Proceedings of Financial Cryptography '2001
(19–22 february 2001, Grand Cayman Island, British West Indies)
P. Syverson Ed., Springer-Verlag, LNCS 2339, pages 305–318.

Monotone Signatures

David Naccache[1], David Pointcheval[2], and Christophe Tymen[1]

[1] Gemplus Card International – 34, rue Guynemer
F-92447 Issy-les-Moulineaux cedex, France
david.naccache,christophe.tymen@gemplus.com – www.gemplus.com/smart
[2] Ecole Normale Supérieure, 45 rue d'Ulm, F-75230 Paris cedex 5, France
david.pointcheval@ens.fr – www.di.ens.fr/users/pointche

Abstract. In many real-life situations, massive quantities of signatures have to be issued on cheap passive supports (*e.g.* paper-based) such as bank-notes, badges, ID cards, driving licenses or passports (hereafter IDs); while large-scale ID replacements are costly and prohibitive, one may reasonably assume that the updating of verification equipment (*e.g.* off-line border checkpoints or mobile patrol units) is exceptionally acceptable.
In such a context, an attacker using coercive means (*e.g.* kidnapping) can force the system authorities to reveal the infrastructure's secret signature keys and start issuing signatures that are indistinguishable from those issued by the authority.
The solution presented in this paper withstands such attacks up to a certain point: after the theft, the authority restricts the verification criteria (by an exceptional verification equipment update) in such a way that the genuine signatures issued before the attack become easily distinguishable from the fresher signatures issued by the attacker.
Needless to say, we assume that at any point in time the verification algorithm is entirely known to the attacker.

Keywords: Digital Signatures, Coercion, Bank Notes, ID Cards.

1 Introduction

In settings where passive (paper-based) bank notes, passports or ID cards are massively delivered to users, document security specialists (*e.g.* [22]) distinguish between two different threats:

- *duplication*, which consists in copying information from a genuine document into a new physical support (the copy). By analogy to the *double-spending problem* met in e-cash schemes and software copyright protection, it seems impossible to prevent duplication without relying on specific physical assumptions, simply because symbols are inherently copyable. This difficulty explains why duplication is mainly fought by optical means such as holograms, iridescent printing (different colors being displayed at different angles of observation), luminescent effects (the emission of radiation by an atom in the course of a transition from a higher to a lower state of energy, which is typically achieved by submitting the ID to ultraviolet excitation) or standard document security features such as planchettes, fibers and thread.
  In the last decade, chip-based IDs appeared (*e.g.* Venezuela's driving license). Again, these are based on the assumption that appropriately designed microchips can reasonably withstand malicious cloning attempts.
- *forgery*, which assumes that attackers have successfully passed the physical barrier and are now able to reproduce documents using exactly the same materials and production techniques used to create the original. Note that although forgers may copy any existing ID, they can still fail in creating new contents *ex nihilo* if the ID happens to rely on logical protections such as MACs or signatures.

It seems very hard to quantify or compare the security of physical anti-duplication technologies; partially because the effectiveness of such solutions frequently relies on their secrecy, let alone the wide diversity of physical technologies mixed in one specific protection. By opposition, the protection of digital assets against alteration is much better understood and can be easily used to fight forgery.

As is obvious, if the authority's signature or MAC keys are compromised (*e.g.* by theft, cryptanalysis or coercion) forgery becomes possible, and the whole system collapses. Theft can be easily prevented by physically protecting the production facility or better more, by having data signed in protected remote locations and by exchanging information and signatures through a properly protected logical channel.

This is however not sufficient to resist coercion, a *scenario* in which the attacker uses a threat (*e.g.* a kidnapping) to force the authorities to publish the signature keys (*e.g.* in a newspaper [21]). The attacker can then check *in vitro* the correctness of the revealed keys, stop coercing and start issuing fake IDs that are indistinguishable from the genuine ones. The attack can also be motivated by the sole intention to cause losses (global ID replacement).

Large scale ID replacement is, of course, a radical solution but it may both entail prohibitive costs and require a transition period during which intruders can still sneak through the borders. A second solution consists in performing systematic on-line verifications to make sure that all controlled IDs are actually listed somewhere, but this might be cumbersome in decentralized or poorly networked infrastructures.

As mentioned in the abstract, the problem is, of course, not limited to IDs. Bank notes, public-key directories and any other passive supports carrying signatures or MACs are all equally concerned.

Several authors formalized similar concerns [9] and solutions based on pro-active key updates [8] which, although very efficient in on-line contexts (*e.g.* Internet), do not suit our passive (non-intelligent) IDs; others share the key between n individuals amongst which a *quorum* of k is necessary to sign [10, 19]. This does not seem to solve the fundamental coercion problem either, since the forger can force the authority to instruct k of the share-holders to reveal their secrets, or coerce k share-holders directly.

2 The Idea and a Few Definitions

The proposed solution targets the attacker's ability to ascertain the correctness of the stolen keys; this is achieved by updating the verification algorithm $\mathcal{V}$ so as to distinguish the fake (new) signatures from the genuine (old) ones. We denote by $\{\mathcal{V}_1, \ldots, \mathcal{V}_n\}$ the successive updates of $\mathcal{V}$ in a system designed to withstand at most n coercions.

In our system, the authority's (genuine) signatures are designed to:

- remain *forward compatible i.e.* be valid for all the verification algorithms $\mathcal{V}_i$ to come.

– remain *computationally indistinguishable* from the signatures generated by the i-th attacker until the disclosure of $\mathcal{V}_{i+1}$.

The technique is thus analogous to the strategy of national banks who implement several (secret) security features in their bank notes. As forgeries appear, the banks examine the fakes and publicize some of the secret features to stop the circulation of forgeries.

Our construction relies on the following definitions:

Definition 1 (Monotone Predicates). Let $\mathcal{V}_1(x), \ldots, \mathcal{V}_n(x)$ be n predicates. The set $\{\mathcal{V}_i(x)\}$ is *monotone* if

$$\forall i < n, \quad \mathcal{V}_{i+1}(x) \Rightarrow \mathcal{V}_i(x)$$

Example 2. The set of predicates:

$$\begin{aligned}
\mathcal{V}_1(x) &\stackrel{\text{def}}{=} x \in \mathbb{R} \\
\mathcal{V}_2(x) &\stackrel{\text{def}}{=} x \in \mathbb{N} \\
\mathcal{V}_3(x) &\stackrel{\text{def}}{=} x \text{ is prime} \\
\mathcal{V}_4(x) &\stackrel{\text{def}}{=} x \text{ is a strong prime}
\end{aligned}$$

is monotone since

$$\mathcal{V}_4(x) \Rightarrow \mathcal{V}_3(x) \Rightarrow \mathcal{V}_2(x) \Rightarrow \mathcal{V}_1(x).$$

Definition 3 (Signature Schemes). A *signature scheme* is a collection of three sub-algorithms $\{\mathcal{G}, \mathcal{S}, \mathcal{V}\}$,

– a probabilistic key-generation algorithm $\mathcal{G}$, which produces a pair of related public and secret keys, on input a security parameter k: $\{v, s\} = \mathcal{G}(1^k)$, where v and s respectively denote the public and secret keys used by $\mathcal{V}$ and $\mathcal{S}$, the verification and the signature algorithms (see below).
– a possibly probabilistic signature algorithm $\mathcal{S}$, which produces a signature, given a secret key and a message: $\sigma = \mathcal{S}(s; m)$.
– a verification algorithm, which checks whether the given signature is correct relatively to the message and the public key: $\mathcal{V}(v; m, \sigma) \in \{\mathsf{true}, \mathsf{false}\}$. It must satisfy

$$(\sigma = \mathcal{S}(s; m)) \Rightarrow (\mathcal{V}(v; m, \sigma) = \mathsf{true}).$$

Definition 4 (Monotone Signature Schemes). A *monotone signature scheme (MSS)* is the following generalization of definition 3,

– a probabilistic key-generation algorithm $\mathcal{G}$, which produces a list of public and secret keys, on input two security parameters k and n:

$$\{v_1, \ldots, v_n, s_1, \ldots, s_n\} = \mathcal{G}(1^k, 1^n),$$

where $\{v_i\}$ and $\{s_i\}$ respectively denote the public and secret keys used by the $\mathcal{V}_j$ and $\mathcal{S}$.
– a possibly probabilistic signature algorithm $\mathcal{S}$, which produces a signature, given the list of the n secret keys and a message: $\sigma = \mathcal{S}(s_1, \ldots, s_n; m)$.

– a list of monotone verification algorithms $\mathcal{V}_j$ which check whether the given signature is correct, relatively to the message and the list of public keys:

$$\mathcal{V}_j(v_1, \ldots, v_j; m, \sigma) \in \{\mathsf{true}, \mathsf{false}\}.$$

In other words, we require the three following properties.

1. *completeness*:

$$\sigma = \mathcal{S}(s_1, \ldots, s_n; m) \Rightarrow \forall j \leq n,\ \mathcal{V}_j(v_1, \ldots, v_j; m, \sigma) = \mathsf{true}.$$

2. *soundness*: for any adversary $\mathcal{A}$ which does not know s_{j+1}, the probability, over his internal random coins, to produce an accepted message-signature pair $\{m, \sigma\}$ is negligible

$$\Pr[\mathcal{V}_{j+1}(v_1, \ldots, v_{j+1}; m, \sigma) = \mathsf{true} \,|\, (m, \sigma) = \mathcal{A}] \text{ is negligible.}$$

3. *Indistinguishability*: for any index $j \leq n$, there exists a simulator $\mathcal{S}_j$ such that the distributions of $\mathcal{S}(s_1, \ldots, s_n; x)$ and $\mathcal{S}_j(s_1, \ldots, s_j; x)$, for the internal random coins of the algorithms, are indistinguishable by opponents who do not possess $\{s_{j+1}, \ldots, s_n\}$.

We now categorize the opponents that MSSs will withstand. In essence we consider two types of attackers: *immediate* and *delayed*. Both are going to coerce the signer, get some of his secrets, check their validity (as much as possible, *i.e.* with respect to the currently enforced public-key $\{v_1, \ldots, v_j\}$) and start forging.

Definition 5 (Immediate Attackers). *Immediate attackers* forge signatures using the obtained secret keys $\{s_1, \ldots, s_j\}$, but stop their activity as soon as the new verification algorithm $\mathcal{V}_{j+1}(v_1, \ldots, v_{j+1}; \cdot, \cdot)$ is published.

The next section will be devoted to the study of the long-term validity of such forgeries, produced before $\mathcal{V}_{j+1}(v_1, \ldots, v_{j+1}; \cdot, \cdot)$ is known.

Definition 6 (Delayed Attackers). *Delayed attackers* wait until a new verification algorithm $\mathcal{V}_{j+1}(v_1, \ldots, v_{j+1}; \cdot, \cdot)$ is published and use both the obtained secret keys $\{s_1, \cdots, s_j\}$ and the new verification rules to compute their forgeries.

The global picture is presented on figure 1.

3 Immediate Attacks and Symmetric Monotone Signatures

As one may suspect, immediate attackers are the easiest to deal with. In theory, the situation does not even call for the use of asymmetric primitives. It suffices to add secret information to m or σ (*e.g.* using a subliminal channel as suggested by [20]) but unless secret keys are shared with the verifiers, which is not the case in our setting, the information rate is very low (narrow-band subliminal channel).

- key generation: the authority gets $\{v_1, \ldots, v_n, s_1, \ldots, s_n\} = \mathcal{G}(1^k, 1^n)$
- keys: the authority keeps $\{s_1, \ldots, s_n\}$ secret
  and publishes $\{v_1, \ldots, v_j\}$ for some $j < n$
- signature generation: the authority runs $\mathcal{S}(s_1, \ldots, s_n; m)$ to sign m
- coercion
  - start
  - the authority reveals, to the attackers, the signature algorithm, together with the secret keys $\{s_1, \cdots, s_j\}$
  - stop
- *immediate attackers* try to issue signatures using only $\{s_1, \cdots, s_j\}$ and $\mathcal{V}_j$
- authority updates $\mathcal{V}_j$ to $\mathcal{V}_{j+1}$ and informs the verifiers
- *delayed attackers* try to issue signatures using $\{s_1, \cdots, s_j\}$, $\mathcal{V}_j$ and $\mathcal{V}_{j+1}$

Fig. 1. Coercion Model

Better results are obtained by adding to σ some hidden randomness. In other words, the actually signed message will be $\mu(m, r)$ where μ is a padding function and r a randomly-looking (pseudo-random) bit string. The expression *randomly-looking* translates the fact that r embeds information which is meaningful to who knows how to interpret it :

$$\begin{array}{ll} \text{let} & r =< r_1 \ldots r_n > \in \{0,1\}^n \\ \text{and} & \begin{cases} r_i = f_{k_i}(\{r_\lambda\}_{\lambda \in E'}) \text{ for all } i \in E \subseteq \{1, \ldots, n\}, \\ r_i \in_R \{0,1\} \text{ for all } i \notin E, \end{cases} \end{array}$$

where E and E' are two disjoint subsets of $\{1, \ldots, n\}$; $\{f_i\}$ is a family of pseudo-random functions returning one bit; and the values $\{k_i\}$, for $i \in E$, are auxiliary secret keys. More concretely, the set E' contains the indices of the bits used for generating redundancy, and the set E contains the indices of the redundancy bits.

The signer knows s as well as the complete collection of auxiliary secrets $\{k_i\}$. To issue an ID containing m, he generates a randomly looking r (which satisfies the required secret redundancy) and a signature σ of $\mu(m, r)$. The ID contains $\{m, r, \sigma\}$.

The verifier knows v and the values of some k_i, for $i \in F \subseteq E$. Upon presentation of the ID, he verifies the redundancy of r with respect to the k_i values that *he* knows. If this succeeds, he proceeds and verifies σ.

After coercion, the attacker obtains s and the k_i for $i \in G$ with, at least, $F \subseteq G$ (recall that the attacker verifies the validity of the produced signatures before stopping coercion). As long as $G \neq E$, the verification algorithm can be fixed and the system saved.

After revealing H (strictly bigger than G) and the k_i, for $i \in H$, signatures are considered valid if and only if all the r_i for $i \in H$ are correct. Given the unpredictable nature of the $\{r_i\}$ for $i \in H \backslash G$ (and well-chosen functions $\{f_i\}$), the forged signatures are accepted with probability smaller then $\epsilon = 2^{-c}$ where $c = \#(H \backslash G)$. If c is sufficiently large, ϵ is negligible and the forgeries are almost certainly spotted.

Figure 2 describes this protocol that we call *symmetric MSS*, since it relies on auxiliary *secrets*, eventually revealed to the verifiers. More formally, the ver-

Initialization
$\{\mathcal{G}, \mathcal{S}, \mathcal{V}\}$, signature scheme $\{f_k\}$, pseudo-random family of functions
Key generation
Generation of $\{s, v\}$ with $\mathcal{G}$ select two disjoint subsets E and E' of $\{1, \ldots, n\}$ $\forall i \in E, k_i \in_R \{0,1\}^{128}$ **Public**: v and E', and some $F \subset E$ (which determines the degree of verification) **Private**: s, E and the k_i
Signature
$\forall i \notin E, r_i \in_R \{0,1\}$ $\forall i \in E, r_i = f_{k_i}(\{r_\lambda\}_{\lambda \in E'}) \in \{0,1\}$ $h = H(m\|r)$ and $\sigma = \mathcal{S}(s; h)$
Verification of $\{m, r, \sigma\}$ for $F \subseteq E$
make sure that for all $i \in F$, $r_i = f_{k_i}(\{r_\lambda\}_{\lambda \in E'})$ compute $h = H(m\|r)$ and check that $\mathcal{V}(v; h, \sigma) = \mathsf{true}$

Fig. 2. Symmetric Monotone Signature Scheme

ification algorithm $\mathcal{V}_F$ checks the validity of the signature σ, but furthermore checks the redundancy of all the bits indexed by F. We can state the following theorem.

Theorem 7. *Let $\{\mathcal{G}, \mathcal{S}, \mathcal{V}\}$ be a signature scheme transformed into a symmetric MSS as suggested in figure 2.*

- *The signatures issued by the authority leak no information on the subset E;*
- *Assume that an attacker manages to obtain s, and then the k_i for $i \in G \supseteq F$. Let $H \subseteq E$ be such that G is strictly included in H. Let us denote by c the cardinality of $H \backslash G$. The signatures issued by an attacker knowing G will be accepted with respect to H with probability smaller than 2^{-c}.*

Proof. First assume that $f_k(.) = f(k, .)$, where f is, in the first part of the proof, modeled by a random oracle which outputs one bit to each query:

- The r_i are all random for $i \notin E$, by construction, as well as for $i \in E$ because of the randomness of f. Therefore, the signatures do not reveal any information on E (other than the fact that $F \subseteq E$).
- By virtue of this indistinguishability property for E in $\{1, \ldots, n\}$, the attacker can not know if G is the entire set E. Assume that this is not the case and G is strictly included in E. Define H as an intermediate subset, $G \subset H \subseteq E$, and let c denote the cardinality of $H \backslash G$. Since f is a random oracle, without knowing the k_j for $j \in H \backslash G$, the attacker can not produce the valid r_j bits without a bias. Therefore the probability to produce a valid forgery is smaller than 2^{-c}.

Now, if by replacing f (secret random oracle [13]) by the family f_k, the attacker manages to produce valid signatures with probability larger than $2^{-c} + \alpha$, then the attacker can be used as distinguisher between the family of functions $\{f_k\}$ and a perfectly random function with an advantage α, which contradicts the assumption that $\{f_k\}$ is a family of pseudo-random functions. □

Given the symmetric nature of the auxiliary secrets (except the unique asymmetric private key revealed immediately after an attack), it is clear that this process can not withstand *delayed* attacks. Actually, the information owned by the verifier after the update is sufficient for producing valid forgeries. We therefore focus the coming section on asymmetric MSS that can thwart delayed attacks.

4 Delayed Attacks and Asymmetric Monotone Signatures

4.1 Simple Concatenation

A trivial example of asymmetric MSS can be obtained by concatenating signatures:

- Let $\{\mathcal{G}, \mathcal{S}, \mathcal{V}\}$ be a signature scheme and denote by ℓ the size of each signature;
- The concatenated signature of m over the set $E \subseteq \{1, \ldots, n\}$, is the tuple:

$$\mathcal{S}'\left(\{s_i\}_{i\in E}; m\right) = \sigma = \{\sigma_1, \ldots, \sigma_n\}$$
$$\text{where } \sigma_i = \begin{cases} \mathcal{S}(s_i; m) & \text{if } i \in E, \text{ using the secret key } s_i \\ \rho_i \in_R \{0,1\}^\ell & \text{if } i \notin E \end{cases}$$

- Verification consists in evaluating the predicate:

$$\mathcal{V}'_F(\{v_i\}_{i\in F}; m, \sigma) = \wedge_{i\in F}\mathcal{V}(v_i; m, \sigma_i),$$

  where the set $F \subseteq E$ determines the degree of verification.

However, for E not to be detectable, the two following distributions must be indistinguishable, for any pair $\{s, m\}$ of secret key and message:

$$\delta_0 = \{\rho \in_R \{0,1\}^\ell\}$$
$$\delta_1(s, m) = \{\mathcal{S}(s, m)\}$$

This latter distribution is over the internal random coins used in the probabilistic signature process. Thus, not all signature algorithms lead themselves to such a construction. For instance, the concatenation of RSA [17] signatures does not yield an asymmetric MSS, because of the deterministic nature of σ as a function of m (unless one uses a probabilistic padding scheme such as PSS [3] or PKCS#1 v 2.0, the distribution $\delta_1(s, m)$ contains only one point, by opposition to the uniform distribution δ_0.)

On the other hand, if the distribution of signatures is indistinguishable from the uniform distribution, a mix between random numbers and signatures of m will resist coercion up to a certain point. We formalize this in the following theorem.

Theorem 8. *Let $\{\mathcal{G}, \mathcal{S}, \mathcal{V}\}$ be a signature scheme for which the distribution $\delta_1(s, m)$ is indistinguishable (for any pair $\{s, m\}$) from the uniform distribution. Let $\{\mathcal{G}', \mathcal{S}', \mathcal{V}'\}$ be the concatenated version of $\{\mathcal{G}, \mathcal{S}, \mathcal{V}\}$.*

- *The signatures produced by the authority do not reveal any information on the subset E;*
- *Consider an attacker $\mathcal{A}$ who got hold of the s_i for $i \in G \supseteq F$. Let $H \subseteq E$ be such that G is strictly included in H, whose associated verification keys have been published. If $\mathcal{A}$ can produce a forgery for $\{\mathcal{G}', \mathcal{S}', \mathcal{V}'\}$ with respect to H then he is able to produce a forgery for $\{\mathcal{G}, \mathcal{S}, \mathcal{V}\}$.*

A second disadvantage of RSA is the size of σ (recall that we actually talk about n such signatures). A more compact alternative is Schnorr's signature. The next paragraph describes a concatenated signature based on this scheme.

4.2 Concatenation of Schnorr's Signatures

We recall the description of the Schnorr's scheme [18]:

- An authority generates a (k_1 bit) prime p such that $p-1$ has a large prime factor q of k_2 bits. The authority also generates an element g of $\mathbb{Z}_p^\star$ of order q and publishes a hash function H which outputs are in $\mathbb{Z}_q$;
- $\mathcal{G}(p, q, g)$ returns $x \in_R \mathbb{Z}_q^\star$ and $y = g^x \bmod p$;
- $\mathcal{S}(x; m) = \{e, s\}$ where $t \in_R \mathbb{Z}_q^\star$, $r = g^t \bmod p$, $e = H(m, r)$ and $s = t - ex \bmod q$;
- $\mathcal{V}(y; m, e, s) = (H(m, g^s y^e \bmod p) \stackrel{?}{=} e)$.

This scheme is provably secure in the random oracle model [16]. More precisely, it withstands existential forgeries even against adaptive chosen-message attacks [7]. Moreover, $\delta_1(x, m) = \{\mathcal{S}(x, m)\} = \{\{e, s\} \in_R \mathbb{Z}_q \times \mathbb{Z}_q\}$ is indistinguishable from a uniform distribution, when y is unknown.

Remark 9. We insist on the format of the Schnorr's signature. Indeed, sometimes one outputs $\{r, s\}$ as the signature, instead of $\{e, s\}$. We use this latter for two reasons:

- Because of the shorter size of the resulting signature. Note that in elliptic curve settings, this is irrelevant, since both representations are as short.
- For the randomly-looking property of the pair $\{e, s\}$. Indeed, to distinguish a list of actual signatures $\{\{e_i, s_i\}\}$, for a given pair of keys $\{x, y\}$, from a list of truly random pairs, one has to find this common y, which can not be found without the r_i (hidden in the query asked to H). But with the r_i, one could easily compute $e_i = H(m, r_i)$ and $(r_i / g^{s_i})^{1/e_i}$. This latter value would be a constant: y.

By virtue of theorem 8, we can construct a concatenated variant that is as secure as the initial scheme, that is, existentially unforgeable against adaptive chosen-message attacks. Figure 3 describes such a variant.

The resulting MSS outputs $2nk_2$ bit signatures, and since usually $k_2 \cong 160$, this would amount to $320n$ bits in practice. Note that efficient batch algorithms for generating and verifying multiple Schnorr's signatures may considerably improve the parties' workloads [12, 1, 11].

Initialization
p, q, g and H as in Schnorr's scheme
Key generation
Select a subset E of $\{1, \ldots, n\}$ $\forall i \in E$, let $x_i \in \mathbb{Z}_q^\star$ and $y_i = g^{x_i} \bmod p$ **Private**: E and the x_i for $i \in E$ **Public**: some $F \subset E$, and y_i for $i \in F$
Signature
$\forall i \in E, \sigma_i = \{e_i, s_i\} = \mathcal{S}(x_i; m)$ $\forall i \notin E, \sigma_i = \{e_i, s_i\} \in_R \mathbb{Z}_q \times \mathbb{Z}_q^\star$ let $\sigma = \{\sigma_1, \ldots, \sigma_n\}$
Verification of $\{m, \sigma\}$ **for** $F \subseteq E$
$\forall i \in F$, $H(m, g^{s_i} {y_i}^{e_i} \bmod p) \stackrel{?}{=} e_i$

Fig. 3. Concatenated Schnorr's Signatures

4.3 Introducing Degrees of Freedom

Instead of concatenating signatures and random values, the asymmetric MSS described in this section relies on hidden relations between the different parts of the signature that give additional degrees of freedom to the signer. It's main advantage over concatenation is a substantial improvement in signature size (50%).

The Okamoto-Schnorr Signature. The new scheme is based on the Okamoto's variant of Schnorr's scheme [15]. The mechanism relies on the representation problem [4], and is recalled in figure 4.

Initialization
p, q and H as in Schnorr's scheme $g_1, \ldots, g_n \in \mathbb{Z}_p^\star$ of order q
Key generation
Private: $x_1, \ldots, x_n \in \mathbb{Z}_q^\star$ **Public**: $y = g_1^{x_1} \times \ldots \times g_n^{x_n} \bmod p$
Signature
$t_1, \ldots, t_n \in \mathbb{Z}_q^\star$ and $r = g_1^{t_1} \times \ldots \times g_n^{t_n} \bmod p$ $e = H(m, r)$ then for $i = 1, \ldots, n$, $s_i = t_i - e x_i \bmod q$ $\mathcal{S}(x_1, \ldots, x_n; m) = (e, s_1, \ldots, s_n)$
Verification
$H(m, g_1^{s_1} \times \ldots \times g_n^{s_n} \times y^e \bmod p) \stackrel{?}{=} e$

Fig. 4. Okamoto–Schnorr Signatures

General outline. The main idea is to impose and keep secret relations between the g_i. For simplicity, suppose that $n = 2$. Instead of choosing g_1 and g_2 at random, we choose g_2 as before, but set $g_1 = g_2^a \bmod p$ and $y = g_2^x \bmod p$, where a is a secret element of $\mathbb{Z}_q^\star$, and thus $x = a x_1 + x_2 \bmod q$ (with the notations of the figure 4). Then we keep the verification condition

$$H(m, g_1^{s_1} g_2^{s_2} y^e) \stackrel{?}{=} e \tag{1}$$

But now, we can choose s_1 as we want (*e.g.* at random), as well as a random t, compute $r = g_2^t \bmod p$, $e = H(m, r)$ and then we want

$$g_2^{s_2} = ry^{-e}g_1^{-s_1} = g_2^t g_2^{-ex} g_2^{-as_1} = g_2^{t-ex-as_1} \bmod p.$$

Therefore, $s_2 = t - ex - as_1 \bmod q$ provides a valid signature. As the signer can choose s_1 arbitrarily (even after having chosen t), we say that he gets an additional *degree of freedom*. This signature will still satisfy the verification formula (1), and will be indistinguishable from a classical Okamoto-Schnorr signature. Furthermore, instead of choosing s_1 at random, we may choose it to be randomly-looking. Explicitly, we may set $s_1 = f_k(m\|r)$ where f_k is a pseudo-random function and k an auxiliary secret. When coerced, the signer reveals x_1 and x_2, but keeps a and k secret. The attacker is thus capable of forging signatures satisfying formula (1). Then, the signer publishes an additional verification condition, namely $s_1 \stackrel{?}{=} f_k(m\|r)$. From that moment, in order to forge valid signatures, the attacker must compute a from g_2^a, or equivalently, find a discrete logarithm in $\mathbb{Z}_p^\star$.

This idea can be generalized to any arbitrary n. We set an i in $\{2, \cdots, n-1\}$, and for $j = 1, \cdots, i-1$, we impose $g_j = g_i^{a_i} \bmod p$, where the a_i are kept secret, and therefore $y = g_i^{x_i} \times \ldots \times g_n^{x_n} \bmod p$ for some tuple $\{x_i, \ldots, x_n\}$. To produce a signature, we proceed as follows: set $r = g_i^{t_i} \cdots g_n^{t_n} \bmod p$, for random t_j. The signer has $i-1$ *degrees of freedom*, that is, he can set, for all $j < i$, $s_j = f_{k_j}(m\|r)$. In addition, to be compatible with the verification condition

$$H(m, g_1^{s_1} \times \ldots \times g_n^{s_n} \times y^e \bmod p) \stackrel{?}{=} e, \tag{2}$$

we set $s_i = t_i - ex_i - a_1 s_1 - \cdots - a_{i-1}s_{i-1} \bmod q$, and $s_k = t_k - ex_k \bmod q$ for $k > i$. Trivially, the verification formula (2) still works for this signature generation:

$$\begin{aligned}
g_1^{s_1} \times \ldots \times g_n^{s_n} \times y^e &= g_i^{a_1 s_1} \times \ldots \times g_i^{a_{i-1}s_{i-1}} \times g_i^{s_i} \times \prod_{k=i+1}^{k=n} g_k^{s_k} \times y^e \\
&= g_i^{a_1 s_1 + \ldots + a_{i-1}s_{i-1}} \times g_i^{t_i - ex_i - a_1 s_1 - \cdots - a_{i-1}s_{i-1}} \times \prod_{k=i+1}^{k=n} g_k^{t_k - ex_k} \times y^e \\
&= g_i^{t_i - ex_i} \times \prod_{k=i+1}^{k=n} g_k^{t_k - ex_k} \times y^e = \prod_{k=i}^{k=n} g_k^{t_k - ex_k} \times \prod_{k=i}^{k=n} g_k^{ex_k} = \prod_{k=i}^{k=n} g_k^{t_k} = r \bmod p.
\end{aligned}$$

But now, we can disclose some partial secrets k_i and a_i to an attacker, and then add new verification conditions as shown in the case $n = 2$.

As a last generalization, we suppress the special role played by the first i indices in the previous construction, and hide the indices of the generators for which one knows some relations. That means that we can apply a secret permutation P to the indices, imposing that $g_{P(j)} = g_{P(i)}^{a_{P(j)}}$ for $1 \le j \le i-1$. The signature generation remains the same, except that the sets $\{1, \cdots, i-1\}, \{i\}$ and $\{i+1, \cdots, n\}$ are replaced respectively by $P\langle\{1, \cdots, i-1\}\rangle$, $\{P(i)\}$ and $P\langle\{i+1, \cdots, n\}\rangle$.

Initialization
p, q and H as in Schnorr's scheme.
$f_k(.) = H(k, .)$, a family of random functions
Key generation
Choose a permutation P of $\{1, 2, \ldots, n\}$
Choose $i < n$, the degree of freedom
Set $E = P(\{1, \cdots, i-1\})$
Choose $F \subset E$
Choose $x_i, \ldots, x_n \in_R \mathbb{Z}_q^\star$
Choose $a_{P(1)}, \ldots, a_{P(i-1)} \in_R \mathbb{Z}_q^\star$
Choose $k_{P(1)}, \ldots, k_{P(i-1)}$ random keys
Choose $g_{P(i)}, g_{P(i+1)}, \ldots, g_{P(n)} \in_R \mathbb{Z}_p^\star$ of order q
Set $g_{P(j)} = g_{P(i)}^{a_{P(j)}} \bmod p$ for $j = 1, \ldots, i-1$
Set $y = g_{P(i)}^{x_i} \times \ldots \times g_{P(n)}^{x_n} \bmod p$
Private: P, $\{a_j, k_j\}_{j \in E}$ and $x_i, \ldots, x_n$
Public: y, g_j for $j = 1, \ldots, n$, F and k_j for $j \in F$
Signature generation
Pick $t_i, \ldots, t_n \in_R \mathbb{Z}_q^\star$
Set $r = g_{P(i)}^{t_i} \times \ldots \times g_{P(n)}^{t_n} \bmod p$
$e = H(m, r)$
Set, for $j = 1, \ldots, i-1$, $s_{P(j)} = f_{k_{P(j)}}(m \| r)$
Set $s_{P(i)} = t_i - e x_i - a_{P(1)} s_{P(1)} - \ldots - a_{P(i-1)} s_{P(i-1)} \bmod q$
Set, for $j = i+1, \ldots, n$, $s_{P(j)} = t_j - e x_j \bmod q$
$\sigma = (e, s_1, \ldots, s_n)$
Verification of (m, σ) **for** $F \subseteq E$
$H(m, g_1^{s_1} \times \ldots \times g_n^{s_n} \times y^e \bmod p) \stackrel{?}{=} e$.
$\forall j \in F$, $s_j \stackrel{?}{=} f_{k_j}(m \| r)$

Fig. 5. Okamoto–Schnorr Signatures with $i-1$ Degrees of Freedom

Formal description of the scheme. The complete protocol is described in figure 5. The validity of this new scheme comes from the fact the

$$g_1^{s_1} \times \ldots \times g_n^{s_n} \times y^e = g_{P(1)}^{s_{P(1)}} \times \ldots \times g_{P(n)}^{s_{P(n)}} \times y^e \bmod p.$$

After the first coercion, the signer reveals $x_1, \cdots, x_n$, for some randomly chosen $x_1, \ldots, x_{i-1}$ thanks to the a_j's. He also reveals a set G, which necessarily satisfies $F \subseteq G \subseteq E$, and the values a_j and k_j for $j \in G$. The point is that G strictly includes the indices possibly known from previous attacks (and thus included in the current public key). If such a G, strictly included in E, exists, the signer can withstand the attack. When the choice of such a G is impossible, the system finally collapses. Note that for the first attack, it is possible to choose $F = \emptyset$.

After the attack, the signer publishes an additional verification condition,

$$s_\kappa \stackrel{?}{=} f_{k_\kappa}(m \| r),$$

where $\kappa \in E \backslash G$. The forgery of valid signatures will require knowing a_κ. For an attacker, this implies determining a_κ from $g_{P(i)}^{a_\kappa}$, and the difficulty of this problem is equivalent to the security of the initial scheme.

Security. We can claim the following security result.

Theorem 10. *Consider the Okamoto-Schnorr signature scheme with $i-1$ degrees of freedom of figure 5, in the random oracle model.*

- *The signatures produced by the authority do not reveal any information on the subset E;*
- *Consider an attacker $\mathcal{A}$ knowing a representation of y, $k < i$ relations between the g_j and k secret keys k_j. If, after revealing one more k_i, $\mathcal{A}$ can still produce a signature accepted by the new verification algorithm, then $\mathcal{A}$ can compute discrete logarithms.*

Proof. We assume H to behave like a random oracle. For the first part of the theorem, using classical simulation techniques ([6, 16]), we can prove that there exists a simulator that does not know any secret value, but which is able to generate signatures that are indistinguishable from the true signatures, thanks to the random oracles simulation (for H but also the f_k's). This simulator proceeds as follows: it chooses e, then the s_j's, and computes the correct value of r. Finally, it sets $H(m, r) = e$, and when a k_κ is revealed, it sets $f_{k_\kappa}(m\|r) = s_\kappa$.

Consequently, no information on E or the a_i's leaks from the signatures produced by the scheme.

For the second part, assume that an attacker knows a representation of y in the base g_j. Assume also that he knows k values $P(j)$, the associated $a_{P(j)}$, and $k+1$ elements k_j. Let i_0 be the index of the last verification condition disclosed by the signer. Producing valid signatures is now equivalent to finding an α such that $g_{i_0} = g_{P(i)}^{\alpha}$, and if $\mathcal{A}$ succeeds in doing so with a non-negligible probability, then it can be used as an oracle to solve the discrete logarithm problem. □

Efficiency. This technique offers several advantages compared to concatenation:

- Signature generation requires $n - i + 1$ exponentiations, this parameter depends on the number of coercions that the system has to withstand.
- Verification requires the same number of exponentiations as the concatenated Schnorr variant.
- The size of a signature is $(n + 1)160$ bits, instead of $320n$ bits

Roughly speaking, most characteristics are improved by a factor of two, which represents a significant improvement.

5 Conclusion

We proposed new signature mechanisms that tolerate, up to a certain point, secret disclosure under constraint. More precisely, we introduced symmetric and asymmetric monotone signatures to thwart different types of attacks. The asymmetric monotone scheme offers the broadest protection for the signer. We gave a practical example of such a scheme, based on the Okamoto-Schnorr signature. The new scheme, called Okamoto-Schnorr with i degrees of freedom, is provably secure against adaptive chosen-message attacks. We believe that the proposed solution can be practically deployed at the scale of a country.

References

1. M. Bellare, J. A. Garay, T. Rabin, *Fast Batch verification for modular exponentiation and digital signatures*, Advances in Cryptology EUROCRYPT'98, Springer-Verlag, LNCS 1403, pp. 236–250, 1998.
2. M. Bellare, P. Rogaway, *Random oracles are practical: a paradigm for designing efficient protocols*, Proceedings of the 1-st ACM conference on computer and communications security, pp. 62–73, 1993.
3. M. Bellare, P. Rogaway, *The exact security of digital signatures - How to sign with RSA and Rabin*, Advances in Cryptology EUROCRYPT'96, Springer-Verlag, LNCS 1070, pp. 399–416, 1996.
4. S. Brands, *An efficient off-line electronic cash system based on the representation problem*, Technical report, CWI (Centrum voor Wiskunde en Informatica), 1993.
   Also available on-line : `http://www.cwi.nl/cwi/publications` CS-R9323.
5. T. El Gamal, *A public key cryptosystem and a signature scheme based on discrete logarithms*, IEEE Transactions on Information Theory, vol. IT–31, no. 4, pp. 469–472, 1985.
6. U. Feige, A. Fiat, A. Shamir, *Zero-knowledge proofs of identity*, Journal of Cryptology, vol. 1, no. 2, pp. 77-95, 1988.
7. S. Goldwasser, S. Micali, R. Rivest, *A Digital signature scheme secure against adaptative chosen-message attacks*, SIAM journal of computing, vol. 17, pp. 281–308, 1988.
8. A. Herzberg, S. Jarecki, H. Krawczyk, M. Yung, *Proactive secret sharing, or: how to cope with perpetual leakage*, Advances in Cryptology CRYPTO'95, Springer-Verlag, LNCS 963, pp. 339–352, 1995.
9. M. Jakobsson, M. Yung, *Revokable and versatile electronic money*, Proceedings of the 3-rd ACM conference on computer and communications security, pp. 76–87, 1996.
10. C. Li, T. Hwang, M. Lee, (t, n)*-threshold signature schemes based on discrete logarithm.* Advances in Cryptology EUROCRYPT'94, Springer-Verlag, LNCS 950, pp. 191–200, 1995.
11. D. M'raïhi, D. Naccache, S. Vaudenay, D. Raphaeli *Can D. S. A. be improved ? Complexity trade-offs with the digital signature standard*, Advances in Cryptology EUROCRYPT'94, Springer-Verlag, LNCS 950, pp. 77–85, 1995.
12. D. M'raïhi, D. Naccache, *Batch exponentiation - A fast DLP-based signature generation strategy*, 3-rd ACM conference on communications and computer security, pp. 58–61, 1996.
13. D. M'raïhi, D. Naccache, D. Pointcheval, S. Vaudenay, *Computational alternatives to random number generators*, Proceedings of the fifth annual workshop on selected areas in cryptography, LNCS 1556, pp. 72–80, 1998. Springer-Verlag.
14. NIST, *Digital Signature Standard (DSS)*, Federal Information Processing Standards Publication 186, 1994.
15. T. Okamoto, *Provably secure and practical identification schemes and corresponding signature schemes*, Advances in Cryptology CRYPTO'92, Springer-Verlag, LNCS 740, pp. 31–53, 1992.
16. D. Pointcheval, J. Stern, *Security arguments for digital signatures and blind signatures*, Journal of Cryptology, vol. 13, no. 3, pp. 361–396, 2000.
17. R. Rivest, A. Shamir, L. Adleman, *Method for obtaining digital signatures and public key cryptosystems*, Communications of the ACM, vol. 21, pp. 120–126, 1978.
18. C. Schnorr, *Efficient signature generation by smart cards*, Journal of Cryptology, vol. 4, no. 3, pp. 161-174, 1991.
19. V. Shoup, *Practical threshold signatures*, Technical report, IBM Research, June 1999. Report RZ 3121.
20. G. Simmons, *The subliminal channel and digital signatures*, Advances in Cryptology EUROCRYPT'84, Springer-Verlag, LNCS 209, pp. 364–378, 1985.
21. S. von Solms, D. Naccache, *On blind signatures and perfect crimes*, Computers & Security, vol.11, pp. 581–583,1992
22. R. L. Van Renesse, *Optical document security*, Artech House Optoelectronics Library, 2-nd edition, 1998.