ECOLE NORMALE SUPERIEURE

On Provable Security
for Digital Signature Algorithms

David POINTCHEVAL
Serge VAUDENAY

LIENS - 96 - 17

Département de Mathématiques et Informatique

CNRS URA 1327

On Provable Security
for Digital Signature Algorithms

David POINTCHEVAL
Serge VAUDENAY

LIENS - 96 - 17

October 1996

Laboratoire d'Informatique de l'Ecole Normale Supérieure
45 rue d'Ulm 75230 PARIS Cedex 05

Tel : (33)(1) 44 32 00 00
Adresse électronique : ... @dmi.ens.fr

On Provable Security for Digital Signature Algorithms

David Pointcheval
David.Pointcheval@ens.fr

Serge Vaudenay
Serge.Vaudenay@ens.fr

Laboratoire d'Informatique
École Normale Supérieure
45, rue d'Ulm
F – 75230 PARIS Cedex 05

Abstract

In this paper we consider provable security for ElGamal-like digital signature schemes. We point out that the good security criterion on the underlying hash function is pseudorandomness. We extend Pointcheval-Stern's results about the use of the random oracle model to prove the security of two variants of the US Digital Signature Algorithm against adaptive attacks which issue an existential forgery. We prove that a very practical use of the random oracle model is possible whith tamper-resistant modules.

1 The security of cryptographic hash functions

Cryptographic hash functions are commonly used for providing message authentication. So far, several security criteria have been considered. The most popular criteria are collision freedom and one-wayness. Roughly, collision freedom is the property that no practical algorithm can issue a pair (x, x') such that $x \neq x'$ and $F(x) = F(x')$ (see Damgård [12, 13] and Merkle [25]). One-wayness is the property that no practical algorithm can find x out from $F(x)$ with a significant fraction of x (see Rabin [32] and Merkle [24, 25]). There exist more technical criteria. (See for instance Preneel [31].)

So far, a few results have been found on the interaction of those criteria and the security of (public-key) cryptographic schemes. For instance, Girault and Stern showed how hash functions provide security in most of popular identification schemes [19]. When used in digital signature schemes, we know no results about which criterion shall be considered. Here, we consider the interesting criterion of "pseudorandomness".

Pseudorandomness has already been considered in context with the security of pseudorandom generators in the early beginning of the history of public key cryptography. This has been formalized by Goldreich, Goldwasser and Micali [17], following previous work from Shamir [38], Blum and Micali [9] and Yao [42]. This notion stands on the Turing Test [21]: a function is not a pseudorandom generator if an intelligent opponent can distinguish (with relevant probability) the output of the generator from a truly random sequence.

Let us formally define this notion.

Definition 1. Let $\mathcal{H} = (h_K)_{K\in\mathcal{K}}$ be a family of hash functions from a finite set A to a finite set B, where the random key K follows a distribution $\mathcal{K}$. We say $\mathcal{H}$ is a (n,ϵ)-pseudorandom hash functions family if no probabilistic Turing machine which can send only n queries to an oracle can distinguish the random oracle h_K from a truly random oracle within an advantage greater than ϵ. More precisely, for any n-limited probabilistic Turing machine $\mathcal{A}$, we have

$$\left|\Pr_{\omega,K}\left[\mathcal{A}^{h_K}(\omega) \text{ accept}\right] - \Pr_{\omega,O}\left[\mathcal{A}^{O}(\omega) \text{ accept}\right]\right| \leq \epsilon$$

where ω is the random tape and O is a truly random A-to-B mapping.

In this paper we show that this criterion can be used for providing security of digital signature schemes.

So far, this criterion has not been consider for hash functions. It has already been considered for encryption functions. Actually, Luby and Rackoff proved that a truly random 3-round Feistel cipher for ℓ-bit messages is $(n, n^2/2^{\ell/2})$-pseudorandom, so that the privacy is safe until $n \approx 2^{\ell/4}$ messages have been encrypted [22]. This was to argue that the US Data Encryption Standard [1] was strong. Later on, Patarin generalized this result [27]. He proved that a truly random 6-round Feistel cipher is $(n, 12n^3/2^\ell)$-pseudorandom, so that it is safe until $n \approx 2^{\ell/3}$. He further conjectured it is still safe until $n \approx 2^{\ell/2}$, maybe with a little more rounds. We point out this is the strongest possible result since a r-round random Feistel cipher is defined by $r(\ell/2)2^{\ell/2}$ bits and Shannon's Theorem states we cannot expect any unconditional security whenever $n\ell$ is greater than this number [40].

This construction can be adapted to make pseudorandom hash functions.

Theorem 2. *Let B be the set of ℓ-bit strings, and $A = B^2$. From a $2\ell 2^\ell$-bit key $K = (F,G)$ we define two B-to-B functions denoted F and G. We define $h_K(x,y) = y \oplus G(x \oplus F(y))$. The family $(h_K)_K$ is $(n, (1/2)n^2/2^\ell)$-pseudorandom.*

This family is nothing but a truncated two-round Feistel construction.

Proof. This proof is freely adapted from previous work [22, 27] and also from Maurer [23]. The bulk of the proof consist in finding a meaningful lower bound for the probability that n different x_iy_i's produce n given z_i's. More precisely, the ratio between this probability and the same probability for a truly random function needs to be greater than $1-\epsilon$.

Let $T = x \oplus F(y)$. We have

$$\begin{aligned}\Pr[h_K(x_iy_i) = z_i; i = 1,\ldots,n] &\geq \Pr[h_K(x_iy_i) = z_i \text{ and } T_i \text{ differents}]\\ &\geq \left(\frac{1}{2^\ell}\right)^n \left(1 - \frac{n(n-1)}{2}\min_{i,j}\Pr[T_i = T_j]\right)\end{aligned}$$

and for any $i \neq j$, since we have $x_i y_i \neq x_j y_j$, either we have $y_i \neq y_j$ so we get $\Pr[T_i = T_j] = 1/2^\ell$, or we have $y_i = y_j$ and $x_i \neq x_j$ and then $\Pr[T_i = T_j] = 1$. So, we have

$$\Pr[h_K(x_i y_i) = z_i; i = 1, \ldots, n] \geq \left(\frac{1}{2^\ell}\right)^n \left(1 - \frac{n(n-1)}{2}\frac{1}{2^\ell}\right)$$

so $\epsilon = (1/2)n^2/2^\ell$.

From this lemma, let us consider a probabilistic distinguisher $\mathcal{A}^O$ which uses a random tape ω. We have

$$\begin{aligned}
\Pr_{\omega,K}[\mathcal{A}^{h_K}(\omega) \text{ accepts}] &= \sum_{\substack{\text{accepting} \\ x_1 y_1 z_1 \ldots x_n y_n z_n}} \Pr_{\omega,K}[x_1 y_1 z_1 \ldots x_n y_n z_n] \\
&= \sum_{x_i y_i z_i} \Pr_{\omega}[x_i y_i z_i / x_i y_i \xrightarrow{O} z_i] \Pr_{K}[h_K(x_i y_i) = z_i] \\
&\geq (1-\epsilon) \sum_{x_i y_i z_i} \Pr_{\omega}[x_i y_i z_i / x_i y_i \xrightarrow{O} z_i] \Pr_{O}[O(x_i y_i) = z_i] \\
&= (1-\epsilon) \Pr_{\omega,O}[\mathcal{A}^O(\omega) \text{ accepts}]
\end{aligned}$$

so we have

$$\Pr_{\omega,K}[\mathcal{A}^{h_K}(\omega) \text{ accepts}] - \Pr_{\omega,O}[\mathcal{A}^O(\omega) \text{ accepts}] \geq -\epsilon$$

therefore the advantage is smaller than ϵ by symmetrical argument (*i.e.* by using another distinguisher which accepts if and only if $\mathcal{A}$ rejects). □

We also remark we can improve this construction by replacing F by a random linear function: if $K = (a, G)$ where a is an ℓ-bit string, we define $h_K(x) = y \oplus G(x \oplus a \times y)$ where $a \times y$ is the product in $\mathrm{GF}(2^\ell)$. Actually, we only used that $F(y_i) \oplus F(y_j)$ were uniform in the proof (which corresponds to the notion of xor-universal hash function, see Carter and Wegman [11]). Similarly, by replacing G by a polynomial with degree $n-1$, the same result holds. We thus obtain a family of $(n, (1/2)n^2/2^\ell)$-pseudorandom hash functions with a $(n+1)\ell$-bit key.

On a more practical level we can consider usual hash functions like Rivest's MD4 or MD5 functions [33, 34] in a keyed mode. One can assume a given hash function family is pseudorandom as one assumes the discrete logarithm is not computable. For instance, we can define $h_K(x) = \mathrm{MD4}_K(x)$ where K is used as the initial value (see [33]). Although Dobbertin proved MD4 is not collision-resistant [14] and that MD5 is likely not to be so too [15], we can wonder whether this has any consequence on the assumed pseudorandomness.

2 Short-term provably secure scheme

We recall that a digital signature scheme is defined by a distribution Gen over the keys (which can be given by a pseudorandom generation algorithm), a signing algorithm Sig which depends on the secret key and which may be probabilistic, and a verification algorithm Ver which depends on the public key (see Goldwasser, Micali and Rivest [18]). Here we also assume that both Sig and Ver use an oracle O. This commonly formalizes the cryptographic hash function which issues a message digest from the document to be signed.

Definition 3. Let $\Sigma^O = (\mathrm{Gen}, \mathrm{Sig}^O, \mathrm{Ver}^O)$ be a signature scheme which depends on a random oracle O with a specified distribution. We say Σ is (n, t, ϵ)-secure against adaptive attacks for existential forgery if no probabilistic Turing machine which is allowed to have only n accesses to O and Sig can forge a pair (m, σ) within a time t where σ is a valid signature for the message m within a probability greater than ϵ. More precisely, for any (n, t)-limited probabilistic Turing machine $\mathcal{A}$ which outputs valid signatures or fails, we have

$$\Pr_{\omega,O}\left[\mathcal{A}^{O,\mathrm{Sig}}(\omega) \text{ succeeds}\right] \leq \epsilon$$

where ω is the random tape.

We note that the oracle O must be used as a black box in this model. This corresponds to a notion of black-box security which claims to have a practical application if we think about h_K as a tamper resistant device provided by a trusted authority. In earlier work [6, 7, 29, 37, 30], the random oracle model used to consider uniform distribution over all possible oracles. This means no practical issue since the sample space is quite huge: a real tamper resistant black box on all the 128-bit strings would have require a 2^{102}Gbyte random access memory! In this paper we show how to derandomize the oracle so that this notion of security gets some practical mean.

We however note that our notion of security does not include security against denying signatures: disability to forge signatures does not mean undeniability by the secret key holder.

Theorem 4. *If $\mathcal{H}$ is a (n, ϵ_1)-pseudorandom hash functions family and Σ^O is (n, t, ϵ_2)-secure against adaptive attacks for existential forgery where O is a random oracle uniformly distributed, then $\Sigma^{\mathcal{H}}$ is $(n, t, \epsilon_1 + \epsilon_2)$-secure as well.*

We recall that the hash function h_K must be used as a black box for this Theorem to be applicable. That is to say the signer must not have access to the key K. (Note that the hash function family is not pseudorandom if we can get K after a few requests to the black box.)

Proof. Let $\mathcal{A}^{\mathcal{H},\mathrm{Sig}}$ be a Turing machine which forges valid signature for h_K within a probability greater than $\epsilon_1 + \epsilon_2$. We distinguish h_K from a random oracle O by applying $\mathcal{A}$ and considering whether it is a success or not. Since $\mathcal{A}^{O,\mathrm{Sig}}$ cannot forge a signature within a probability greater than ϵ_2, the advantage is greater than ϵ_1 which contradicts the hypothesis. $\square$

An interesting consequence of this Theorem is that if the computers of the planet cannot compute n hashed value within a time t, the parameter K can be chosen at random by a trusted authority, with a temporal validity. If we only need short-term signatures, namely if the validity period of a signature is no greater than t, this approach provides a provable signature scheme. Otherwise, one needs to provide a timestamp on the signature: we need a trusted timestamping service which basically signs all long-term signatures before the end of the period. Of course, this should be signed with the next public random oracle as any new signature!

This situation is quite reasonable since the use of long-term signature already requires timestamp. A classical argument claims that a secret key can be accidentally compromised, so we need a way to prove when a signature was made to validate or invalidate it [20, 5].

To sum up the protocol, we assume there is a trusted authority and several timestamping notaries. At the beginning of a period, the authority provides a new random hash function h_K as a tamper-proof black box. Then, the notaries sign all previously deposited documents with this new black box. (If they are not trusted, the signers may request for those signatures to check that it has been done.) Then, all the users can sign with the new black box. For any long-term signature, it is necessary to deposit the corresponding document to a timestamping notary.

This only holds for avoiding forgery attacks. In order to avoid deniability, the signer should re-sign himself his document and request for a new timestamp, if the signature scheme provides security against this kind of attack during this period.

3 Two provably secure variants of the US DSA algorithms

In 1994 was proposed DSA as the US Digital Signature Standard [3]. We will show now that slight variants of DSA are provably secure under some definite hypothesis. First of all we recall the DSA algorithm.

There are public parameters: a 512-bit (or longer) prime p, a 160-bit prime factor q of $p-1$, and a qth primitive root of unity g modulo p. Each signer has a secret key x and a public key $y = g^x \bmod p$. The Gen distribution of x is the uniform distribution over integers from 1 to $q-1$. To sign a message m, the Sig algorithm picks a random k from 1 to $q-1$ and computes

$$r = g^k \bmod p \bmod q \tag{1}$$

$$s = \frac{\mathrm{SHA}(m) + xr}{k} \bmod q \tag{2}$$

where SHA denotes the algorithm described in the US Secure Hash Standard [2]. Formally, the random tape ω defines k and $\mathrm{Sig}(\omega, m) = (r, s)$. The Ver algorithm consists in checking whether s is invertible modulo q and

$$g^{\frac{\mathrm{SHA}(m)}{s}} y^{\frac{r}{s}} \bmod p \bmod q = r \tag{3}$$

or not.

3.1 First variant

At the Crypto'96 conference, Brickell claimed the following variant of the DSA is secure in the random oracle model [10]. First we replace SHA by a random oracle H_1. Second, we replace the mod q hash function in Equation (1) by another random oracle H_2. So, we have

$$r = H_2(g^k \bmod p) \tag{4}$$
$$s = \frac{H_1(m) + xr}{k} \bmod q \tag{5}$$

and

$$H_2\left(g^{\frac{H_1(m)}{s}} y^{\frac{r}{s}} \bmod p\right) = r \tag{6}$$

becomes the verification formula.

Theorem 5. *The digital signature scheme* $(\mathrm{Gen}, \mathrm{Sig}^{H_1,H_2}, \mathrm{Ver}^{H_1,H_2})$ *defined by the Equations (4,5,6) is* (n, t, ϵ)*-secure against adaptive attacks for existential forgery where* H_1 *and* H_2 *are uniformly distributed unless we can compute the discrete logarithm in the group spanned by* g *within a time* $11nt/\epsilon$ *with probability greater than* $1/11$*.*

Moreover we obtain in a straightforward way that if H_2 and H_1 mod q are collision resistant, than the signatures are undeniable.

Proof. We adapt the proof of security from Pointcheval and Stern on a variant of the ElGamal signature scheme [29, 28]. We assume there exists a (probabilistic) algorithm $\mathcal{A}$ which on input y can issue a valid signed message (m, r, s) by querying the random oracles H_1, H_2 and the signature algorithm Sig. We assume the $\mathcal{A}$'s valid signature has not been queried to the Sig oracle. (Otherwise, this is not an attack!) Let $u = g^{H_1(m)/s} y^{r/s} \bmod p$. Since the signature is valid, we must have

$$u = g^{\frac{H_1(m)}{s}} y^{\frac{H_2(u)}{s}} \bmod p \tag{7}$$

from Equation (6). We can assume that both m and u has been queried to oracles H_1 and H_2 respectively (otherwise, the signature is not valid for any other oracle which answer the same to the queries).

By arguments similar than in [29] if we run $\mathcal{A}$ on a random (H_1, H_2) $1/\epsilon$ times in order to get a valid signature, depending on which, from m and u was the last query, we fork H_1 and H_2 and rerun $\mathcal{A}$ an extra $11n/\epsilon$ times. By using an improved forking lemma (due to Pointcheval [28]) we obtain two forking signatures. By applying the Lemma 7, we obtain a fork with probability $1/11$ within a time $11nt/\epsilon$.

- If m is queried before u, we obtain $H_1(m) = H_1'(m) = h$ and $H_2(u) = r$ and $H_2'(u) = r'$ with $r \neq r'$. Hence we have

$$g^{\frac{h}{s}} y^{\frac{r}{s}} \equiv g^{\frac{h}{s'}} y^{\frac{r'}{s'}} \pmod p$$

from Equation (7) which leads to the discrete logarithm of y in basis g (we cannot have $r/s = r'/s'$ otherwise we obtain $h/s = h/s'$ then $s = s'$ then $r = r'$ which is false).

- If m is queried after u, we obtain $H_1(m) = h$, $H_1'(m) = h'$ with $h \neq h'$ and $H_2(u) = H_2'(u) = r$. From Equation (7) we obtain

$$g^{\frac{h}{s}} y^{\frac{r}{s}} \equiv g^{\frac{h'}{s'}} y^{\frac{r}{s'}} \pmod p$$

which leads to x too (we cannot have $r/s = r/s'$ for similar reasons).

Therefore, we proved $\mathcal{A}$ can be used to compute discrete logarithms modulo p by having access to the Sig oracle.

Now it is easy to argue we can simulate a Sig oracle with the same distribution. Actually, on query m, we can forge a valid (h_1, r, s)-signature since $H_1(m)$ and $r = H_2(u)$ are truly random. Hence there is no way for the discrete logarithm algorithm to distinguish the real Sig from the simulator, so it is able to compute the discrete logarithm on its own. □

Although Brickell did not publish his proof, we can reasonably say this one is fairly similar [10]. This approach however suffers from the drawback that we cannot argue the US DSA is secure by handwaving the mod q mapping behaves like a random oracle!

3.2 Second variant

The DSA essentially comes from the ElGamal signature scheme [16]. There are two main differences between them. Firstly, the change from $p-1$ itself to a prime factor q of $p-1$ which was originally due to Schnorr [35, 36]. This fixes some weaknesses later on discovered by Bleichenbacher [8], van Oorschot and Wiener [26] and also discussed by Anderson and Vaudenay [4]. Secondly, the use of the hash function mod q (also used by Schnorr in another way) in order to reduce the length of the signature. We have just seen that replacing mod q by a random oracle enables to prove the security.

In another variant, we consider a fixed function F such as mod q in order to reduce the size of the signature, but we introduce another idea due to Schnorr, which consists in hashing the message together with the r value. So, we have

$$r = F(g^k \bmod p) \tag{8}$$

$$s = \frac{H(m,r) + xr}{k} \bmod q \tag{9}$$

and

$$F\left(g^{\frac{H(m,r)}{s}} y^{\frac{r}{s}} \bmod p\right) = r \tag{10}$$

becomes the verification formula. We note that F needs to be a mapping from the q-ordered subgroup spanned by g in $\mathbb{Z}_p{}^*$ to a finite set. In context with $F(x) = x \bmod q$, we assume this set to have cardinality q too. We say F has a ℓ-collision if there exist ℓ two-wise different inputs $x_1, \ldots, x_\ell$ such that $F(x_1) = \ldots = F(x_\ell)$.

Theorem 6. *The digital signature scheme* $(\mathrm{Gen}, \mathrm{Sig}^H, \mathrm{Ver}^H)$ *defined above by the Equations (8,9,10) is* (n, t, ϵ)*-secure against adaptive attacks for existential forgery where* H *is uniformly distributed provided that the discrete logarithm is intractable within a time* $44tn\ell \log 2\ell/\epsilon$ *operations, and that we cannot find any* ℓ*-collisions within the same time for* F*.*

Proof. By similar argument, an attacking algorithm will have a fork on (m, r). If we have an ℓ-fork, we have ℓ two-wise different signatures (r, s_i) for the same message m. For this, we need another improved forking lemma we prove below. Hence, the values $t_i = g^{h_i/s_i} y^{r/s_i}$ cannot be two-wise different (otherwise, they make an ℓ-collision), so we have an equation $g^{h_i/s_i} y^{r/s_i} = g^{h_j/s_j} y^{r/s_j}$ which leads to the secret key x.

Simulating the Sig algorithm to forge a valid (h, r, s) from a message m is easy: we first pick two random values for h/s and r/s, then we compute r from Equation (10). From r/s and r we compute s, then from h/s and s we compute s. $\square$

Lemma 7 (Improved Forking Lemma). *If there exists an algorithm* $\mathcal{A}$ *which produces a signature within a probability* ϵ *and* n *queries, we can make an algorithm which produces an* ℓ*-fork for the signature scheme within a probability greater than* $1/11$ *by* $4n\ell \log 2\ell/\epsilon$ *executions of* $\mathcal{A}$*.*

We also mention there is a uniform version of this algorithm (see the Appendix).

Proof. Let $q_1, \ldots, q_n$ be the queries to the oracle. We assume without loss of generality that the queries are two-wise different. We also assume that all the necessary oracle values used in the verification scheme are queries, for any accepted oracle. Let $A_1, \ldots, A_n$ be the (random) answers from the oracle. Any execution of the algorithm $\mathcal{A}$ is defined by the random tape ω and the answers to the queries. We denote $\mathcal{G}$ the set of all accepting $(\omega, A_1, \ldots, A_n)$. For such an accepting execution, we denote $I(\omega, A_1, \ldots, A_n)$ the index of the last useful query q_i for the verification process of the signature.

Now we define an algorithm $\mathcal{B}$ which uses $\mathcal{A}$ to obtain an ℓ-fork.

1. First $\mathcal{B}$ runs $\mathcal{A}$ on random $(\omega, A_1, \ldots, A_n)$ until $\mathcal{A}$ accepts. We denote $(\omega^1, A_1^1, \ldots, A_n^1)$ the accepted one and $I_1 = I(\omega^1, A_1^1, \ldots, A_n^1)$.

2. We repeat $4n\ell \log 2\ell/\epsilon$ times the algorithm $\mathcal{A}$ on $(\omega^1, A_1^1, \ldots, A_{I_1-1}^1)$ tailed with random $(A_{I_1}, \ldots, A_n)$. If we obtain a success such that I_1 is the index of the last useful query, we register the input as a new $(\omega^j, A_1^j, \ldots, A_n^j)$.

Now we will prove we obtain an ℓ-fork $(\omega^j, A_1^j, \ldots, A_n^j)$ for $j = 1, \ldots, \ell$ with probability greater than 1/11.

Let

$$\mathcal{I} = \left\{ i; \Pr\left[I(\omega, A_1, \ldots, A_n) = i/(\omega, A_1, \ldots, A_n) \in \mathcal{G}\right] \geq \frac{1}{2n} \right\}$$

be the set of all i's such that the probability that an accepting execution has its last useful index equal to i greater than $1/2n$. We have

$$\Pr\left[I_1 \in \mathcal{I}\right] = 1 - \Pr\left[\exists i \notin \mathcal{I} \quad I_1 = i\right] \geq 1 - n\frac{1}{2n}$$

so this probability is greater than 1/2.

For each i in $\mathcal{I}$, let $\mathcal{G}_i$ be the set of all $(\omega, A_1, \ldots, A_n)$ in $\mathcal{G}$ such that $I = i$ and let

$$\mathcal{G}'_i = \left\{ (\omega, A_1, \ldots, A_{i-1}); \Pr\left[(\omega, A_1, \ldots, A_n) \in \mathcal{G}_i\right] \geq \frac{\epsilon}{4n} \right\}$$

the set of all heads $(\omega, A_1, \ldots, A_{i-1})$ such that the probability over all tailing $(A_i, \ldots, A_n)$ to be in $\mathcal{G}_i$ is greater than $\epsilon/4n$. From an usual technic, since we have $\Pr[(\omega, A_1, \ldots, A_n) \in \mathcal{G}_i] \geq \epsilon/2n$

$$\begin{aligned} \Pr\left[(\omega, A_1, \ldots, A_{i-1}) \in \mathcal{G}'_i/(\omega, A_1, \ldots, A_n) \in \mathcal{G}_i\right] &= 1 - \Pr\left[\mathcal{G}_i/\bar{\mathcal{G}}'_i\right] \frac{\Pr\left[\bar{\mathcal{G}}'_i\right]}{\Pr\left[\mathcal{G}_i\right]} \\ &\geq 1 - \frac{\epsilon}{4n}\frac{2n}{\epsilon} = \frac{1}{2} \end{aligned}$$

so the first step gets such an $(\omega, A_1, \ldots, A_{i-1})$ (with a good I) with probability greater than 1/4: we have

$$\Pr\left[\left(\omega, A_1^1, \ldots, A_{I_1-1}^1\right) \in \mathcal{G}'_{I_1} \text{ and } I_1 \in \mathcal{I}\right] \geq \frac{1}{4} \tag{11}$$

Now we assume the first step of $\mathcal{B}$ gets an I_1 in $\mathcal{I}$ and $(\omega, A_1^1, \ldots, A_{I_1-1}^1)$ in $\mathcal{G}'_{I_1}$.

If we run $(4n/\epsilon)\log 2\ell$ times the algorithm $\mathcal{A}$ with new random $(A_{I_1}, \ldots, A_n)$, we get at least a fork on I_1 within a probability greater than

$$1 - \left(1 - \frac{\epsilon}{4n}\right)^{\frac{4n \log 2\ell}{\epsilon}}$$

which tends towards $1 - 1/2\ell$ and stay greater than this limit. If we do this ℓ times, we obtain ℓ forks with probability greater than e^{-1}. If ℓ^2 is small against the number of possible tails, there is no collision between the forks, and we obtain an ℓ-fork. Thus, the algorithm $\mathcal{B}$ produce an ℓ-fork within a probability greater than 1/11 and essentially less than $4n\ell \log 2\ell/\epsilon$ executions of $\mathcal{A}$. $\square$

Hence, in order to use a definite F function, either we prove it has no ℓ-collision for a reasonably small ℓ, or we use a probabilistic argument.

Lemma 8. *If F is a random function from a set with q elements to itself, the probability it has no $\log q$-collision tends to zero.*

Proof. We have

$$\begin{aligned}
\Pr[F \text{ has no } \ell\text{−collision}] &\leq q \Pr[\#F^{-1}(0) \geq \ell] \\
&= q \sum_{i=\ell}^{q} \binom{q}{i} \left(\frac{1}{q}\right)^{i} \left(1 - \frac{1}{q}\right)^{q-i} \\
&\leq q \left(1 - \frac{1}{q}\right)^{q} \sum_{i=\ell}^{q} \frac{1}{i!} \left(1 - \frac{1}{q}\right)^{-i} \\
&\leq q e^{-1} e^{\left(1-\frac{1}{q}\right)} \frac{1}{\ell!} \left(1 - \frac{1}{q}\right)^{-\ell} \\
&\leq e \frac{q}{\ell!} \left(1 - \frac{1}{q}\right)^{-\ell}
\end{aligned}$$

So if ℓ is negligible against q, the probability is approximately less than $eq/\ell!$, which can be very small for a reasonable ℓ. More precisely, if $\ell = \log q$, the righthand term of previous Inequality is negligible against $q^{3-\log\log q}$ which tends towards zero. □

For instance, if $q \approx 2^{256}$, we notice that the probability that F has no 50-collision is less than 2^{-43}. So, for a random choice of (p, q, g), the mapping $x \mapsto x \bmod q$ on the subgroup spanned by g has no 50-collision unless we are very unlucky or there is some unknown mathematical property.

We notice that using $\ell = \log q$ and $q > 2^{80}$ we have $\log\log q > 4$ so we have no ℓ-collisions with probability greater than $1 - 1/q$.

This result is robust in the sence it is highly improbable the mod q mapping has any easy 2-collision. Indeed, such a collision would lead to an important weakness in the DSA design by an attack similar than Vaudenay's [41]. If for a given (p, q, g) provided by a honest authority someone happens to find out a 2-collision

$$r = g^k \bmod p \bmod q = g^{k'} \bmod p \bmod q$$

then, by assuming he knows k and k', he can choose two different messages m and m' and choose a particular x as his secret key so that the signature of m and m' collides. Namely, if

$$x = \frac{k\mathrm{SHA}(m') - k'\mathrm{SHA}(m)}{r(k - k')} \bmod q$$

then

$$\frac{\mathrm{SHA}(m) + xr}{k} \bmod q = s = \frac{\mathrm{SHA}(m') + xr}{k'} \bmod q$$

so that he can reveal the signature (r, s) of m and later on claim it was the signature of m'. Since we believe such an attack is impossible, we can apply the Theorem with $\ell = 2$. In this case, we can guaranty undeniability when F and $H \bmod q$ are collision-resistant.

4 Conclusion, application

In this Section we construct a signature scheme which uses results from the previous sections. First of all we assume we are given random p, q and g such that p is a 512-bit prime, q is a 256-bit prime factor of $p-1$ and g is a qth primitive root of unity modulo p. Each entity chooses his own secret key x as a random integer between 1 and $q-1$ and communicates his public key $y = g^x \bmod p$ (which is verifyed to have order exactly q). We agree on a random function F. Every year, each user receives a tamper-resistant module from the trusted authority which implements the function $H(m) = \mathrm{trunc}[\mathrm{SHA}(K_1M)\mathrm{SHA}(K_2M)]$. This is the SHA function truncated to 255 bits so that $H \bmod q$ is collision-resistant if and only if H is collision-free and where $K = (K_1, K_2)$ is a yearly secret parameter. The signature and verification algorithms are described in Section 3.2.

We assume that

1. the tamper-resistant implementation of the function H has an access time limited to $100ms$ (which is reasonably slow)

2. no opponent has access to more than 1000 tamper-resistant modules

3. the validity period is two years

4. H is $(n, 1/2)$-pseudorandom for $n = 2^{40}$ (which corresponds to accesses to 1000 devices H during two years)

5. F has no 50-collisions (which is true with probability 2^{-43})

6. the discrete logarithm problem modulo p in the subgroup spanned by g is intractable within less than 2^{128} multiplications (which corresponds to Shanks's baby-step giant-step algorithm [39]) which cannot be performed within less than $2^{80}s$ (this corresponds to the use of a $10,000,000$-processor parallel machine with processors doing a 512-bit modular multiplication within $400ns$)

so, let $t = 2^{26}s$ equal to two years and $\epsilon = 1/2$. From Theorem 6 we obtain that the signature scheme is secure against any adaptive attacks for existential forgery which runs within a time less than two years with probability of success greater than 1/2 when H is random. Then from Theorem 4 we obtain that our signature scheme is secure as well. Signatures are moreover undeniable provided that both F and H are collision-resistant.

We presented an approach on how to construct practical ElGamal-like signature schemes provably as secure as the discrete logarithm problem. We also pointed out what is the suitable security hypothesis on the hash function for providing secure signatures. The question of removing the hypothesis on the tamper-resistance hypothesis is still an open problem.

Note

The first author works in the University of Caen. The second one in the Ecole Normale Supérieure. Both in laboratories (GREYC, LIENS) supported by the Centre National pour la Recherche Scientifique.

References

[1] Data Encryption Standard. *Federal Information Processing Standard Publication 46*, U.S. National Bureau of Standards, 1977.

[2] U.S. Department of Commerce, National Institute of Standards and Technology. Secure Hash Standard. Federal Information Processing Standard Publication 180-1, 1995.

[3] U.S. Department of Commerce, National Institute of Standards and Technology. Digital Signature Standard. Federal Information Processing Standard Publication 186, 1994.

[4] R. Anderson, S. Vaudenay. Minding your p's and q's. To appear in the Asiacrypt'96 proceedings.

[5] D. Bayer, S. Haber, W. S. Stornetta. Improving the efficiency and reliability of digital time-stamping. In *Sequences II, Methods in Communication, Security and Computer Science*, pp. 329–334, Springer-Verlag, 1993.

[6] M. Bellare, P. Rogaway. Random oracle are practical: a paradigm for designing efficient protocols. In *Proceedings of the 1st IEEE Symposium on Foundations of Computer Science*, Fairfax, Virginia, U.S.A., pp. 62–73, IEEE, 1993.

[7] M. Bellare, P. Rogaway. The exact security of digital signatures – How to sign with RSA and Rabin. In *Advances in Cryptology EUROCRYPT'96*, Zaragoza, Spain, Lectures Notes in Computer Science 1070, pp. 399–416, Springer-Verlag, 1996.

[8] D. Bleichenbacher. Generating ElGamal signatures without knowing the secret key. In *Advances in Cryptology EUROCRYPT'96*, Zaragoza, Spain, Lectures Notes in Computer Science 1070, pp. 10–18, Springer-Verlag, 1996.

[9] M. Blum, S. Micali. How to generate cryptographically strong sequences of pseudorandom bits. In *Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science*, Chicago, Illinois, U.S.A., pp. 112–117, IEEE, 1982.

[10] E. F. Brickell. Invited lecture given at the Crypto'96 conference. Unpublished.

[11] L. Carter, M. Wegman. Universal hash functions. *Journal of Computer and System Sciences*, vol. 18, pp. 143–154, 1979.

[12] I. B. Damgård. Collision free hash functions and public key signature schemes. In *Advances in Cryptology EUROCRYPT'87*, Amsterdam, Holland, Lectures Notes in Computer Science 304, pp. 203–216, Springer-Verlag, 1988.

[13] I. B. Damgård. *The Application of Claw Free Functions in Cryptography*, PhD Thesis, Aarhus University, Mathematical Institute, 1988.

[14] H. Dobbertin. Cryptanalysis of MD4. In *Fast Software Encryption*, Cambridge, United Kingdom, Lectures Notes in Computer Science 1039, pp. 53–69, Springer-Verlag, 1996.

[15] H. Dobbertin. *CryptoBytes*, vol. 2, num. 2, pp. 1–6, 1996.

[16] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *IEEE Transactions on Information Theory*, vol. IT-31, pp. 469–472, 1985.

[17] O. Goldreich, S. Goldwasser, S. Micali. How to construct random functions. In *Proceedings of the 25th IEEE Symposium on Foundations of Computer Science*, Singer Island, U.S.A., pp. 464–479, IEEE, 1984.

[18] S. Goldwasser, S. Micali, R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal of Computing*, vol. 17, pp. 281–308, 1988.

[19] M. Girault, J. Stern. On the length of cryptographic hash-values used in identification schemes. In *Advances in Cryptology CRYPTO'94*, Santa Barbara, California, U.S.A., Lectures Notes in Computer Science 839, pp. 202–215, Springer-Verlag, 1994.

[20] S. Haber, W. S. Stornetta. How to timestamp a digital document. *Journal of Cryptology*, vol. 3, pp. 99–111, 1991.

[21] A. Hodges. *Alan Turing: The Enigma of Intelligence*, Unwin Paperbacks, 1985.

[22] M. Luby, C. Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM Journal on Computing*, vol. 17, pp. 373–386, 1988.

[23] U. M. Maurer. A simplified and generalized treatment of Luby-Rackoff pseudorandom permutation generators. In *Advances in Cryptology EUROCRYPT'92*, Balatonfüred, Hungary, Lectures Notes in Computer Science 658, pp. 239–255, Springer-Verlag, 1993.

[24] R. Merkle. *Secrecy, Authentication and Public Key Systems*, UMI Research Press, 1979.

[25] R. Merkle. One way hash functions and DES. In *Advances in Cryptology CRYPTO'89*, Santa Barbara, California, U.S.A., Lectures Notes in Computer Science 435, pp. 428–446, Springer-Verlag, 1990.

[26] P. C. van Oorschot, M. J. Wiener. On Diffie-Hellman key agreement with short exponents. In *Advances in Cryptology EUROCRYPT'96*, Zaragoza, Spain, Lectures Notes in Computer Science 1070, pp. 332–343, Springer-Verlag, 1996.

[27] J. Patarin. *Etude des Générateurs de Permutations Basés sur le Schéma du D.E.S.*, Thèse de Doctorat de l'Université de Paris 6, 1991.

[28] D. Pointcheval. *Les Preuves de Connaissance et leurs Preuves de Sécurité*, PhD Thesis, Université de Caen, 1996.

[29] D. Pointcheval, J. Stern. Security proofs for signature schemes. In *Advances in Cryptology EUROCRYPT'96*, Zaragoza, Spain, Lectures Notes in Computer Science 1070, pp. 387–398, Springer-Verlag, 1996.

[30] D. Pointcheval, J. Stern. Provably Secure Blind Signature Schemes. To appear in the Asiacrypt'96 proceedings.

[31] B. Preneel. *Analysis and Design of Cryptographic Hash Functions*, PhD Thesis, Katholieke Universiteit Leuven, Departement Elektrotechniek, 1993.

[32] M. O. Rabin. Digitalized signatures. In *Foundations of Secure Computation*, R. Lipton and R. DeMillo Eds, pp. 155–166, Academic Press, New York, 1978.

[33] R. L. Rivest. The MD4 message digest algorithm. In *Advances in Cryptology CRYPTO'90*, Santa Barbara, California, U.S.A., Lectures Notes in Computer Science 537, pp. 303–311, Springer-Verlag, 1991.

[34] R. L. Rivest. The MD5 message-digest algorithm. *Request for Comments 1321*, Internet Activities Board, Internet Privacy Task Force, 1992.

[35] C. P. Schnorr. Efficient identification and signature for smart cards. In *Advances in Cryptology CRYPTO'89*, Santa Barbara, California, U.S.A., Lectures Notes in Computer Science 435, pp. 239–252, Springer-Verlag, 1990.

[36] C. P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, vol. 4, pp. 161–174, 1991.

[37] C. P. Schnorr. Security of 2^t-root identification and signatures. In *Advances in Cryptology CRYPTO'96*, Santa Barbara, California, U.S.A., Lectures Notes in Computer Science 1109, pp. 143–156, Springer-Verlag, 1996.

[38] A. Shamir. On the generation of cryptographically strong pseudo-random sequences. In *8th International Colloquium on Automata, Languages and Programming*, Lectures Notes in Computer Science 62, pp. 544–550, Springer-Verlag, 1981.

[39] D. Shanks. Class number, a theory of factorization, and genera. In *Proceedings of the Symposium on Pure Mathematics*, pp. 415–440, AMS, 1971.

[40] C. E. Shannon. Communication theory of secrecy systems. *Bell system technical journal*, vol. 28, pp. 656–715, 1949.

[41] S. Vaudenay. Hidden collisions on DSS. In *Advances in Cryptology CRYPTO'96*, Santa Barbara, California, U.S.A., Lectures Notes in Computer Science 1109, pp. 83–88, Springer-Verlag, 1996.

[42] A. C. Yao. Theory and applications of trapdoor functions. In *Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science*, Chicago, Illinois, U.S.A., pp. 80–91, IEEE, 1982.

Appendix: uniform version of the forking lemma

Lemma 9 (Uniform Improved Forking Lemma). *There exists a uniform reduction which transforms any signing algorithm $\mathcal{A}$ into an ℓ-fork for the signature scheme within a probability greater than $1/224$ by $12n\ell \log \ell/\epsilon$ executions of $\mathcal{A}$ when $\mathcal{A}$ is limited to n queries to the oracles an makes a signature within a probability ϵ.*

Proof. The idea consists in estimating the $1/\epsilon$ value by using the running time T_1 of the first step of $\mathcal{B}$:

1. First $\mathcal{B}$ runs $\mathcal{A}$ on random $(\omega, A_1, \ldots, A_n)$ until $\mathcal{A}$ accepts. We denote T_1 the number of iterations used. We denote $(\omega^1, A_1^1, \ldots, A_n^1)$ the accepted one and $I_1 = I(\omega^1, A_1^1, \ldots, A_n^1)$.

2. We repeat $8nT_1\ell\log\ell$ times the algorithm $\mathcal{A}$ on $(\omega^1, A_1^1, \ldots, A_{I_1-1}^1)$ tailed with random $(A_{I_1}, \ldots, A_n)$. If we obtain a success such that I_1 is the index of the last useful query, we register the input as a new $(\omega^j, A_1^j, \ldots, A_n^j)$.

The distribution of T_1 tends towards a Poisson distribution. We have

$$0.14 \le \Pr\left[\frac{1}{2\epsilon} \le T_1 \le \frac{3}{2\epsilon}\right] \le 0.56$$

hence this probability is greater than $1/7$.

In the analysis, we replace Equation 11 by

$$\Pr\left[\left(\omega, A_1^1, \ldots, A_{I_1-1}^1\right) \in \mathcal{G}'_{I_1} \text{ and } I_1 \in \mathcal{I} \text{ and } \frac{1}{2\epsilon} \le T_1 \le \frac{3}{2\epsilon}\right] \ge \frac{1}{28}$$

and we obtain a probability of success of $1/224$ and $12n\ell\log\ell/\epsilon$ runs of $\mathcal{A}$. $\square$