. A natural construction is then to use a signature scheme $\mathcal{S}$ and a zero knowledge identification protocol $\mathcal{Z}$ that are based on the same problem, as proposed in Section 4.

3.2 Security of the scheme

We will prove that our scheme is secure in the random oracle model even when the attacker is given access to the signature of q_s messages of her choice. The adversary may also invoke random oracles returning the hash value of q_h inputs (more precisely $q_{\mathcal{H}}$ and $q_{\mathcal{G}}$ queries to $\mathcal{H}$ and $\mathcal{G}$, respectively). We prove that an attacker against our signature scheme can be used to solve either $\mathcal{P}_{\mathcal{S}}$ or $\mathcal{P}_{\mathcal{Z}}$ with a probability approximately equal to the attacker's success probability. More formally, we state the following theorem:

Theorem 1. *Let $\mathcal{A}$ be an adversary producing with success probability ε and within time bound τ an existential forgery of the proposed scheme $(\mathcal{Z}, \mathcal{S})$ under a chosen-message attack. Then, there is an algorithm that solves either $\mathcal{P}_{\mathcal{Z}}$ or $\mathcal{P}_{\mathcal{S}}$ with probability ε' in time τ' where*

$$\varepsilon' \geq \varepsilon - \frac{(q_{\mathcal{H}} + q_s)^2}{|H|} - \frac{(q_{\mathcal{G}} + q_s)^2}{|G|} - \frac{(q_{\mathcal{G}} + q_s) \cdot q_s}{|X|}$$

and

$$\tau' \leq \tau + q_s T_W + T_0,$$

after $q_{\mathcal{H}}$ queries to $\mathcal{H}$, $q_{\mathcal{G}}$ queries to $\mathcal{G}$ and q_s queries to the signing oracle respectively, noting $T_{\mathcal{W}}$ the time of evaluating $\mathcal{W}$ and T_0 the time needed by the $\mathcal{S}$ oracle to compute and send q_s random message-signature pairs.

Classically, we use the formalism of incremental games, starting with the real game $\mathbf{G}_0$ and ending up with $\mathbf{G}_6$. We are given a EUF-KMA-secure signature scheme $\mathcal{S}$ and a Σ-protocol $\mathcal{Z}$. The goal of our proof is to use an attacker against our scheme to solve one of the two problems $\mathcal{P}_{\mathcal{Z}}$ or $\mathcal{P}_{\mathcal{S}}$.

Game $\mathbf{G}_0$: This is the real attack game, in the random oracle model, which includes the verification step. This means that the attack game consists in giving the public key to the adversary, as well as a full access to the signing oracle. If a forgery is output, it is checked for validity. Note that the adversary is authorized to ask q_s queries to the signing oracle, $q_{\mathcal{H}}$ queries to the hash oracle $\mathcal{H}$ and $q_{\mathcal{G}}$ queries to the hash oracle $\mathcal{G}$. We are interested in the event S_0 which occurs if the verification step succeeds (and the signature was never returned by the signing oracle).

$$\mathsf{Succ}_{(\mathcal{Z},\mathcal{S})}^{\mathsf{euf-cma}}(\mathcal{A}) = \Pr[\mathsf{S}_0]. \tag{1}$$

Game $\mathbf{G}_1$: In this game, we simulate the hash oracle and the signing oracle, as well as the last verification step, as shown on Figure 2. From this simulation, we easily see that the game is perfectly indistinguishable from the real attack.

$$\Pr[\mathsf{S}_1] = \Pr[\mathsf{S}_0]. \tag{2}$$

Game $\mathbf{G}_2$:

In the next game, we use the $\mathcal{S}$ simulatability property. From the simulator $\mathcal{S}im$, we receive q_s pairs (r_i, σ_i) where the σ_i are valid signatures of r_i. This makes the game perfectly indistinguishable.

$$\Pr[\mathsf{S}_2] = \Pr[\mathsf{S}_1]. \tag{3}$$

Game $\mathbf{G}_3$:

In this new game, we perform the following step before running the attacker, and consequently before receiving any query from it. We generate q_s random pairs $(y_i, g_i) \in Y \times G$. Then, for each of them, we compute and store $u_i = \mathcal{W}(\alpha, g_i, y_i)$.

$\mathcal{H}$ oracle

For a hash-query $\mathcal{H}(q)$ such that $\exists h, (q, h) \in \mathsf{H\text{-}List}$, output h. Otherwise the output h is defined according to the following rule:

▶**Rule** $\mathcal{H}^{(1)}$

> Choose a random element $h \in H$. The record (q, h) is appended to $\mathsf{H\text{-}List}$.

$\mathcal{G}$ oracle

For a hash-query $\mathcal{G}(q)$, such that $\exists g, (q, g) \in \mathsf{G\text{-}List}$, output g. Otherwise the output g is defined according to the following rule:

▶**Rule** $\mathcal{G}^{(1)}$

> Choose a random element $g \in G$. The record (q, g) is appended to $\mathsf{G\text{-}List}$.

Sign oracle

For a sign-query **Sign**(m), we use the following rule:

▶**Rule Sign**$^{(1)}$

> One first generates a random $r \in R$, computes $u = \mathcal{U}(r)$ and $h = \mathcal{H}(u)$, and then computes $x = \mathsf{S}ig(d, h)$. A query to the simulation of the $\mathcal{G}$ oracle follows to obtain $g = \mathcal{G}(m, x)$. Finally one computes $y = \mathcal{V}(s, r, g)$. The output signature is then (x, y).

Verify oracle

The game ends with the verification of the output (x, y) of the adversary. One first asks to the oracle $g' = \mathcal{G}(m, x)$, then computes $u' = \mathcal{W}(\alpha, g', y)$ and $h' = \mathcal{H}(u')$. One then checks whether $\mathsf{V}er(\beta, x, h') \stackrel{?}{=} \textsc{True}$, in which case the signature is a valid signature of m. Once again, it is supposed that $\mathcal{G}$ and $\mathcal{W}$ are only defined on their respective set, verifying implicitly the fact that $x \in X$ and $y \in Y$. Otherwise this test is added in the verification step.

Fig. 2. Simulation of the Attack Game

Obviously, this maintains the game perfectly indistinguishable from the previous one:

$$\Pr[\mathsf{S}_3] = \Pr[\mathsf{S}_2]. \tag{4}$$

Game $\mathbf{G}_4$: In this game, we change the way we simulate the $\mathcal{H}$ oracle.

▶**Rule** $\mathcal{H}^{(4)}$

> – if the query q is equal to one of the u_i, we set $h = r_i$
> – otherwise we choose a random element $h \in H \backslash \{r_i\}$ with some probability χ and $h \in \{r_i\}$ with probability $(1 - \chi)$.
>
> The record (q, h) is appended to $\mathsf{H\text{-}List}$.

Parameter χ is chosen so that each element of H has an equal probability to be output. The evaluation of χ is not done here but trivially follows from simple considerations. As r_i are unknown to the attacker, this game is perfectly indistinguishable from the previous one.

$$\Pr[\mathsf{S}_4] = \Pr[\mathsf{S}_3]. \tag{5}$$

Game $\mathbf{G}_5$:
In this game, we number the queries to the signature oracle with some index i. From now, we are able to sign any message, as follows:

▶**Rule Sign**$^{(5)}$

> For the i-th query, if $\mathcal{G}(m, \sigma_i)$ is already defined, the game stops. Otherwise, $((m, \sigma_i), g_i)$ is appended to $\mathsf{G\text{-}List}$.
> Then the returned signature for message m is (σ_i, y_i).

As one may observe, the signature is valid: by definition, $\mathcal{G}(m, \sigma_i) = g_i$, $u_i = \mathcal{W}(\alpha, g_i, y_i)$, $\mathcal{H}(u_i) = r_i$ and σ_i is a valid signature of r_i.

This game is indistinguishable from the previous one, except that bad events may happen. More precisely, because σ_i can not be guessed by the attacker better than randomly (because it is the signature of a random element), the fact that $\mathcal{G}(m, \sigma_i)$ must not be defined introduces a factor $(1 - (q_\mathcal{G} + q_s)/|X|)^{q_s}$.

Hence, this game is such that:

$$\Pr[\mathsf{S}_5] \geq \Pr[\mathsf{S}_4] \cdot \left(1 - \frac{(q_\mathcal{G} + q_s)}{|X|}\right)^{q_s}. \tag{6}$$

Game $\mathbf{G}_6$: This game is the final one, in which we use a forge output by the attacker. By definition, after q_h hash queries and q_s signature queries, the attacker $\mathcal{A}$ is able to output a signature $(\hat{x}, \hat{y})$ of some message $\hat{m}$ with probability ε and within time τ. If the attacker succeeds, we show how to use the forge to break one of the two computational problems. If no forge is output, the game is aborted.

First of all, we compute $\hat{g} = \mathcal{G}(\hat{m}, \hat{x})$ and $\hat{u} = \mathcal{W}(\alpha, \hat{g}, \hat{y})$. If one among these hash values was never queried by the attacker, the adequate oracles are solicited to recover its output. Using the same technique, we set $\hat{h} = \mathcal{H}(\hat{u})$. Then, we have three cases, as explained on Fig. 3.

Finally, in the three cases, this game can be used to solve one of the two supposedly intractable problems with probability

$$\Pr[\mathsf{S}_6] \geq \Pr[\mathsf{S}_5] \cdot \left(1 - \frac{(q_\mathcal{H} + q_s)}{|H|}\right)^{q_\mathcal{H}+q_s} \cdot \left(1 - \frac{q_\mathcal{G} + q_s}{|G|}\right)^{q_\mathcal{G}+q_s}. \tag{7}$$

Combining previous equations, one can see that

$$\varepsilon' \geq \varepsilon \cdot \left(1 - \frac{(q_\mathcal{H} + q_s)^2}{|H|} - \frac{(q_\mathcal{G} + q_s)^2}{|G|} - \frac{(q_\mathcal{G} + q_s) \cdot q_s}{|X|}\right).$$

□

Case One: In the first case, $\hat{h}$ is not an r_i. Then, $(\hat{h}, \hat{x})$ is a valid forgery against the signature scheme $\mathcal{S}$.

Case Two: In the second case, $\hat{h} = r_i$ and $\hat{x} \neq \sigma_i$ for some i. Then $(\hat{h}, \hat{x})$ is a valid forgery against the signature. Remark that this case can happen only if the signature is a probabilistic signature.

Case Three: In the last case, $\hat{h} = r_i$ and $\hat{x} = \sigma_i$ for some i. Then with overwhelming probability we have $\hat{u} = u_i$, otherwise a collision on $\mathcal{H}$ has been found. Hence, $\hat{u} = u_i$ with probability greater than $(1 - q_{\mathcal{H}}/|H|)^{q_{\mathcal{H}}}$.
By definition, we then have:

$$\hat{u} = u_i = \mathcal{W}(\alpha, g_i, y_i) = \mathcal{W}(\alpha, \hat{g}, \hat{y}) \ .$$

As one can see, we are now in the hypothesis of the Forking Lemma, *but without having to restart the attacker in any sense*, contrarily to what is usually done when using the Forking Lemma. Consequently, as $\mathcal{Z}$ is a Σ-protocol, we are able from this equality to recover the secret, as soon as $(g_i, y_i) \neq (\hat{g}, \hat{y})$.
As the forged signature is a *new* signature, we must have $(\sigma_i, m_i, y_i) \neq (\hat{x}, \hat{m}, \hat{y})$, which in this case means that $(m_i, y_i) \neq (\hat{m}, \hat{y})$.

- If $m_i = \hat{m}$, we immediately have $(g_i, y_i) \neq (\hat{g}, \hat{y})$.
- Otherwise, if $m_i \neq \hat{m}$, as $g_i = \mathcal{G}(m_i, x_i)$ and $\hat{g} = \mathcal{G}(\hat{m}, \hat{x})$, we have either $g_i \neq \hat{g}$ or a collision on $\mathcal{G}$ has been found by the attacker. This can not happen with a factor greater than $(1 - (q_{\mathcal{G}} + q_s)/|G|)^{q_{\mathcal{G}}+q_s}$.

Hence with probability $(1 - (q_{\mathcal{H}} + q_s)/|H|)^{q_{\mathcal{H}}+q_s} \cdot (1 - (q_{\mathcal{G}} + q_s)/|G|)^{q_{\mathcal{G}}+q_s}$, the second case of this final game allows to recover the secret s.

Fig. 3. Breaking problems $\mathcal{P}_{\mathcal{Z}}$ or $\mathcal{P}_{\mathcal{S}}$.

4 An Instantiation of the Proposed Scheme

In this section, we give a typical example of a scheme based on our generic construction. This example relies on the RSA problem, and uses FDH-RSA as the signature $\mathcal{S}$ and Poupard-Stern as the zero-knowledge identification protocol $\mathcal{Z}$.

4.1 Poupard-Stern: a Σ-protocol Equivalent to Factoring

Poupard-Stern is a zero-knowledge identification protocol described in [22]. Its security relies on integer factorization. The scheme is described on Fig. 4.

Poupard-Stern uses as public key an RSA modulus n and a base a of maximal order modulo n, and the private key is $s = n - \varphi(n)$ of bitlength $\|s\| = \|n\|/2$.

- Prover P picks a random $r \in \{0, 2^{\Gamma} - 1\}$ and computes $x = a^r \bmod n$, which is sent to the verifier.
- The verifier verifies that $x \in \mathbb{Z}_n$ and sends a random $g \in \{0, 2^{\|g\|} - 1\}$ to the prover.
- The prover replies with $y = r + sg$, and the verifier verifies that $x = a^{y-ng} \bmod n$ and $0 \leq y < 2^{\Omega}$.

Fig. 4. The Poupard-Stern Zero-Knowledge Identification Protocol.

The set Y is a critical part for the security of the scheme. Indeed, an attacker could try to use $\hat{y} = r + ng$ instead of the legitimate y that she can not forge. Because $s \ll n$, it is possible to thwart such a forge by maximizing the authorized y. Furthermore, an attacker should not use a y (or even a collection of y that could have been logged) to recover a part of the secret s. Hence, r must be large enough to ensure that s is totally hidden within y, with respect to a security parameter[3] $\kappa = 80$.

All of this can be done by using Y as the set of positive integers smaller than 2^{Ω}, and R as the sets of positive integers smaller than 2^{Γ} with $\Gamma = \|g\| + \frac{\|n\|}{2} + \kappa$ and $\Omega = \Gamma + 1$. We refer the reader to [22] for a more accurate analysis of the Poupard-Stern protocol.

SECURITY. Poupard-Stern is a Σ-protocol: being given $(g_1, y_1) \neq (g_2, y_2)$ so that $a^{y_1 - ng_1} = a^{y_2 - ng_2}$, one can deduce the factorization of n and thus s. Indeed, $y_1 - ng_1 = y_2 - ng_2 \bmod \lambda(n)$, and consequently $(y_1 - y_2) - n(g_1 - g_2)$ is a multiple of $\lambda(n)$. As $y_1 \ll n$ and $y_2 \ll n$, it is a non-zero multiple of $\lambda(n)$. Then, using Miller's algorithm [16], one can recover the factors of n in polynomial time.

[3] *i.e.*, the legitimate y is statistically indistinguishable from a random of the same size, where the statistical distance is controlled by the security parameter κ.

4.2 A Signature Scheme with Coupons and Tight Reduction to RSA

As explained earlier, coupons are not supported when using RSA and its numerous variants, including in this respect Guillou-Quisquater [14]. The scheme we now propose uses RSA and Poupard-Stern as instances of systems $\mathcal{S}$ and $\mathcal{Z}$ and fully supports coupons.

Our scheme makes use of an RSA modulus n of secret factorization which is common to the zero-knowledge scheme and the signature scheme. Like with RSA, the key pair contains a public and a private exponent e and d such that $ed = 1 \bmod \lambda(n)$. The integer $s = n - \varphi(n)$ is kept secret as in Poupard-Stern.

This scheme is described as follows.

Key: The public key is (e, n, a) while the private key is (s, d).

Signature: To sign a message $m \in \mathcal{M}$, one randomly chooses $r \in \{0, 2^{\Gamma} - 1\}$ and computes $u = a^r \bmod n$, $h = \mathcal{H}(u)$ and $x = h^d \bmod n$. Upon receiving the message m, one computes $g = \mathcal{G}(m, x)$ and $y = r + sg$.
The signature on m is $\sigma = (x, y)$.

Verification: To verify a signature $\sigma = (x, y)$, one computes $g' = \mathcal{G}(m, x)$, $u' = a^{y - ng'} \bmod n$ and $h' = \mathcal{H}(u')$. Finally, the signature σ is accepted iff $h' \stackrel{?}{=} x^e \bmod n$ and $0 \leq y < 2^{\Omega}$ and $0 \leq x < n$.

As one can notice, we can not use directly the theorem of the previous section, as the $\mathcal{H}$ function of the general scheme and the full-domain hash function of the FDH-RSA signature have been combined. The complete proof of such a particular but interesting case is given in appendix. The theorem stated there proves that in case of self-reducibility over the signature domain (as it is the case with FDH-RSA), one can combine the $\mathcal{H}$ function of the general scheme and the full-domain hash function of the FDH signature while keeping the security tightly equivalent to problems $\mathcal{P}_{\mathcal{S}}$ and $\mathcal{P}_{\mathcal{Z}}$.

In our case, $\mathcal{P}_{\mathcal{S}}$ is the RSA (*i.e.*, e-th root extraction) problem, while $\mathcal{P}_{\mathcal{Z}}$ is the factoring problem. As the RSA problem is easier than factoring, one can deduce that our proposed scheme is tightly equivalent to the RSA problem.

Hence, our scheme is as secure as RSA-PSS but also presents the practical advantage of allowing coupons. Using these, generating a complete signature is as fast as Poupard-Stern's second step: namely, we require to perform a multiplication, an addition and a hash computation, consequently resulting in a much faster procedure than carrying out a modular exponentiation with RSA-PSS. Again, this speed-up is not at the cost of a loose reduction, as it is the case with the non-interactive version of the original Poupard-Stern, which proof makes use of the Forking Lemma in a classical way.

A thorough comparison between RSA-PSS and our proposed scheme is provided in the following subsection.

4.3 Comparison of Our Scheme with RSA-PSS

RSA-PSS is described as a signature standard [19] and is extensively used worldwide, although not as fast as signature schemes with coupons. We compare our RSA-based scheme with RSA-PSS using a modulus of $\|n\| \geq 1024$ bits.

SIZE OF HASHES. Because of Th. 1, one can see that a hash function $\mathcal{G}$ with an output size of $\|G\| = 160$ bits is sufficient to resist an attacker allowed to make 2^{79} queries to the $\mathcal{G}$ oracle. Indeed, as $\mathcal{H}$ is by definition a full-domain hash, we have that $\|H\| = \|n\|$ and by construction, $\|X\| = \|n\|$.

SIZE OF PARAMETERS. In RSA-PSS, the public parameters are n and e (e is usually short), and the private key is d. In our proposed scheme, the public key is formed by a, e and n, while in the basic presentation of the scheme, the private key is d and s. However, one can see that we can take a small value for a (the only property we require is that a is of maximal order). Furthermore, d and s are redundant secrets, and one can easily compress them, if the size of the private key elements is more important than the execution time of the signature. Using d and $k_d = \frac{ed-1}{\varphi(n)}$ of bitsize $\|e\|$, one can recover $s = n - \frac{ed-1}{k_d}$ and then use it to sign.

Overall, the public keys have roughly the same size in RSA-PSS and in our scheme. The private key in our scheme is 50% longer than the private key in RSA-PSS, but one can compress this private key at the cost of an additional division by a small number during the generation of a coupon. In this last case, the size of both private keys are almost equivalent.

SIGNATURE SIZE. An RSA-PSS signature has $\|n\|$ bits while the size of our signature is $\|x\| + \|y\| = \|n\| + \|g\| + \frac{\|n\|}{2} + \kappa = \frac{3\|n\|}{2} + 240$. Hence, there is an advantage for RSA-PSS on this point, even if the proposed signature is in fact not quite twice as long as a RSA-PSS signature.

PERFORMANCES OF SIGNATURE GENERATION. Our scheme, very much like RSA-PSS, supports the Chinese Reminder Theorem [23] thereby allowing implementations with improved efficiency.

As mentioned earlier, our scheme is useful essentially if used with coupons. When this is the case, the comparison of execution times makes our scheme very appealing. RSA-PSS requires hash computations and a modular exponentiation of $\|n\|$ bits: using a modular exponentiation of complexity $2 \geq c \geq 1$, this requires $c\|n\|$ modular multiplications of $\|n\|$ bits if hash computations are neglected. On the other hand, our scheme with coupons is as fast as the last step in Poupard-Stern, *i.e.*, one hash computation and one integer multiplication of $\|n\|$ bits times 160 bits. Clearly, our scheme (*i.e.*, the online part of our signature) is more than $\|n\|$ times faster than RSA-PSS.

EASE OF IMPLEMENTATION. Most importantly, our scheme and RSA-PSS are similarly simple to implement, as they both make use of the same routines (modular exponentiation, hash functions). There is just an extra addition and integer multiplication in our scheme, and coding this remains a very simple operation.

SECURITY. On this point, both schemes are equivalent since equally tightly related to the RSA problem.

Conclusion. Hence, as far as the signature size is not a bottleneck, our scheme could be preferred over RSA-PSS as it allows a dramatic improvement in terms of performance for an equivalent security.

4.4 Other Signatures Based on Other Problems

As shown in the Section 3, our scheme is generic and the same technique can be applied to a large variety of signatures schemes and zero-knowledge identification protocols. The most interesting combinations seem however to be the ones where the underlying problems are the same in the two components, as in the example we proposed above where unforgeability is entirely based on the RSA problem.

There exist however other combinations that we do not explicit here, such as combining a signature scheme due to Goh and Jarecki [11], which is tightly equivalent to the Diffie-Hellman problem, with Schnorr's zero-knowledge protocol, which is a Σ-protocol proven secure under the discrete logarithm assumption. As the discrete log problem is at least as hard as the Diffie-Hellman problem, the combination of these systems gives a signature scheme tightly equivalent to the Diffie-Hellman problem, and that also supports coupons.

Another example is a combination of the Poupard-Stern Σ-protocol that we did already use in our RSA-based scheme and of the FDH-Rabin signature [24], which is a EUF-KMA signature scheme. Such a combination gives a scheme equivalent to integer factoring under a tight reduction which supports coupons as well.

We believe that other combinations of great interest are made possible with our construct.

5 Conclusion

In this paper, we proposed a new generic signature scheme constructed from a zero-knowledge identification protocol and a signature scheme. This new scheme features a tight provable security relatively to the problems which underly the security of its components. In addition to that, our scheme enjoys the appealing property of enabling the use of coupons. These two advantages were never before proposed together in a signature scheme. We also proposed an efficient application of our technique using the RSA problem which provides a high-speed alternative to RSA-PSS.

Acknowledgements

The author would like to thank his careful PhD advisor, David Pointcheval (ENS-CNRS), for teaching him so much about provable security. Many thanks also go to Marc Joye and Pascal Paillier for their attention and fruitful support in our research. Finally, the author thanks Jean-François Dhem, Philippe Proust and David Naccache.

References

1. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *1st ACM Conference on Computer and Communications Security*, pp. 62–73. ACM Press, 1993.
2. M. Bellare and P. Rogaway. The exact security of digital signatures - How to sign with RSA and Rabin. In *Advances in Cryptology – EUROCRYPT '96*, vol. 1070 of *LNCS*, pp. 399–416. Springer-Verlag, 1996.
3. B. Chevallier-Mames. New signature schemes with coupons and tight reduction. Full version available from http://www.di.ens.fr/users/pointche/Chevallier-Mames.
4. J.-S. Coron. Optimal security proofs for PSS and other signature schemes. In *Advances in Cryptology – EUROCRYPT '02*, vol. 2332 of *LNCS*, pp. 272–287. Springer-Verlag, 2002.
5. R. Cramer and V. Shoup. Signature scheme based on the strong RSA assumption. Theory of Cryptography Library, 99-01, January 1999.
6. S. Even, O. Goldreich, and S. Micali. On-line/Off-line digital signatures. In *Advances in Cryptology – CRYPTO '89*, vol. 435 of *LNCS*, pp. 263–277. Springer-Verlag, 1990.
7. U. Feige, A. Fiat, and A. Shamir. Zero-knowledge proofs of identity. *Journal of Cryptology*, 1(2):77–94, 1988.
8. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology – Crypto '86*, vol. 263 of *LNCS*, pp. 186–194. Springer-Verlag, 1987.
9. R. Gennaro, S. Halevi, and T. Rabin. Secure hash-and-sign signatures without the random oracle. In *Advances in Cryptology – EUROCRYPT '99*, vol. 1592 of *LNCS*, pp. 123–139. Springer-Verlag, 1999.
10. M. Girault. An identity-based identification scheme based on discrete logarithms modulo a composite number. In *Advances in Cryptology – EUROCRYPT '90*, vol. 473 of *LNCS*, pp. 481–486. Springer-Verlag, 1991.
11. E.-J. Goh and S. Jarecki. A signature scheme as secure as the Diffie-Hellman problem. In *Advances in Cryptology – EUROCRYPT '03*, LNCS, pp. 401–415. Springer-Verlag, 2003.
12. S. Goldwasser, S. Micali, and R. Rivest. A "paradoxical" solution to the signature problem. In *Proceedings of the 25th FOCS*, pp. 441–448. IEEE, 1984.
13. S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen message attacks. *SIAM Journal of Computing*, 17(2):281–308, 1988.
14. L.C. Guillou and J.-J. Quisquater. A practical zero-knowledge protocol fitted to security processor minimizing both transmission and memory. In *Advances in Cryptology – EUROCRYPT '88*, vol. 330 of *LNCS*, pp. 123–128. Springer-Verlag, 1988.
15. H. Krawczyk and T. Rabin. Chameleon signatures. In *Symposium on Network and Distributed System Security – NDSS '00*, pp. 143–154. Internet Society, 2000.
16. G.L. Miller. Riemann's hypothesis and tests for primality. *Journal of Computer and System Sciences*, pp. 300–317, 1976.
17. A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone. *Handbook of applied cryptography.* CRC Press, 1997.
18. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology – EUROCRYPT '99*, vol. 1592 of *LNCS*, pp. 223–238. Springer-Verlag, 1999.

19. PKCS #1 v2.1: RSA cryptography standard. RSA Laboratories, June 14, 2002.
20. D. Pointcheval and J. Stern. Security proofs for signature schemes. In *Advances in Cryptology – EUROCRYPT '96*, vol. 1070 of *LNCS*, pp. 387–398. Springer-Verlag, 1996.
21. G. Poupard and J. Stern. Security analysis of a practical "on the fly" authentication and signature generation. In *Advances in Cryptology – EUROCRYPT '98*, vol. 1403 of *LNCS*, pp. 422–436. Springer-Verlag, 1998.
22. G. Poupard and J. Stern. On the fly signatures based on factoring. *ACM Conference on Computer and Communications Security*, pp. 37–45, 1999.
23. J.-J. Quisquater and C. Couvreur. Fast decipherment algorithm for RSA public-key cryptosystem. *Electronics Letters*, 18:905–907, 1982.
24. M.O. Rabin. Digital signatures and public-key functions as intractable as factorization. Tech. Rep. MIT/LCS/TR-212, MIT Laboratory for Computer Science, 1979.
25. R.L. Rivest, A. Shamir, and L.M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.
26. C.-P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.
27. A. Shamir and Y. Tauman. Improved online/offline signature schemes. In *Advances in Cryptology – CRYPTO '01*, vol. 2139 of *LNCS*, pp. 355–367. Springer-Verlag, 2001.

A Proof of Security When $\mathcal{S}$ Has a Self-Reducible Domain

A trapdoor permutation is said *self-reducible* with respect to polynomial functions $SR_1 : \mathcal{M} * R \mapsto \mathcal{M}$ and $SR_2 : X * R \mapsto X$ when for a uniformly distributed random element $r \in R$, $m' = SR_1(m, r)$ is uniformly distributed in the message space $\mathcal{M}$, and when the pre-image $\sigma = \mathcal{F}^{-1}(m)$ of m can be recovered from $\sigma' = \mathcal{F}^{-1}(m')$ by computing $\sigma = SR_2(\sigma', r)$.

Examples of trapdoor self-reducible permutations are RSA [25] and Paillier [18].[4]

We remind that a FDH self-reducible signature of a message $m \in M$, using a full-domain hash $\mathcal{H}_{\mathrm{FDH}}$ and a trapdoor self-reducible permutation $\mathcal{F}$ consists in computing $\sigma = \mathcal{F}^{-1}(\mathcal{H}_{\mathrm{FDH}}(m))$. The verification is just checking that $\mathcal{F}(\sigma) = \mathcal{H}_{\mathrm{FDH}}(m)$.

In case $\mathcal{S}$ has a self-reducible domain, our generic construct can be slightly modified in order to mix functions $\mathcal{H}$ and $\mathcal{H}_{\mathrm{FDH}}$ as shown of Fig. 5.

As before, we prove that the scheme is secure in the random oracle model, even if the attacker is given the signature of q_s messages of her choice, as well as q_h hash computations (more precisely $q_{\mathcal{H}}$ and $q_{\mathcal{G}}$ queries to $\mathcal{H}$ and $\mathcal{G}$ respectively). We show that an attacker forging our scheme can be used to find the pre-image

[4] For the RSA function, $SR_1(m, r) = m \cdot r^e \bmod n$ and $SR_2(\sigma', r) = \sigma \cdot r^{-1} \bmod n$, while for Paillier, $SR_1(m, (r_1, r_2)) = m \cdot g^{r_1} \cdot {r_2}^n \bmod n^2$ and $SR_2((a, b), (r_1, r_2)) = (a - r_1 \bmod n, b - r_2 \bmod n)$, with obvious notations.

Sign(m) :	**Verify**(x, y, m) :
Step 1: $r \leftarrow_R$ $u \leftarrow \mathcal{U}(r)$ $h \leftarrow \mathcal{H}(u)$ $x \leftarrow \mathcal{F}^{-1}(h)$ *Step 2:* $g \leftarrow \mathcal{G}(m, x)$ $y \leftarrow \mathcal{V}(s, r, g)$ **Return** (x, y)	$g' \leftarrow \mathcal{G}(m, x)$ $u' \leftarrow \mathcal{W}(\alpha, g', y)$ $h' \leftarrow \mathcal{H}(u')$ **Return** $x \stackrel{?}{=} \mathcal{F}(h')$

Fig. 5. Our Proposed Signature Scheme with a FDH self-reducible signature

by $\mathcal{F}$ of a challenge $c \in X$ (a problem that we call $\mathcal{P}_{\mathcal{F}^{-1}}$) or to find the secret s of the zero-knowledge identification protocol (we refer to this problem as $\mathcal{P}_{\mathcal{Z}}$), with probability approximately equal to the attacker's. More formally, we state the following theorem.

Theorem 2. *Let $\mathcal{A}$ be an adversary producing with success probability ε and within a time bound τ an existential forgery of the proposed scheme $(\mathcal{Z}, \mathcal{F})$, based on problems $\mathcal{P}_{\mathcal{Z}}$ and $\mathcal{P}_{\mathcal{F}^{-1}}$, under a chosen-message attack and making q_s queries to the signing oracle and $q_{\mathcal{H}}, q_{\mathcal{G}}$ queries to $\mathcal{H}$ and $\mathcal{G}$ respectively. Then, with probability ε', the attacker can be used to break one of the two problem $\mathcal{P}_{\mathcal{Z}}$ and $\mathcal{P}_{\mathcal{F}^{-1}}$ within time τ' with*

$$\varepsilon' \geq \varepsilon \left(1 - \frac{(q_{\mathcal{G}} + q_s)^2}{|G|} - \frac{(q_{\mathcal{G}} + q_s) \cdot q_s}{|X|} \right)$$

and

$$\tau' \leq \tau + q_s(T_W + T_{\mathcal{F}})$$

where $T_{\mathcal{W}}$ is the time needed for evaluating $\mathcal{W}$ and $T_{\mathcal{F}}$ the execution time required to compute $\mathcal{F}$.

Our proof starts at the real game $\mathbf{G}_0$ and ends with $\mathbf{G}_5$. We are given a trapdoor self-reducible permutation $\mathcal{F}$ and a Σ-protocol $\mathcal{Z}$, as well as a random challenge $c \in X$. Self-reducible functions SR_1 and SR_2 are given together with the self-reducible permutation. The goal of our proof is to use an attacker against our scheme to solve one of the two following problems: finding the secret s of $\mathcal{Z}$, or computing the pre-image $\mathcal{F}^{-1}(c)$.

Game $\mathbf{G}_0$: This is the real attack game, in the random permutation model, which includes the verification step. This means that the attack game consists in giving the public key to the adversary, and a full access to the signing oracle. If it outputs its forgery, one checks whether the forgery is valid or not. Note that the adversary is authorized to ask q_s queries to the signing oracle and q_h queries to the hash oracles $\mathcal{G}$. We are interested in the following event: S_0 which occurs if the verification step succeeds (and the signature is new).

$$\mathsf{Succ}^{\mathsf{euf-cma}}_{(\mathcal{Z},\mathcal{F}^{-1})}(\mathcal{A}) = \Pr[\mathsf{S}_0]. \tag{8}$$

Game $\mathbf{G}_1$: In this game, we simulate the hash oracle and the signing oracle, and the last verification step, as shown on Figure 6. From this simulation, we easily see that the game is perfectly indistinguishable from the real attack, in the random oracle model.

$$\Pr[\mathsf{S}_1] = \Pr[\mathsf{S}_0]. \tag{9}$$

$\mathcal{H}$ oracle	For a hash-query $\mathcal{H}(q)$, such that $\exists h$, so that $(q,h) \in$ H-List, the answer is h. Otherwise the answer h is defined according to the following rule: ▶**Rule** $\mathcal{H}^{(1)}$ Choose a random element $h \in H$. The record (q,h) is added to H-List.
$\mathcal{G}$ oracle	For a hash-query $\mathcal{G}(q)$, such that $\exists g$, so that $(q,g) \in$ G-List, the answer is g. Otherwise the answer g is defined according to the following rule: ▶**Rule** $\mathcal{G}^{(1)}$ Choose a random element $g \in G$. The record (q,g) is added to G-List.
Sign oracle	For a sign-query **Sign**(m), in this game, the sequence strictly follows the definition of the proposed scheme: ▶**Rule Sign**$^{(1)}$ One first picks a random $r \in R$, computes $u = \mathcal{U}(r)$, then computes $h = \mathcal{H}(u)$ and $x = \mathcal{F}^{-1}(h)$. Follows a query to the $\mathcal{G}$ oracle to know $g = \mathcal{G}(m,x)$, and finally one computes $y = \mathcal{V}(s,r,g)$. The signature is (x,y).
Verify oracle	The game ends with the verification of the output (x,y) from the adversary. One first asks to the oracle $g' = \mathcal{G}(m,x)$ and then computes $u' = \mathcal{W}(\alpha, g', y)$. Then one computes $h' = \mathcal{H}(u')$ and checks whether $\mathsf{V}er(\beta, x, h') \stackrel{?}{=}$ TRUE, in which case the signature is a valid signature of m. Once again, it is supposed that $\mathcal{G}$ and $\mathcal{W}$ are only defined on their respective set, verifying implicitly the fact that $x \in X$ and $y \in Y$. Otherwise, this test is made within the verification step.

Fig. 6. Simulation of the Attack Game .

Game $\mathbf{G}_2$:

In this new game, we do the following step before running the attacker, and so before receiving any query from her. We pick q_s random pairs $(y_i, g_i) \in Y \times G$. Then for each of them, we compute $u_i = \mathcal{W}(\alpha, g_i, y_i)$. For each and every u_i, we pick a random element $\rho_i \in X$ and compute $h_i = \mathcal{F}(\rho_i)$. We store these (*i.e.*, y_i, g_i, u_i, ρ_i) in memory.

This makes the game indistinguishable from the previous one, as $\mathcal{F}$ is a permutation and the ρ_i are random:

$$\Pr[\mathsf{S}_2] = \Pr[\mathsf{S}_1]. \tag{10}$$

Game $\mathbf{G}_3$: In this game, we change the way we simulate the $\mathcal{H}$ oracle.

►**Rule** $\mathcal{H}^{(3)}$

- if the query q is equal to a u_i, we set $h = h_i$
- otherwise we choose a random element $\rho_j \in X$ and compute $h = SR_1(c, \rho_j)$. We keep each ρ_j associated with each query q in a memory.

The record (q, h) is added to H-List.

One can note that ρ_i is a valid FDH signature of u_i.

Because of the permutation property of $\mathcal{F}$, of the randomness of c, h_i and ρ, and of the self-reducibility of $\mathcal{F}$, this game is equivalent to the previous one:

$$\Pr[\mathsf{S}_3] = \Pr[\mathsf{S}_2]. \tag{11}$$

Game $\mathbf{G}_4$:

In this game, we number the query to signature oracle with index i. From now, we are able to sign any message, as follows:

►**Rule Sign**$^{(4)}$

At the i-th query, if $\mathcal{G}(m, \rho_i)$ is already defined, the game stops. Otherwise, $((m, \rho_i), g_i)$ is added to G-List.
Then the returned signature of the message m is (ρ_i, y_i).

As one can see, the signature is valid: by definition, $\mathcal{G}(m, \rho_i) = g_i$, $u_i = \mathcal{W}(\alpha, g_i, y_i)$ and $\mathcal{H}(u_i) = h_i = \mathcal{F}(\rho_i)$.

This game is indistinguishable from the previous one, except that abortions may happen. More precisely, because ρ_i can not be guessed by the attacker better than randomly (because it is the permutation image of a random element), the fact that $\mathcal{G}(m, \rho_i)$ must not be defined introduces a factor $(1 - (q_{\mathcal{G}} + q_s)/|X|)^{q_s}$.

Hence, this games is such that:

$$\Pr[\mathsf{S}_4] \geq \Pr[\mathsf{S}_3] \cdot \left(1 - \frac{(q_{\mathcal{G}} + q_s)}{|X|}\right)^{q_s}. \tag{12}$$

Game $\mathbf{G}_5$: This game is the final one, in which we use the adversary's forgery. By definition, after q_h hash queries, and q_s signature queries, the attacker $\mathcal{A}$ is able to forge a *new* signature $(\hat{x}, \hat{y})$ of a message $\hat{m}$ with some probability. We now use this forgery to break one of the two supposed-hard problems.

First of all, we compute $\hat{g} = \mathcal{G}(\hat{m}, \hat{x})$ and $\hat{u} = \mathcal{W}(\alpha, \hat{g}, \hat{y})$. Finally, we set $\hat{h} = \mathcal{H}(\hat{u})$. If one of these hashes were never queried by the attacker, we ask them to the oracles. Then, two cases appear:

Case One: In the first case, $\hat{u}$ is not an u_i. Then, by definition, $\hat{h} = SR_1(c, \rho_j)$ and $\hat{x}$ is the valid $\mathcal{F}$-pre-image of $\hat{h}$. Because of the self-reducibility property, one can with a call to SR_2 function, find the pre-image $\mathcal{F}^{-1}(c)$:

$$\mathcal{F}^{-1}(c) = SR_2(\hat{x}, \rho_j)$$

Case Two: In the second case, $\hat{u} = u_i$, for a certain i, which means that $\mathcal{W}(\alpha, \hat{g}, \hat{y}) = \mathcal{W}(\alpha, g_i, y_i)$. As one can see, we are now in the hypothesis of the Forking Lemma, *but without having to restart the attacker in any sense*, contrarily to what is classically done when using the Forking Lemma. Consequently, as $\mathcal{Z}$ is a Σ-protocol, we are able to recover the secret from this equality, as soon as $(g_i, y_i) \neq (\hat{g}, \hat{y})$.
As $\hat{u} = u_i$, we have necessarily $\hat{x} = x_i$, as the FDH signature is a deterministic signature. Furthermore, as the forged signature is a *new* signature, we must have $(x_i, m_i, y_i) \neq (\hat{x}, \hat{m}, \hat{y})$, which in this case means that $(m_i, y_i) \neq (\hat{m}, \hat{y})$.

- If $m_i = \hat{m}$, we immediately have $(g_i, y_i) \neq (\hat{g}, \hat{y})$.
- Otherwise, if $m_i \neq \hat{m}$, as $g_i = \mathcal{G}(m_i, x_i)$ and $\hat{g} = \mathcal{G}(\hat{m}, \hat{x})$, we have that $g_i \neq \hat{g}$ or a collision on $\mathcal{G}$ has been found by the attacker. Because of the security of the $\mathcal{G}$ function, it can not happen with a factor greater than $(1 - (q_\mathcal{G} + q_s)/|G|)^{(q_\mathcal{G}+q_s)}$.

Hence, with this probability, this second case of this final game allows to recover the secret s.

Finally, in both cases, this game can be used to solve one of the two computational problems with probability:

$$\Pr[\mathsf{S}_5] \geq \Pr[\mathsf{S}_4] \cdot \left(1 - \frac{(q_\mathcal{G} + q_s)}{|G|}\right)^{q_\mathcal{G}+q_s} . \tag{13}$$

Combining previous equations, one can see that

$$\varepsilon' \geq \varepsilon \cdot \left(1 - \frac{(q_\mathcal{G} + q_s)^2}{|G|} - \frac{(q_\mathcal{G} + q_s) \cdot q_s}{|X|}\right) .$$

□