Optimal Security Proofs for PSS and other Signature Schemes

Jean-Sébastien Coron
Gemplus Card International
34 rue Guynemer
Issy-les-Moulineaux, F-92447, France
coron@ens.fr

Abstract. The Probabilistic Signature Scheme (PSS) designed by Bellare and Rogaway is a signature scheme provably secure against chosen message attacks in the random oracle model, whose security can be tightly related to the security of RSA. We derive a new security proof for PSS in which a much shorter random salt is used to achieve the same security level, namely we show that $\log_2 q_{sig}$ bits suffice, where q_{sig} is the number of signature queries made by the attacker. When PSS is used with message recovery, a better bandwidth is obtained because longer messages can now be recovered. In this paper, we also introduce a new technique for proving that the security proof of a signature scheme is optimal. In particular, we show that the size of the random salt that we have obtained for PSS is optimal: if less than $\log_2 q_{sig}$ bits are used, then PSS is still provably secure but it cannot have a tight security proof. Our technique applies to other signature schemes such as the Full Domain Hash scheme and Gennaro-Halevi-Rabin's scheme, whose security proofs are shown to be optimal.

Key-words: Probabilistic Signature Scheme, Provable Security.

1 Introduction

Since the invention of public-key cryptography in the seminal Diffie-Hellman paper [9], significant research endeavors were devoted to the design of practical and provably secure schemes. A proof of security is usually a computational reduction from solving a well established problem to breaking the cryptosystem. Well established problems of cryptographic relevance include factoring large integers, computing discrete logarithms in prime order groups, or extracting roots modulo a composite integer.

For digital signature schemes, the strongest security notion was defined by Goldwasser, Micali and Rivest in [13], as *existential unforgeability under an adaptive chosen message attack*. This notion captures the property that an attacker cannot produce a valid signature, even after obtaining the signature of (polynomially many) messages of his choice.

Goldwasser, Micali and Rivest proposed in [13] a signature scheme based on signature trees that provably meets this definition. The efficiency of the scheme was later improved by Dwork and Naor [10], and Cramer and Damgård [7]. A significant drawback of those signature schemes is that the signature of a message depends on previously signed messages: the signer must thus store information relative to the signatures he generates as time goes by. Gennaro, Halevi and Rabin presented in [12] a new hash-and-sign scheme provably secure against adaptive chosen message attacks which is both state-free and efficient. Its security is based on the strong-RSA assumption. Cramer and Shoup presented in [8] a signature scheme provably secure against adaptive chosen message attacks, which is also state-free, efficient, and based on the strong-RSA assumption.

The random oracle model, introduced by Bellare and Rogaway in [1], is a theoretical framework allowing to prove the security of hash-and-sign signature schemes. In this model, the hash function is seen as an oracle that outputs a random value for each new query. Bellare and Rogaway defined in [2] the Full Domain Hash (FDH) signature scheme, which is provably secure in the random oracle model assuming that inverting RSA is hard. [2] also introduced the Probabilistic Signature Scheme (PSS), which offers better security guarantees than FDH. Similarly, Pointcheval and Stern [19] proved the security of discrete-log based signature schemes in the random oracle model (see also [16] for a concrete treatment). However, security proofs in the random oracle are not real proofs, since the random oracle is replaced by a well defined hash function in practice; actually, Canetti, Goldreich and Halevi [4] showed that a security proof in the random oracle model does not necessarily imply that a scheme is secure in the real world.

For practical applications of provably secure schemes, the tightness of the security reduction must be taken into account. A security reduction is tight when breaking the signature scheme leads to solving the well established problem with probability close to one. In this case, the signature scheme is almost as secure as the well established problem. On the contrary, if the above probability is too small, the guarantee on the signature scheme will be weak; in which case larger security parameters must be used, thereby decreasing the efficiency of the scheme.

The security reduction of [2] for Full Domain Hash bounds the probability ε of breaking FDH in time t by $(q_{hash} + q_{sig}) \cdot \varepsilon'$ where ε' is the probability of inverting RSA in time t' close to t and where q_{hash} and q_{sig} are the number of hash queries and signature queries performed by the forger. This was later improved in [5] to $\varepsilon \simeq q_{sig} \cdot \varepsilon'$, which is a significant improvement since in practice q_{sig} happens to be much smaller than q_{hash}. However, FDH's security reduction is still not tight, and FDH is still not as secure as inverting RSA.

On the contrary, PSS is almost as secure as inverting RSA ($\varepsilon \simeq \varepsilon'$). Additionally, for PSS to have a tight security proof in [2], the random salt used to generate the signature must be of length at least $k_0 \simeq 2 \cdot \log_2 q_{hash} + \log_2 1/\varepsilon'$, where q_{hash} is the number of hash queries requested by the attacker and ε' the probability of inverting RSA within a given time bound. Taking $q_{hash} = 2^{60}$

and $\varepsilon' = 2^{-60}$ as in [2], we obtain a random salt of size $k_0 = 180$ bits. In this paper, we show that PSS has actually a tight security proof for a random salt as short as $\log_2 q_{sig}$ bits, where q_{sig} is the number of signature queries made by the attacker. For example, for an application in which at most one billion signatures will be generated, $k_0 = 30$ bits of random salt are actually sufficient to guarantee the same level of security as RSA, and taking a longer salt *does not* increase the security level. When PSS is used with message recovery, we obtain a better bandwidth because a larger message can now be recovered when verifying the signature.

Moreover, we show that this size is optimal: if less than $\log_2 q_{sig}$ bits of random salt are used, PSS is still provably secure, but PSS cannot have exactly the same security level as RSA. First, using a new technique, we derive an upper bound for the security of FDH, which shows that the security proof in [5] with $\varepsilon \simeq q_{sig} \cdot \varepsilon'$ is optimal. In other words, it is not possible to further improve the security proof of FDH in order to obtain a security level equivalent to RSA. This answers the open question raised by Bellare and Rogaway in [2], about the existence of a better security proof for FDH: as opposed to PSS, FDH *cannot* be proven as secure as inverting RSA. The technique also applies to other signature schemes such as Gennaro-Halevi-Rabin's scheme [12] and Paillier's signature scheme [17]. To our knowledge, this is the first result concerning optimal security proofs. Then, using the upper bound for the security of FDH, we show that our size k_0 for the random salt in PSS is optimal: if less than $\log_2 q_{sig}$ bits are used, no security proof for PSS can be tight.

2 Definitions

In this section we briefly present some notations and definitions used throughout the paper. We start by recalling the definition of a signature scheme.

Definition 1 (signature scheme). *A signature scheme* $(\texttt{Gen}, \texttt{Sign}, \texttt{Verify})$ *is defined as follows:*

- The key generation algorithm `Gen` *is a probabilistic algorithm which given* 1^k*, outputs a pair of matching public and private keys,* (pk, sk)*.*

- The signing algorithm `Sign` *takes the message* M *to be signed, the public key* pk *and the private key* sk*, and returns a signature* $x = \texttt{Sign}_{pk,sk}(M)$*. The signing algorithm may be probabilistic.*

- The verification algorithm `Verify` *takes a message* M*, a candidate signature* x' *and* pk*. It returns a bit* $\texttt{Verify}_{pk}(M, x')$*, equal to one if the signature is accepted, and zero otherwise. We require that if* $x \leftarrow \texttt{Sign}_{pk,sk}(M)$*, then* $\texttt{Verify}_{pk}(M, x) = 1$*.*

In the previously introduced *existential unforgeability under an adaptive chosen message attack* scenario, the forger can dynamically obtain signatures of messages of his choice and attempts to output a valid forgery. A *valid forgery* is a message/signature pair (M, x) such that $\texttt{Verify}_{pk}(M, x) = 1$ whereas the signature of M was never requested by the forger.

A significant line of research for proving the security of signature schemes is the previously introduced random oracle model, where resistance against adaptive chosen message attacks is defined as follows [1]:

Definition 2. *A forger $\mathcal{F}$ is said to $(t, q_{hash}, q_{sig}, \varepsilon)$-break the signature scheme* (`Gen`, `Sign`, `Verify`) *if after at most $q_{hash}(k)$ queries to the hash oracle, $q_{sig}(k)$ signatures queries and $t(k)$ processing time, it outputs a valid forgery with probability at least $\varepsilon(k)$ for all $k \in \mathbb{N}$.*

and quite naturally:

Definition 3. *A signature scheme* (`Gen`, `Sign`, `Verify`) *is $(t, q_{sig}, q_{hash}, \varepsilon)$-secure if there is no forger who $(t, q_{hash}, q_{sig}, \varepsilon)$-breaks the scheme.*

The RSA cryptosystem, invented by Rivest, Shamir and Adleman [20], is the most widely used cryptosystem today:

Definition 4 (The RSA cryptosystem). *The RSA cryptosystem is a family of trapdoor permutations, specified by:*
- The RSA generator $\mathcal{RSA}$, which on input 1^k, randomly selects two distinct $k/2$-bit primes p and q and computes the modulus $N = p \cdot q$. It randomly picks an encryption exponent $e \in \mathbb{Z}^*_{\phi(N)}$ and computes the corresponding decryption exponent d such that $e \cdot d = 1 \mod \phi(N)$. The generator returns (N, e, d).
- The encryption function $f : \mathbb{Z}^*_N \to \mathbb{Z}^*_N$ defined by $f(x) = x^e \mod N$.
- The decryption function $f^{-1} : \mathbb{Z}^*_N \to \mathbb{Z}^*_N$ defined by $f^{-1}(y) = y^d \mod N$.

FDH was the first practical and provably secure signature scheme based on RSA. It is defined as follows: the key generation algorithm, on input 1^k, runs $\mathcal{RSA}(1^k)$ to obtain (N, e, d). It outputs (pk, sk), where the public key pk is (N, e) and the private key sk is (N, d). The signing and verifying algorithms use a hash function $H : \{0,1\}^* \to \mathbb{Z}^*_N$ which maps bit strings of arbitrary length to the set of invertible integers modulo N.

$\texttt{SignFDH}_{N,d}(M)$	$\texttt{VerifyFDH}_{N,e}(M, x)$
$y \leftarrow H(M)$	$y \leftarrow x^e \mod N$
return $y^d \mod N$	if $y = H(M)$ then return 1 else return 0.

FDH is provably secure in the random oracle model, assuming that inverting RSA is hard. An *inverting algorithm* $\mathcal{I}$ for RSA gets as input (N, e, y) and tries to find $y^d \mod N$. Its success probability is the probability to output $y^d \mod N$ when (N, e, d) are obtained by running $\mathcal{RSA}(1^k)$ and y is set to $x^e \mod N$ for some x chosen at random in $\mathbb{Z}^*_N$.

Definition 5. *An inverting algorithm $\mathcal{I}$ is said to (t, ε)-break RSA if after at most $t(k)$ processing time its success probability is at least $\varepsilon(k)$ for all $k \in \mathbb{N}$.*

Definition 6. *RSA is said to be (t, ε)-secure if there is no inverter that (t, ε)-breaks RSA.*

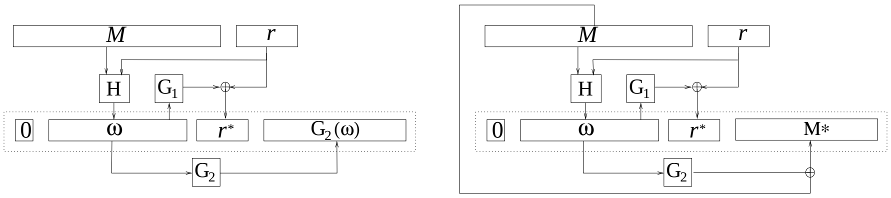

Fig. 1. PSS (left) and PSS-R (right)

The following theorem [5] proves the security of FDH in the random oracle model.

Theorem 1. *Assuming that RSA is* (t_I, ε_I)*-secure, FDH is* $(t_F,\ q_{hash},\ q_{sig},\ \varepsilon_F)$*-secure, with:*

$$t_I = t_F + (q_{hash} + q_{sig} + 1) \cdot \mathcal{O}(k^3) \tag{1}$$

$$\varepsilon_I = \frac{\varepsilon_F}{q_{sig}} \cdot \left(1 - \frac{1}{q_{sig}+1}\right)^{q_{sig}+1} \tag{2}$$

The technique described in [5] can be used to obtain an improved security proof for Gennaro-Halevi-Rabin's signature scheme [12] in the random oracle model and for Paillier's signature scheme [17]. From a forger which outputs a forgery with probability ε_F, the reduction succeeds in solving the hard problem with probability roughly ε_F/q_{sig}, in approximately the same time bound.

The security reduction of FDH is not tight: the probability ε_F of breaking FDH is smaller than roughly $q_{sig} \cdot \varepsilon_I$ where ε_I is the probability of inverting RSA, whereas the security reduction of PSS is tight: the probability of breaking PSS is almost the same as the probability of inverting RSA ($\varepsilon_F \simeq \varepsilon_I$).

3 New Security Proof for PSS

Several standards include PSS [2], among these are IEEE P1363a [14], a revision of ISO/IEC 9796-2, and the upcoming PKCS#1 v2.1 [18]. The signature scheme PSS is parameterized by the integers k, k_0 and k_1. The key generation is identical to FDH. The signing and verifying algorithms use two hash functions $H : \{0,1\}^* \rightarrow \{0,1\}^{k_1}$ and $G : \{0,1\}^{k_1} \rightarrow \{0,1\}^{k-k_1-1}$. Let G_1 be the function which on input $\omega \in \{0,1\}^{k_1}$ returns the first k_0 bits of $G(\omega)$, whereas G_2 is the function returning the remaining $k - k_0 - k_1 - 1$ bits of $G(\omega)$. A random *salt* r of k_0 bits is concatenated to the message M before hashing it. The scheme is illustrated in figure 1. In this section we obtain a better security proof for PSS, in which a shorter random salt is used to generate the signature.

$$\begin{array}{ll}
\texttt{SignPSS}(M): & \texttt{VerifyPSS}(M,x): \\
\quad r \stackrel{R}{\leftarrow} \{0,1\}^{k_0} & \quad y \leftarrow x^e \mod N \\
\quad \omega \leftarrow H(M\|r) & \quad \text{Break up } y \text{ as } b\|\omega\|r^*\|\gamma \\
\quad r^* \leftarrow G_1(\omega) \oplus r & \quad \text{Let } r \leftarrow r^* \oplus G_1(\omega) \\
\quad y \leftarrow 0\|\omega\|r^*\|G_2(\omega) & \quad \text{if } H(M\|r) = \omega \text{ and } G_2(\omega) = \gamma \text{ and } b = 1 \\
\quad \text{return } y^d \mod N & \quad \text{then return 1 else return 0}
\end{array}$$

The following theorem [2] proves the security of PSS in the random oracle model:

Theorem 2. *Assuming that RSA is (t', ε')-secure, the scheme* $\texttt{PSS}[k_0, k_1]$ *is $(t$, q_{sig}, $q_{hash}, \varepsilon)$-secure, where :*

$$t = t' - (q_{hash} + q_{sig} + 1) \cdot k_0 \cdot \mathcal{O}(k^3) \tag{3}$$

$$\varepsilon = \varepsilon' + 3 \cdot (q_{sig} + q_{hash})^2 \cdot \left(2^{-k_0} + 2^{-k_1}\right) \tag{4}$$

Theorem 2 shows that for PSS to be as secure as RSA (*i.e.* $\varepsilon' \simeq \varepsilon$), it must be the case that $(q_{sig} + q_{hash})^2 \cdot \left(2^{-k_0} + 2^{-k_1}\right) < \varepsilon'$, which gives $k_0 \geq k_{min}$ and $k_1 \geq k_{min}$, where:

$$k_{min} = 2 \cdot \log_2(q_{hash} + q_{sig}) + \log_2 \frac{1}{\varepsilon'} \tag{5}$$

Taking $q_{hash} = 2^{60}$, $q_{sig} = 2^{30}$ and $\varepsilon' = 2^{-60}$ as in [2], we obtain that k_0 and k_1 must be greater than $k_{min} = 180$ bits.

The following theorem shows that PSS can be proven as secure as RSA for a much shorter random salt, namely $k_0 = \log_2 q_{sig}$ bits, which for $q_{sig} = 2^{30}$ gives $k_0 = 30$ bits. The minimum value for k_1 remains unchanged.

Theorem 3. *Assuming that RSA is (t', ε')-secure, the scheme* $\texttt{PSS}[k_0, k_1]$ *is $(t$, q_{sig}, q_{hash}, $\varepsilon)$-secure, where :*

$$t = t' - (q_{hash} + q_{sig}) \cdot k_1 \cdot \mathcal{O}(k^3) \tag{6}$$

$$\varepsilon = \varepsilon' \cdot \left(1 + 6 \cdot q_{sig} \cdot 2^{-k_0}\right) + 2 \cdot (q_{hash} + q_{sig})^2 \cdot 2^{-k_1} \tag{7}$$

In appendix A, we give a security proof for a variant of PSS, for which the proof is simpler. The proof of theorem 3 is very similar and can be found in the full version of the paper [6]. The difference with the security proof of [2] is the following: in [2], a new random salt r is randomly generated for each signature query, and if r has appeared before, the inverter stops and has failed. Since at most $q_{hash} + q_{sig}$ random salts can appear during the reduction, the inverter stops after a given signature query with probability less than $(q_{hash} + q_{sig}) \cdot 2^{-k_0}$. There are at most q_{sig} signature queries, so this gives an error probability of:

$$q_{sig} \cdot (q_{hash} + q_{sig}) \cdot 2^{k_0}$$

which accounts for the term $(q_{hash} + q_{sig})^2 \cdot 2^{-k_0}$ in equation (4). On the contrary, in our new security proof, we generate for each new message M_i a list of q_{sig} random salts. Those random salts are then used to answer the signature queries for M_i, so there is no error probability when answering the signature queries.

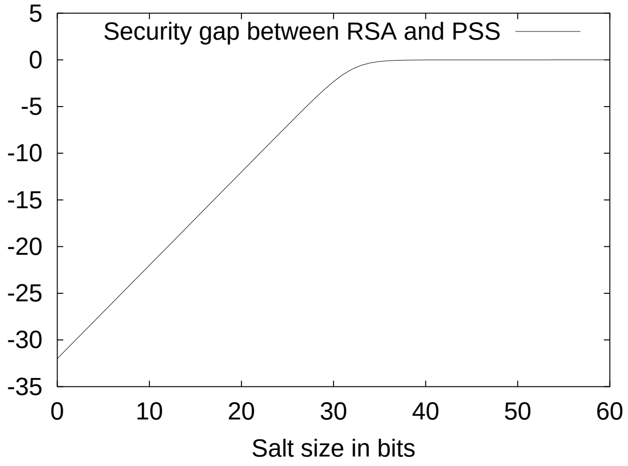

Fig. 2. Security gap between PSS and RSA: $\log_2 \varepsilon'/\varepsilon$ as a function of the salt size k_0 for $q_{sig} = 2^{30}$ signature queries.

3.1 Discussion

Theorem 3 shows that PSS is actually provably secure for any size k_0 of the random salt. In figure 2 we plot $\log_2 \varepsilon'/\varepsilon$ as a function of the size k_0 of the salt, which depicts the relative security of PSS compared to RSA, for $q_{sig} = 2^{30}$ and $k_1 > k_{min}$. For $k_0 = 0$, we reach the security level of FDH, where approximately $\log_2 q_{sig}$ bits of security are lost compared to RSA. For k_0 comprised between zero and $\log_2 q_{sig}$, we gain one bit of security when k_0 increases by one bit. And for k_0 greater than $\log_2 q_{sig}$, the security level of PSS is almost the same as inverting RSA. This shows that PSS has a tight security proof as soon as the salt size reaches $\log_2 q_{sig}$, and using larger salts does not further improve security. For the signer, q_{sig} represents the maximal number of signatures which can be generated for a given public-key. For example, for an application in which at most one billion signatures will be generated, $k_0 = 30$ bits of random salt are actually sufficient to guarantee the same level of security as RSA, and taking a larger salt does not increase the security level.

PSS-R is a variant of PSS which provides message recovery; the scheme is illustrated in figure 1. The goal is to save on the bandwidth: instead of transmitting the message separately, the message is recovered when verifying the signature. The security proof for PSS-R is almost identical to the security proof of PSS, and PSS-R achieves the same security level as PSS. Consequently, using the same parameters as for PSS with a 1024-bits RSA modulus, 813 bits of message can now be recovered when verifying the signature (instead of 663 bits with the previous security proof).

4 Optimal Security Proof for FDH

In section 2 we have seen that the security proof of theorem 1 for FDH is still not tight: the probability ε_F of breaking FDH is smaller than roughly $q_{sig} \cdot \varepsilon_I$ where ε_I is the probability of inverting RSA. In this section we show that the security proof of theorem 1 for FDH is optimal, *i.e.* there is no better reduction from inverting RSA to breaking FDH, and one cannot avoid loosing the q_{sig} factor in the probability bound. We use a similar approach as Boneh and Venkatesan in [3] for disproving the equivalence between inverting low-exponent RSA and factoring. They show that any efficient algebraic reduction from factoring to inverting low-exponent RSA can be converted into an efficient factoring algorithm. Such reduction is an algorithm $\mathcal{A}$ which factors N using an e-th root oracle for N. They show how to convert $\mathcal{A}$ into an algorithm $\mathcal{B}$ that factors integers without using the e-th root oracle. Thus, unless factoring is easy, inverting low-exponent RSA cannot be equivalent to factoring under algebraic reductions.

Similarly, we show that any better reduction from inverting RSA to breaking FDH can be converted into an efficient RSA inverting algorithm. Such reduction is an algorithm $\mathcal{R}$ which uses a forger as an oracle in order to invert RSA. We show how to convert $\mathcal{R}$ into an algorithm $\mathcal{I}$ which inverts RSA without using the oracle forger. Consequently, if inverting RSA is hard, there is no such better reduction for FDH, and the reduction of theorem 1 must be optimal.

Our technique is the following. Recall that resistance against adaptive chosen message attacks is considered, so the forger is allowed to make signature queries for messages of its choice, which must be answered by the reduction $\mathcal{R}$. Eventually the forger outputs a forgery, and the reduction must invert RSA. Therefore we first ask the reduction to sign a message M and receive its signature s, then we rewind the reduction to the state in which it was before the signature query, and we send s as a forgery for M. This is a true forgery for the reduction, because after the rewind there was no signature query for M, so eventually the reduction inverts RSA. Consequently, we have constructed from $\mathcal{R}$ an algorithm $\mathcal{I}$ which inverts RSA without using any forger. Actually, this technique allows to simulate a forger with respect to $\mathcal{R}$, without being able to break FDH. However, the simulation is not perfect, because it outputs a forgery only for messages which can be signed by the reduction, whereas a real forger outputs the forgery of a message that the reduction may or may not be able to sign.

We quantify the efficiency of a reduction by giving the probability that the reduction inverts RSA using a forger that $(t_F, q_{hash}, q_{sig}, \varepsilon_F)$-breaks the signature scheme, within an additional running time of t_R:

Definition 7. *We say that a reduction algorithm* $\mathcal{R}$ $(t_R, q_{hash}, q_{sig}, \varepsilon_F, \varepsilon_R)$ *-reduces inverting RSA to breaking FDH if upon input* (N, e, y) *and after running any forger that* $(t_F,\ q_{hash},\ q_{sig}, \varepsilon_F)$*-breaks FDH, the reduction outputs* $y^d \bmod N$ *with probability greater than* ε_R*, within an additional running time of* t_R*.*

In the above definition, t_R is the running time of the reduction algorithm only and does not include the running time of the forger. Eventually, the time needed to invert RSA is $t_F + t_R$, where t_F is the running time of the forger. For example, the reduction of theorem 1 for FDH $(t_R, q_{hash}, q_{sig}, \varepsilon_F, \varepsilon_R)$-reduces inverting RSA to breaking FDH with $t_R(k) = (q_{hash} + q_{sig}) \cdot \mathcal{O}(k^3)$ and $\varepsilon_R = \varepsilon_F/(4 \cdot q_{sig})$.

The following theorem, whose proof is given in appendix B, shows that from any such reduction $\mathcal{R}$ we can invert RSA with probability greater than roughly $\varepsilon_R - \varepsilon_F/q_{sig}$, in roughly the same time bound.

Theorem 4. *Let $\mathcal{R}$ be a reduction that $(t_R, q_{hash}, q_{sig}, \varepsilon_R, \varepsilon_F)$-reduces inverting RSA to breaking FDH. $\mathcal{R}$ runs the forger only once. From $\mathcal{R}$ we can construct an algorithm that (t_I, ε_I)-inverts RSA, with:*

$$t_I = 2 \cdot t_R \tag{8}$$

$$\varepsilon_I = \varepsilon_R - \varepsilon_F \cdot \frac{\exp(-1)}{q_{sig}} \cdot \left(1 - \frac{q_{sig}}{q_{hash}}\right)^{-1} \tag{9}$$

Theorem 4 shows that from any reduction $\mathcal{R}$ that inverts RSA with probability ε_R when interacting with a forger that outputs a forgery with probability ε_F, we can invert RSA with probability roughly $\varepsilon_R - \varepsilon_F/q_{sig}$, in roughly the same time bound, without using a forger. For simplicity, we omit here the factors $\exp(-1)$ and $(1 - q_{sig}/q_{hash})$ in equation (9). Moreover we consider a forger that makes q_{sig} signature queries, and with probability $\varepsilon_F = 1$ outputs a forgery[1].

Theorem 4 implies that from a polynomial time reduction $\mathcal{R}$ that succeeds with probability ε_R when interacting with this forger, we obtain a polynomial time RSA inverter $\mathcal{I}$ that succeeds with probability $\varepsilon_I = \varepsilon_R - 1/q_{sig}$, without using the forger. If inverting RSA is hard, the success probability ε_I of the polynomial time inverter must be negligible. Consequently, the success probability ε_R of the reduction must be less than $1/q_{sig} + \texttt{negl}$. This shows that from a forger that outputs a forgery with probability one, a polynomial time reduction cannot succeed with probability greater than $1/q_{sig} + \texttt{negl}$. On the contrary, a tight security reduction would invert RSA with probability close to one. Here we cannot avoid the q_{sig} factor in the security proof: the security level of FDH cannot be proven equivalent to RSA, and the security proof of theorem 1 for FDH is optimal.

5 Extension to any Signature Scheme with Unique Signature

We have introduced a new technique that enables to simulate a forger with respect to a reduction. It consists in making a signature query for a message M, rewinding the reduction, then sending the signature of M as a forgery. Actually,

[1] Such forger can be constructed by first factoring the modulus N, then computing a forgery using the factorisation of N.

this technique stretches beyond FDH and can be generalized and applied to any signature scheme in which each message has a unique signature. Moreover, the technique can be generalized to reductions running a forger more than once. The following theorem shows that for a hash-and-sign signature scheme with unique signature, a reduction allowed to run or rewind a forger at most r times cannot succeed with probability greater than roughly $r \cdot \varepsilon_F / q_{sig}$. The definitions and the proof of the theorem are given in the full version of the paper [6].

Theorem 5. *Let $\mathcal{R}$ be a reduction that $(t_R, q_{hash}, q_{sig}, \varepsilon_F, \varepsilon_R)$-reduces solving a problem Π to breaking a hash-and-sign signature scheme with unique signature. $\mathcal{R}$ is allowed to run or rewind a forger at most r times. From $\mathcal{R}$ we can construct an algorithm that (t_A, ε_A)-solves Π, with:*

$$t_A = (r+1) \cdot t_R \tag{10}$$

$$\varepsilon_A = \varepsilon_R - \varepsilon_F \cdot \frac{\exp(-1) \cdot r}{q_{sig}} \cdot \left(1 - \frac{q_{sig}}{q_{hash}}\right)^{-1} \tag{11}$$

6 Security Proofs for Signature Schemes in the Standard Model

The same technique can be applied to security reductions in the standard model, and we obtain the same upper bound in $1/q_{sig}$ for signature schemes with unique signature. The definitions and the proof of the following theorem are given in the full version of the paper [6].

Theorem 6. *Let $\mathcal{R}$ be a reduction that $(t_R, q_{sig}, \varepsilon_F, \varepsilon_R)$-reduces solving Π to breaking a signature scheme with unique signature. $\mathcal{R}$ can run or rewind the forger at most r times. Assume that the size of the message space is at least 2^ℓ. ¿From $\mathcal{R}$ we can construct an algorithm that (t_A, ε_A)-solves Π, with:*

$$t_A = (r+1) \cdot t_R \tag{12}$$

$$\varepsilon_A = \varepsilon_R - \varepsilon_F \cdot \frac{\exp(-1) \cdot r}{q_{sig}} \cdot \left(1 - \frac{q_{sig}}{2^\ell}\right)^{-1} \tag{13}$$

In [6] we give an example of a signature scheme with unique signature, provably secure in the standard model, and reaching the the above bound in $1/q_{sig}$.

7 Optimal Security Proof for PSS

In section 3 we have seen that $k_0 = \log_2 q_{sig}$ bits of random salt are sufficient for PSS to have a security level equivalent to RSA, and taking a larger salt does not further improve the security. In this section, we show that that this length is optimal: if a shorter random salt is used, the security level of PSS cannot be proven equivalent to RSA. Our technique described in section 4 does not apply directly because PSS is not a signature scheme with unique signature. We extend our technique to PSS using the following method.

We consider PSS in which the random salt is fixed to 0^{k_0}, and we denote this signature scheme $\texttt{PSS0}[k_0, k_1]$. Consequently, $\texttt{PSS0}[k_0, k_1]$ is a signature scheme with unique signature. First, we show how to convert a forger for $\texttt{PSS0}[k_0, k_1]$ into a forger for $\texttt{PSS}[k_0, k_1]$. A reduction $\mathcal{R}$ from inverting RSA to breaking $\texttt{PSS}[k_0, k_1]$ uses a forger for $\texttt{PSS}[k_0, k_1]$ in order to invert RSA. Consequently, from a forger for $\texttt{PSS0}[k_0, k_1]$, we can invert RSA using the reduction $\mathcal{R}$. This means that from $\mathcal{R}$ we can construct a reduction $\mathcal{R}_0$ from inverting RSA to breaking $\texttt{PSS0}[k_0, k_1]$. Since $\texttt{PSS0}[k_0, k_1]$ is a signature scheme with unique signature, theorem 5 gives an upper bound for the success probability of $\mathcal{R}_0$, from which we derive an upper bound for the success probability of $\mathcal{R}$.

Theorem 7. *Let $\mathcal{R}$ a reduction that $(t, q_{hash}, q_{sig}, \varepsilon_F, \varepsilon_R)$-reduces inverting RSA to breaking $\texttt{PSS}[k_0, k_1]$, with $q_{hash} \geq 2 \cdot q_{sig}$. The reduction can run or rewind the forger at most r times. ¿From $\mathcal{R}$ we can construct an inverting algorithm for RSA that (t_I, ε_I)-inverts RSA, with:*

$$t_I = (r+1) \cdot (t_R + q_{sig} \cdot \mathcal{O}(k)) \tag{14}$$

$$\varepsilon_I = \varepsilon_R - r \cdot \varepsilon_F \cdot \frac{2^{k_0+2}}{q_{sig}} \tag{15}$$

Proof. The proof is given in the full version of the paper [6].

Let consider as in section 4 a forger for $\texttt{PSS}[k_0, k_1]$ that makes q_{sig} signature queries and outputs a forgery with probability $\varepsilon_F = 1/2$. Then, from a polynomial time reduction $\mathcal{R}$ that succeeds with probability ε_R when running once this forger, we obtain a polynomial time inverter that succeeds with probability $\varepsilon_I = \varepsilon_R - 2^{k_0+1}/q_{sig}$, without using the forger. If inverting RSA is hard, the success probability ε_I of the polynomial time inverter must be negligible, and therefore the success probability ε_R of the reduction must be less than $2^{k_0+1}/q_{sig} + \texttt{negl}$. Consequently, in order to have a tight security reduction ($\varepsilon_R \simeq \varepsilon_R$), we must have $k_0 \simeq \log_2 q_{sig}$. The reduction of theorem 3 is consequently optimal.

8 Conclusion

We have described a new technique for analyzing the security proofs of signature schemes. The technique is both general and very simple and allows to derive upper bounds for security reductions using a forger as a black box, both in the random oracle model and in the standard model, for signature schemes with unique signature. We have also obtained a new criterion for a security reduction to be optimal, which may be of independent interest: we say that a security reduction is optimal if from a better reduction one can solve a difficult problem, such as inverting RSA. Our technique enables to show that the Full Domain Hash scheme, Gennaro-Halevi-Rabin's scheme and Paillier's signature scheme have an optimal security reduction in that sense. In other words, we have a matching lower and upper bound for the security reduction of those signature schemes: one cannot do better than losing a factor of q_{sig} in the security reduction.

Moreover, we have described a better security proof for PSS, in which a much shorter random salt is sufficient to achieve the same security level. This is of practical interest, since when PSS is used with message recovery, a better bandwidth is obtained because larger messages can be embedded inside the signature. Eventually, we have shown that this security proof for PSS is optimal: if a smaller random salt is used, PSS remains provably secure, but it cannot have the same level of security as RSA.

Acknowledgements :
I would like to thank Burt Kaliski, Jacques Stern and David Pointcheval for helpful discussions and the anonymous referees for their comments.

References

1. M. Bellare and P. Rogaway, *Random oracles are practical: a paradigm for designing efficient protocols*. Proceedings of the First Annual Conference on Computer and Commmunications Security, ACM, 1993.
2. M. Bellare and P. Rogaway, *The exact security of digital signatures - How to sign with RSA and Rabin*. Proceedings of Eurocrypt'96, LNCS vol. 1070, Springer-Verlag, 1996, pp. 399-416.
3. D. Boneh and R. Venkatesan, *Breaking RSA may not be equivalent to factoring*. Proceedings of Eurocrypt' 98, LNCS vol. 1403, Springer-Verlag, 1998, pp. 59–71.
4. R. Canetti, O. Goldreich and S. Halevi, *The random oracle methodology, revisited*, STOC' 98, ACM, 1998.
5. J.S. Coron, *On the exact security of Full Domain Hash*, Proceedings of Crypto 2000, LNCS vol. 1880, Springer-Verlag, 2000, pp. 229-235.
6. J.S. Coron, *Security proofs for PSS and other signature schemes*, Cryptology ePrint Archive, Report 2001/062, 2001. `http://eprint.iacr.org`
7. R. Cramer and I. Damgård, *New generation of secure and practical RSA-based signatures*, Proceedings of Crypto'96, LNCS vol. 1109, Springer-Verlag, 1996, pp. 173-185.
8. R. Cramer and V. Shoup, *Signature schemes based on the Strong RSA Assumption*, May 9, 2000, revision of the extended abstract in Proc. 6th ACM Conf. on Computer and Communications Security, 1999; To appear, ACM Transactions on Information and System Security (ACM TISSEC). Available at `http://www.shoup.net/`
9. W. Diffie and M. Hellman, *New directions in cryptography*, IEEE Transactions on Information Theory, IT-22, 6, pp. 644-654, 1976.
10. C. Dwork and M. Naor, *An efficient existentially unforgeable signature scheme and its applications*, In J. of Cryptology, 11 (3), Summer 1998, pp. 187-208.
11. FIPS 186, *Digital signature standard*, Federal Information Processing Standards Publication 186, U.S. Department of Commerce/NIST, 1994.

12. R. Gennaro, S. Halevi and T. Rabin, *Secure hash-and-sign signatures without the random oracle*, proceedings of Eurocrypt '99, LNCS vol. 1592, Springer-Verlag, 1999, pp. 123-139.
13. S. Goldwasser, S. Micali and R. Rivest, *A digital signature scheme secure against adaptive chosen-message attacks*, SIAM Journal of computing, 17(2), pp. 281-308, April 1988.
14. IEEE P1363a, *Standard Specifications For Public Key Cryptography: Additional Techniques*, available at `http://www.manta.ieee.org/groups/1363`
15. A. Lenstra and H. Lenstra (eds.), *The development of the number field sieve*, Lecture Notes in Mathematics, vol 1554, Springer-Verlag, 1993.
16. K. Ohta and T. Okamoto, *On concrete security treatment of signatures derived from identification*. Prooceedings of Crypto '98, Lecture Notes in Computer Science vol. 1462, Springer-Verlag, 1998, pp. 354-369.
17. P. Paillier, *Public-key cryptosystems based on composite degree residuosity classes*. Proceedings of Eurocrypt'99, Lecture Notes is Computer Science vol. 1592, Springer-Verlag, 1999, pp. 223-238.
18. PKCS #1 v2.1, *RSA Cryptography Standard (draft)*, available at `http://www.rsa security.com /rsalabs/pkcs`.
19. D. Pointcheval and J. Stern, *Security proofs for signature schemes*. Proceedings of Eurocrypt'96, LNCS vol. 1070, Springer-Verlag, pp. 387-398.
20. R. Rivest, A. Shamir and L. Adleman, *A method for obtaining digital signatures and public key cryptosystems*, CACM 21, 1978.

A Security Proof of a variant of PSS

We describe a variant of PSS that we call PFDH, for Probabilistic Full Domain Hash, for which the security proof is simpler. The scheme is similar to Full Domain Hash except that a random salt of k_0 bits is concatenated to the message M before hashing it. The difference with PSS is that the random salt is not recovered when verifying the signature; instead the random salt is transmitted separately. As FDH, the scheme uses a hash function $H : \{0,1\}^* \to \mathbb{Z}_N^*$.

$\texttt{SignPFDH}(M)$:	$\texttt{VerifyPFDH}(M, s, r)$:
$r \stackrel{R}{\leftarrow} \{0,1\}^{k_0}$	$y \leftarrow s^e \mod N$
$y \leftarrow H(M \| r)$	if $y = H(M \| r)$ then return 1
return $(y^d \mod N, r)$	else return 0

The following theorem proves the security of PFDH in the random oracle model, assuming that inverting RSA is hard. It shows that PFDH has a tight security proof for a random salt of length $k_0 = \log_2 q_{sig}$ bits.

Theorem 8. *Suppose that RSA is (t', ε')-secure. Then the signature scheme* $\texttt{PFDH}[k_0]$ *is $(t, q_{hash}, q_{sig}, \varepsilon)$-secure, where:*

$$t = t' - (q_{hash} + q_{sig}) \cdot \mathcal{O}(k^3) - q_{hash} \cdot q_{sig} \cdot \mathcal{O}(k) \tag{16}$$

$$\varepsilon = \varepsilon' \cdot \left(1 + 6 \cdot q_{sig} \cdot 2^{-k_0}\right) \tag{17}$$

Proof. Let $\mathcal{F}$ be a forger that $(t, q_{sig}, q_{hash}, \varepsilon)$-breaks PFDH. We construct an inverter I that (t', ε')-breaks RSA. The inverter receives as input (N, e, η) and must output $\eta^d \mod N$. We assume that the forger never repeats a hash query. However, the forger may repeat a signature query, in order to obtain the signature of M with distinct integers r. The inverter $\mathcal{I}$ maintains a counter i, initially set to zero.

When a message M appears for the first time in a hash query or a signature query, the inverter increments the counter i and sets $M_i \leftarrow M$. Then, the inverter generates a list L_i of q_{sig} random integers in $\{0,1\}^{k_0}$.

When the forger makes a hash query for $M_i \| r$, we distinguish two cases. If r belongs to the list L_i, the inverter generates a random $x \in \mathbb{Z}_N^*$ and returns $H(M_i \| r) = x^e \mod N$. Otherwise, the inverter generates a random $x \in \mathbb{Z}_N^*$ and returns $\eta \cdot x^e \mod N$. Consequently, for each message M_i, the list L_i contains the integers $r \in \{0,1\}^{k_0}$ such that the inverter knows the signature x corresponding to $M_i \| r$.

When the forger makes a signature query for M_i, the inverter takes the next random r in the list L_i. Since the list contains initially q_{sig} integers and there are at most q_{sig} signature queries, this is always possible. If there was already a hash query for $M_i \| r$, we have $H(M_i \| r) = x^e \mod N$ and the inverter returns the signature x. Otherwise the inverter generates a random $x \in \mathbb{Z}_N^*$, sets $H(M_i \| r) = x^e \mod N$ and returns the signature x.

When the forger outputs a forgery (M, s, r), we assume that it has already made a hash query for M, so $M = M_i$ for a given i. Otherwise, the inverter goes ahead and makes the hash query for $M \| r$. Then if r does not belong to the list L_i, we have $H(M_i \| r) = \eta \cdot x^e \mod N$. From $s = H(M_i \| r)^d = \eta^d \cdot x \mod N$, we obtain $\eta^d = s/x \mod N$ and the inverter succeeds in outputting $\eta^d \mod N$.

Since the forger has not made any signature query for the message M_i in the forgery (M_i, s, r), the forger has no information about the q_{sig} random integers in the list L_i. Therefore, the probability that r does not belong to L_i is $(1-2^{-k_0})^{q_{sig}}$. If the size k_0 of the random salt is greater than $\log_2 q_{sig}$, we obtain if $q_{sig} \geq 2$:

$$\left(1 - 2^{-k_0}\right)^{q_{sig}} \geq \left(1 - \frac{1}{q_{sig}}\right)^{q_{sig}} \geq \frac{1}{4}$$

Since the forger outputs a forgery with probability ε, the success probability ε' of the inverter is then at least $\varepsilon/4$, which shows that for $k_0 \geq \log_2 q_{sig}$ the probability of breaking PFDH is almost the same as the probability of inverting RSA.

For the general case, *i.e.* if we do not assume $k_0 \geq \log_2 q_{sig}$, we generate fewer than q_{sig} random integers in the list L_i, so that the salt r in the forgery (M_i, s, r) belongs to L_i with lower probability. More precisely, starting from an empty list L_i, the inverter generates with probability β a random $r \leftarrow \{0,1\}^{k_0}$, adds it to L_i, and starts again until the list L_i contains q_{sig} elements. Otherwise (so with probability $1 - \beta$) the inverter stops adding integers to the list. The number a_i of integers in L_i is then a random variable following a geometric law of parameter β:

$$\Pr[a_i = j] = \begin{cases} (1-\beta) \cdot \beta^j & \text{if } j < q_{sig} \\ \beta^{q_{sig}} & \text{if } j = q_{sig} \end{cases} \tag{18}$$

The inverter answers a signature query for M_i if the corresponding list L_i contains one more integer, which happens with probability β (otherwise the inverter must abort). Consequently, the inverter answers all the signature queries with probability greater than $\beta^{q_{sig}}$. Note that if $\beta = 1$, the setting boils down to the previous case: all the lists L_i contain exactly q_{sig} integers, and the inverter answers all the signature queries with probability one.

The probability that r in the forgery (M_i, s, r) does not belong to the list L_i is then $(1 - 2^{-k_0})^j$, when the length a_i of L_i is equal to j. The probability that r does not belong to L_i is then:

$$f(\beta) = \sum_{j=0}^{q_{sig}} \Pr[a_i = j] \cdot \left(1 - 2^{-k_0}\right)^j \tag{19}$$

Since the forger outputs a forgery with probability ε, the success probability of the inverter is at least $\varepsilon \cdot \beta^{q_{sig}} \cdot f(\beta)$. We select a value of β which maximizes this success probability; in [6], we show that for any (q_{sig}, k_0), there exists β_0 such that:

$$\beta_0^{q_{sig}} \cdot f(\beta_0) \geq \frac{1}{1 + 6 \cdot q_{sig} \cdot 2^{-k_0}} \tag{20}$$

which gives (17). The running time of $\mathcal{I}$ is the running time of $\mathcal{F}$ plus the time necessary to compute the integers $x^e \mod N$ and to generate the lists L_i, which gives (16).

B Proof of Theorem 4

¿From $\mathcal{R}$ we build an algorithm $\mathcal{I}$ that inverts RSA, without using a forger for FDH. We receive as input (N, e, y) and our goal is to output $y^d \mod N$ using $\mathcal{R}$. We select q_{hash} distinct messages $M_1, \ldots, M_{q_{hash}}$ and start running $\mathcal{R}$ with (N, e, y).

First we ask $\mathcal{R}$ to hash the q_{hash} messages $M_1, \ldots, M_{q_{hash}}$, and obtain the hash values $h_1, \ldots, h_{q_{hash}}$. We select a random integer $\beta \in [1, q_{hash}]$ and a random sequence α of q_{sig} integers in $[1, q_{hash}] \setminus \{\beta\}$, which we denote $\alpha = (\alpha_1, \ldots, \alpha_{q_{sig}})$. We select a random integer $i \in [1, q_{sig}]$ and define the sequence of i integers $\alpha' = (\alpha_1, \ldots, \alpha_{i-1}, \beta)$. Then we make the i signature queries corresponding to α' to $\mathcal{R}$ and receive from $\mathcal{R}$ the corresponding signatures, the last one being the signature s_β of M_β. For example, if $\alpha' = (3, 2)$, this corresponds to making a signature query for M_3 first, and then for M_2.

Then we rewind $\mathcal{R}$ to the state it was after the hash queries, and this time, we make the q_{sig} signature queries corresponding to α. If $\mathcal{R}$ has answered all the signature queries, then with probability ε_F, we send (M_β, s_β) as a forgery to $\mathcal{R}$. This is a true forgery for $\mathcal{R}$ because after the rewind of $\mathcal{R}$, there was no signature query for M_β. Eventually $\mathcal{R}$ inverts RSA and outputs $y^d \mod N$.

We denote by $\mathcal{Q}$ the set of sequences of signature queries which are correctly answered by $\mathcal{R}$ after the hash queries, in time less than t_R. If a sequence of signature queries is correctly answered by $\mathcal{R}$, then the same sequence without the last signature query is also correctly answered, so for any $(\alpha_1, \ldots, \alpha_j) \in \mathcal{Q}$, we have $(\alpha_1, \ldots, \alpha_{j-1}) \in \mathcal{Q}$. Let us denote by `ans` the event $\alpha \in \mathcal{Q}$, which corresponds to $\mathcal{R}$ answering all the signature queries after the rewind, and by `ans'` the event $\alpha' \in \mathcal{Q}$, which corresponds to $\mathcal{R}$ answering all the signature queries before the rewind.

Let us consider a forger that makes the same hash queries, the same signature queries corresponding to α, and outputs a forgery for M_β with probability ε_F. By definition, when interacting with such a forger, $\mathcal{R}$ would output $y^d \mod N$ with probability at least ε_R. After the rewind, $\mathcal{R}$ sees exactly the same transcript as when interacting with this forger, except if event `ans` is true and `ans'` is false: in this case, the forger outputs a forgery with probability ε_F, whereas our simulation does not output a forgery. Consequently, when interacting with our simulation of a forger, $\mathcal{R}$ outputs $y^d \mod N$ with probability at least:

$$\varepsilon_R - \varepsilon_F \cdot \Pr[\texttt{ans} \wedge \neg \texttt{ans'}] \tag{21}$$

The proof of the following lemma is given in the full version of the paper [6].

Lemma 1. *Let $\mathcal{Q}$ be a set of sequences of at most n integers in $[1, k]$, such that for any sequence $(\alpha_1, \ldots, \alpha_j) \in \mathcal{Q}$, we have $(\alpha_1, \ldots, \alpha_{j-1}) \in \mathcal{Q}$. Then the following holds:*

$$\Pr_{\substack{i \leftarrow [1,n] \\ (\alpha_1, \ldots, \alpha_n, \beta) \leftarrow [1,k]^{n+1}}} [(\alpha_1, \ldots, \alpha_n) \in \mathcal{Q} \wedge (\alpha_1, \ldots, \alpha_{i-1}, \beta) \notin \mathcal{Q}] \leq \frac{\exp(-1)}{n}$$

Using lemma 1 with $n = q_{sig}$ and $k = q_{hash}$, we obtain:

$$\Pr[\texttt{ans} \wedge \neg \texttt{ans'}] \leq \frac{\exp(-1)}{q_{sig}} \left(1 - \frac{q_{sig}}{q_{hash}}\right)^{-1} \tag{22}$$

The term $(1 - q_{sig}/q_{hash})$ in equation (22) is due to the fact that we select $\alpha_1, \ldots, \alpha_{q_{sig}}$ in $[1, q_{hash}] \setminus \{\beta\}$ whereas in lemma 1 the integers are selected in $[1, q_{hash}]$. ¿From equations (21) and (22) we obtain that $\mathcal{I}$ succeeds with probability greater than ε_I given by (9). Because of the rewind, the running time of $\mathcal{I}$ is at most twice the running time of $\mathcal{R}$, which gives (8) and terminates the proof.