Fine-Grained Forward-Secure Signature Schemes without Random Oracles

(Extended Abstract)

Jan Camenisch
IBM Zurich Research Laboratory

Maciej Koprowski*
BRICS† University of Aarhus

Abstract

We propose the concept of fine-grained forward secure signature schemes. Such signature schemes not only provide non-repudiation w.r.t. past time periods the way ordinary forward secure signature schemes do but, in addition, allow the signer to specify which signatures of the current time period remain valid in case the public key needs to be revoked. This is an important advantage if the signer processes many signatures per time period as otherwise the signer would have to re-issue those signatures (and possibly re-negotiate the respective messages) with a new key.

Apart from a formal model for fine-grained forward-secure signature schemes we present practical instantiations and prove them secure under the Strong RSA Assumption only, i.e., we do not resort to the random oracle model to prove security. As a side-result, we provide an ordinary forward-secure scheme whose key-update time is significantly smaller than for other known schemes that do not assume random oracles.

1 Introduction

Ordinary digital signature schemes suffer from a fundamental shortcoming: when the public key is revoked, e.g., because the secret key got exposed, all signature made become reputable. Therefore ordinary signature schemes can pre se not provide non-repudiation.

One possibility to overcome this problem is to use a so-called time-stamping service. Here, each signature is sent to a trusted third party who signs a message containing the signature and the current date and time. A signature is now considered non-reputable if its time-stamped predates the time of the public key's revocation. Hence non-repudiation is guaranteed, assuming that the secret key of the time-stamping service is never leaked. However, such a time-stamping service is likely to be a bottle-neck, in particular as running such a service is a difficult task similar to the one of running a certification authority. Hence this solution is not very desirable.

Another solution to the problem is to change the keys frequently, e.g., every day, and to delete past secret keys. It then is understood that a signature is valid if the user did not revoke the corresponding public key during the *same* time period she used that key. In practice, however, the user must be allowed to revoke the key also for some while after the period has passed. This is because some time must be allowed to the user to discover key

*Part of the work done while visiting IBM Zurich Research Laboratory.
†Basic Research in Computer Science, Centre of the Danish National Research Foundation.

leakage and for a revocation request to be processed by the certification authority. Note that as long as the time period plus the extra time allowed has not passed one cannot be sure about a signature's non-repudiation because a (dishonest) signer can always revoke the key to "recall" a signature. This, however, seems to be unavoidable.

Unfortunately, this above solution either requires frequent interaction with the certification authority to register the public keys or, otherwise, the public key becomes large (e.g., consists of a list of daily public keys). This drawback is overcome by forward secure signature schemes as introduced by Anderson [2] (and formalized by Bellare and Miner [3]). These schemes allow the users to have a different secret key for each time period but only a single (small) public key [3, 1, 12, 15, 14]. In fact, most forward secure signature schemes allow one to derive the secret key for the current period from the one of the previous period in a one-way fashion.

Still, forward secure signature schemes are not completely satisfactory as at least all the signatures made in the current time period become invalid when the user revokes the public key. Hence the signer needs to re-issue those signatures (and possibly re-negotiate the corresponding messages) with a new key.

One way to solve this problem is to update the secret key very frequently, e.g., every second. This is of course only feasible if the secret-key update is very efficient. This is the case for the scheme by Kozlov and Reyzin [14] and the scheme obtained from the MMM construction [15] applied to the Schnorr signature scheme [18]. Unfortunately, they both are not provably secure in the strict sense, i.e., they are only proven secure in the random oracles model, which is a heuristic only (in fact, it has been proven that such oracles cannot be realized with any hash function [4, 17]). For the known schemes that are provably secure in the strict sense, the time to update the secret key is too large for this approach to work.

We therefore propose *fine-grained forward-secure signature schemes* as a better solution. The idea here is that, apart from using a new secret key for each time-period, all the signatures carry an ascending index such that once an index is used (and the secret key updated accordingly), no signature can be made w.r.t. a lower index. Thus, whenever the secret key gets compromised, the signer can, together with revoking the public key, just announce the index used before the secret key got compromised.

Apart from putting forth a formal model of fine-grained forward-secure signature schemes, we also present practical schemes that are provably secure under the Strong RSA Assumption. Our scheme also yields an ordinary forward-secure signature scheme whose secret-key update time is about a factor $\Omega(\log t)$ faster than compared to the best known schemes that are provably secure in the strict sense, where t is the number of time periods, while the time to sign a message is essentially the same.

Recently, key-insulated signature schemes [9] and intrusion-resilient signatures [13] have been introduced. In these schemes not only past but also future secret keys are protected in case the current secret key is compromised. Unfortunately those solutions require signer's communication with a safe computing device and assume that both this device and the signer's system cannot be broken into during the same time period. We stress that our fine-grained protocols do not require existence of such additional devices. Nevertheless our fine-grained techniques could be combined with key-insulated and intrusion-resilient signatures to provide them with supplementary fine-grained forward security within time periods.

2 Model

Here we formally define the notion of fine-grained forward secure signature scheme. Our model resembles the definitions of forward secure signature scheme [3]. The main difference is that instead of only considering time periods we also take into account the ordering of signatures within a time period.

First we describe features of fine-grained key-evolving signature scheme. Afterwards we define its forward security.

Definition 1. *Let k be a security parameter, I the maximal amount of signatures that can be made per time period, and T the maximal amount of time-periods.*

A fine-grained key-evolving signature scheme *consists of the following four procedures.*

$\mathsf{KG}(k, I, T)$. *The key generation algorithm takes as input k, I, and T, and outputs a public key PK and an initial secret key $SK_{(0,0)}$.*

$\mathsf{Upd}(SK_{(t,i)})$. *The (time-period) update algorithm takes as input the current secret key $SK_{(t,i)}$ and outputs the secret key $SK_{(t+1,0)}$ for the next time-period.*

$\mathsf{Sig}(m, SK_{(t,i)})$. *On input a message $m \in \{0,1\}^*$ and the current secret key $SK_{(t,i)}$, this algorithm outputs either a signature (t, i, σ) and a new secret key $SK_{(t,i+1)}$, or the symbol $\perp$.*

$\mathsf{Ver}(PK, m', (t', i', \sigma'))$. *On input a public key PK, a message m', and a candidate signature (t', i', σ'), the verification algorithm outputs either 1 or 0.*

A useful signing algorithm outputs $\perp$ only if all T time-periods have passed, or if the maximum amount of I signatures (total or per time-period) were already made. Note that each signature carries a unique index (t, i).

Definition 2 (Validity of Signatures). *A signature (t, i, σ) on a message m is considered valid if $\mathsf{Ver}(PK, m, (t, i, \sigma)) = 1$ and t is not a future time period. However, if the signer revoked the public key w.r.t. the signature index i' in time-period t', then additionally either $t < t'$ or $i < i' \wedge t = t'$ must hold.*

Let Δ_R is the extra time allowed to the signer and certification authority to process a revocation. Note that the validity of a signature is definitely determined only when the time period t plus some extra margin Δ_R has passed.

We extend the notion of adaptively chosen message attacks [11] to fine-grained forward secure schemes.

Definition 3 (Security). *A fine-grained key-evolving signature scheme $(\mathsf{KG}, \mathsf{Upd}, \mathsf{Sig}, \mathsf{Ver})$ provides fine-grained forward security if there exists no polynomial time adversary who can adaptively ask for messages to be signed, then learns a key $SK_{(t_A, i_A)}$, and outputs a valid signature (t', i', σ') on a new message m' either with $t' < t_A$ or $t' = t_A \wedge i' < i_A$ with nonnegligible probability.*

It should now be clear that Definition 3 allows a signer to maintain the validity of *all* the signatures made up to the point when the secret key gets compromised, and not only those made in prior time periods.

Furthermore, from the way validity of signatures is defined, it follows that if we assume for instance a time-stamping service for revocation, a dishonest signer can recall only the signatures made in the current time period (and in the previous one if Δ_R has not yet passed). If we do not make such an assumption, there is no way a (dishonest) signer can be prevented from revoking all the signatures issued. In case a honest signer has noticed a break-in after the i'th message was signed, he has the possibility to announce the index i (together with time period t) which protects the signatures issued prior to that break-in even during the current time period.

We note that we allow the signer to revoke a key several times as otherwise an adversary who knows the secret key could send a revocation message with a index that is higher than the signer's current index.

3 Fine-Grained Forward-Secure Signature Schemes With A Single Time Period

In this section we provide fine-grained forward secure schemes which allows for one time period only; thus a dishonest signer could recall all the signatures he made with it. These schemes, however, are not our final ones but rather the basis for the schemes we construct in the following sections.

In principle, a fine-grained forward secure signature scheme with a single time period can be obtained from any ordinary forward-secure signature scheme S by abandoning the fixed time periods. More precisely, to sign the i-th message m, one uses the i-th time-period's secret key of S, and then updates the secret-key of S.

In fact, using the scheme obtained by applying the MMM construction [15] to the Schnorr signature scheme [18] in this way gives a pretty efficient fine-grained forward-secure signature scheme with a single time-period. However, it is provably secure only in the random oracle model. Unfortunately, applying the MMM construction to any of the known signature scheme that are provably secure in the strict sense yields no efficient fine-grained forward-secure signature scheme. This is because we now sign only one message per time period and thus signing includes running the key-generation algorithm of the underlying signature scheme that for these schemes consists of the costly generation of an RSA modulus. One could also use MMM construction with a one-time signature scheme, which would be more efficient but result in longer signatures. This section is therefore dedicated to efficient fine-grained forward-secure signature scheme with a single time period that are provable secure solely under the strong RSA assumption.

3.1 Basic Scheme With A Single Time Period

As mentioned, our construction is based on the strong RSA assumption and forward security is achieved by erasing roots. Similar techniques were used in previous forward-secure signature schemes [3, 1, 12]. However, different from those schemes, we are base ours on the provably secure standard signature scheme due to Fischlin [10].

Let k and ℓ be security parameters. For current security requirements we suggest $k = 2000$ and $\ell = 160$. With respect to efficiency, we suggest $I = 1024$. Let $\mathcal{H}(\cdot)$ be a collision-resistant hash function whose output is of size ℓ bits, i.e., for any s we have $0 \leq \mathcal{H}(s) < 2^{\ell}$. Finally, let $\mathbf{QR}_N$ denote the subgroup of squares in $\mathbb{Z}_N^*$.

The scheme. We new describe the algorithm of our scheme. Note that because we only allow for a single time period, there is no $\mathsf{Upd}(\cdot)$ algorithms to update the secret key to the one of the next time period.

$\mathsf{KG}(k, I)$:

1. Choose two random safe primes of size about $k/2$ bits and let N be their product.
2. Choose a random seed W and use it with some pseudo-random generator to construct I random unique $\ell + 1$-bit prime numbers $e_0, \ldots, e_{I-1}$.
3. Draw random elements $\hat{g}_0, \hat{h}_0, \hat{x}_0 \in_R \mathbf{QR}_N$ and compute $e := \prod_{0 \le i < I} e_i \mod \varphi(N)$, $g := \hat{g}_0^e$, $h := \hat{h}_0^e$, and $x := \hat{x}_0^e$.
4. Output $PK := (N, g, h, x, W, I)$ and $SK_0 := (0, \hat{g}_0, \hat{h}_0, \hat{x}_0)$.

$\mathsf{Sig}(m, (i, \hat{g}_i, \hat{h}_i, \hat{x}_i))$:

1. If $i = I$ then output $\perp$ and quit.
2. Compute elements g_i, h_i, and x_i such that
$$g_i^{e_i} = g \ , \qquad h_i^{e_i} = h \ , \qquad \text{and} \qquad x_i^{e_i} = x \ .$$
3. Choose a random element $\alpha \in_R \{0,1\}^\ell$ and compute
$$y := x_i g_i^\alpha h_i^{\alpha \oplus \mathcal{H}(m)} \ .$$
4. Update the secret key: compute
$$\hat{g}_{i+1} := \hat{g}_i^{e_i} \ , \qquad \hat{h}_{i+1} := \hat{h}_i^{e_i} \ , \qquad \text{and} \qquad \hat{x}_{i+1} := \hat{x}_i^{e_i}$$
and set $SK_{i+1} := (i+1, \hat{g}_{i+1}, \hat{h}_{i+1}, \hat{x}_{i+1})$.
5. Output the signature $(i, (y, \alpha))$ and the new secret key SK_{i+1}.

Remark on Step 2. The value g_i can be computed as $g_i := \hat{g}_i^{\prod_{i<j<I} e_j}$ from $\hat{g}_i$ as we have $g = \hat{g}_i^{\prod_{i \le j < I} e_j}$ by construction. Analogous statements hold for h_i and x_i. However, this would result in $O(I\ell k^2)$ bit operations to produce a signature which might be far too expensive for many applications. A more efficient way to compute these values is to use the pebbling method by Itkis and Reyzin [12]. It allows one to compute the e_i-th root of g using at most $O(\log(I-i))$ single exponentiations with an $(l+1)$-bit exponent. The price for this efficiency improvement are $O(k \log(I-i))$ additional bits in the secret key.

$\mathsf{Ver}(PK, m, (i, (y, \alpha)))$:

1. If $i \notin [0, I)$ output 0.
2. Generate the prime exponent e_i from the seed W.
3. Output 1 if
$$y^{e_i} = x g^\alpha h^{\alpha \oplus \mathcal{H}(m)}$$
holds and 0 otherwise.

Security proof. The above algorithms constitute a fine-grained forward-secure signature scheme in terms of Definition 3 if the Strong RSA Assumption holds and the hash function $\mathcal{H}(\cdot)$ is collision-resistant.

We sketch here our security proof. Much of this analysis is similar to the one given by Cramer and Shoup [7] and by Fischlin [10] for their respective signature schemes.

Recall the strong RSA assumption: given a random RSA modulus N and a random element $x \bmod N$, it is difficult to solve the flexible RSA problem, i.e., to find an element $y \bmod N$ and integer $e > 1$ such that $y^e = x$

We are given an instance (N, z), with $z \in \mathbb{Z}_N^*$, of the flexible RSA problem and we will simulate signing for the forger who is allowed to adaptively query signatures on messages until some index i_A, $0 \leq i_A \leq I$, when he asks to see SK_{i_A} and finally outputs a signature $(\tilde{\jmath}, (y, \alpha))$ on a message m with index $\tilde{\jmath} < i_A$ that was not signed w.r.t. index $\tilde{\jmath}$.

We guess j and furthermore whether $\alpha_j \neq \alpha$ or $\alpha_j \oplus \mathcal{H}(m_j) \neq \alpha \oplus \mathcal{H}(m)$ will be the case (one of them must hold since $\mathcal{H}()$ is collision-resistant and $m \neq m_j$). Let's us discuss the former case $\alpha_j \neq \alpha$. The latter one can be dealt with in a similar way.

We choose I random unique $(l+1)$-bit primes $e_0, \ldots, e_{I-1}$, random $v, w \in_R \mathbb{Z}_n^*$, and $\beta \in_R \{0,1\}^l$ and compute

$$g := z^{2\prod_{i=0; i\neq j}^{I-1} e_i} \quad , \qquad h := v^{2\prod_{i=0}^{I-1} e_i} \quad , \qquad \text{and} \qquad x := w^{2\prod_{i=0}^{I-1} e_i} g^{-\beta} \quad .$$

With this setup we are able produce (simulate) signatures on I messages using random α_i's for $i \neq j$ and $\alpha_j = \beta$. Then, if the forged signature satisfies $j < i_A$ and $j = \tilde{j}$, by combining it with the signature $(\jmath, (y_j, \alpha_j))$ on message m_j, we obtain the equation

$$z^{(\alpha-\alpha_j)2\prod_{s=0; s\neq j}^{I-1} e_s} = \left(v^{((\alpha_j \oplus \mathcal{H}(m_j)) - (\alpha \oplus \mathcal{H}(m)))2\prod_{s=0; s\neq j}^{I-1}(e_s)} y/y_j\right)^{e_j} \quad .$$

It can be shown that $e_j \nmid (\alpha - \alpha_j) 2 \prod_{s=0; s\neq j}^{I-1} e_s$ and hence one can compute the e_j-th root of z from the above equation using the extended Euclidean algorithm.

3.2 A One-Period Flat-Tree Scheme

Based on our basic scheme, we now build a more efficient fine-grained forward-secure signature scheme with only one period. We achieve a much faster key generation and we can avoid regenerating primes because here we need less primes, so that it may become affordable to retain them in memory. Still, this scheme is not the final solution but just a building block for protocols presented in the following sections.

We recall the idea of flat authentication tree (see [5, 6, 8]). Let us construct a tree of degree I_0 and depth $d-1$. Each node of the tree is going to be associated with one instance of our basic scheme with the maximal number of signatures equal I_0. For all these instances we use the same RSA modulus N and the same primes $e_0, e_1, \ldots, e_{I_0-1}$ (but different x, g, and h). For each internal node, the associated scheme is used to authenticate the public keys of the schemes of its children. Finally, each leaf's scheme allows us to sign I_0 messages and thus we can sign $I = I_0^d$ messages.

This scheme can be seen as an iterated product from [15] applied to our basic scheme. Hence we can deduce that it is a fine-grained forward-secure signature with one period.

We note that there is no need to store information about the schemes of all nodes in the tree. It is sufficient to keep in memory the information related to the signature schemes associated with nodes on the path from the root to the actual leaf of the authentication tree.

The instances of our basic scheme associated with the leaf nodes should us the pebbling-method to improve performance of signing. However, this is not necessary for the internal nodes of the authentication tree as there computing roots can be distributed to the signing with the schemes associated with the children of the node such that the signing cost increases only insignificantly.

4 An Ordinary Forward-Secure Signature Scheme

In this section we construct a new ordinary forward-secure signature scheme from the fine-grained basic scheme with a single time period presented in §3. Our new scheme has a much faster secret-key update time than previously known ones that are secure without assuming random oracles while the signing-time is about the same.

Given any fine-grained forward-secure signature F scheme for a single time period (such as the ones put forth in the previous section) and some ordinary signature scheme S, one can build an ordinary forward-secure signature scheme F' as follows. The public key of F' is generated by running the key generation of F. Furthermore, each signature index i of F is assigned to the i-th time-period of F'. When time period t starts, the key generation algorithm of S is run and the thereby-obtained public key P_t of S is signed with F using signature index t. Let us denote this signature σ_t. Now, the signature on message m that is produced in time period t consists of P_t, σ_t and the signature on m produced with S using the secret key corresponding to P_t. This construction we just detailed can be seen as the product composition (defined in [15]) of F and S.

Now, let F be the basic scheme provided in §3 (with using pebbling), let S be the Cramer-Shoup signature scheme [7]. We compare the efficiency of the so obtained F' with the efficiency of the scheme obtained by applying the MMM construction to the Cramer-Shoup signature scheme, which was hitherto the most efficient forward-secure signature scheme provably secure not assuming random oracles. It is not hard to see that signing time for both schemes is the same, i.e., essentially the time of signing a message with the Cramer-Shoup signature scheme. The secret-key update time of our scheme is dominated by one generation of Cramer-Shoup keys while in case of MMM scheme it is $\Omega(\log t)$ key generations of S on average, where t is the number of time periods used so far.

Under usually satisfiable conditions on the maximal number of time periods T, our key generation is also faster. More precisely, for our scheme it includes one key generation of S and preparing the keys for F. In the MMM's scheme the main effort in key generation is to initialize l signature schemes S. As long as $Tl^3/\ln l < \frac{k^5}{16\ln^2(k/2)}$ and $T < \frac{k^4}{16\ln^2(k/2)}$ our construction is going to be faster than MMM with respect to key generation. For example, if $k = 2000$, $l = 160$, and $T = 1024$ then key generation for MMM would require approximately 2^{52} bit operations as opposed to approximately 2^{45} bit operations for our scheme.

Secret key size is $k(3\log_2 T + 8) + \log_2 T(4\log_2 T + 5) + 2l + 2$ bits in our protocol. A simple straightforward implementation of MMM needs $O(k + l(\log t + \log l))$ bits of secret storage. If we require a fast worst-case update time that results (by our analysis and also [16]) in $O(k\log^2 t)$ bits of secret storage.

5 Fine-Grained Forward Secure Signature Schemes

We now discuss ways to build full fledged fine-grained forward-secure signature schemes from any fine-grained forward-secure signature schemes with at least a single time period, e.g., the schemes presented in §3. If the underlying scheme is provably secure without assuming random oracles then so is the resulting scheme.

The advantage of having several time periods is that once the time-period (plus some additional courtesy time Δ_R) has passed, the signer can no longer invalidate the signatures made in this time period by revoking her public key. That is, if we had only one time period, then the signer could at any time invalidate all the signature made.

We remind the reader that the validity of a signature can only be finally determined if the time-period in which it was produced plus the courtesy time Δ_R has passed. This, however, it the case for any signature scheme.

5.1 A Two-Level Scheme

We use one instance of one-period flat-tree scheme (or any other one-period fine-grained scheme), call it A-scheme, where each index denotes a time-period, i.e., index i denotes the time period T_i from $t_0 + i \cdot t_\Delta$ to $t_0 + (i+1)t_\Delta$, where t_0 is the starting time and t_Δ is the duration of the time periods. The public key of this scheme becomes the public key of the overall scheme. Furthermore, a parameter Δ_R is published that controls the time allowed to the user to realize that the secret key got compromised and to react upon this.

Then, for each time-period T_i we use a second instance of our basic scheme (call it B_i-scheme) and sign its public key using the A-scheme w.r.t. the index i of that time-period. After this, the secret key of the A-scheme is updated and its current index is set to $i+1$.

To sign a the j-th message of the current time period T_i, we use the B_i-scheme with index j. The signature on the message consist of this signature, the public key of the B_i-scheme, and the signature on this public key made with the A-scheme. Again, after signing the secret key of the B_i scheme is updated and new current index is $j+1$.

Now, whenever a signer wants to revoke her key w.r.t. index j' and time-period $T_{i'}$, she sends the TTP a message indicating this. The TTP checks whether $T_{i'}$ denotes the current time period or whether less time than Δ_R has passed since the period $T_{i'}$ ended. If this is the case the TTP accepts the revocation and publishes j' and $T_{i'}$ appropriately.

A user's signature with indices i and j is considered valid if no revocation happened, or if a revocation w.r.t. indices i' and j' happened (where i' and j' are the smallest indices published by the TTP), if $i < i'$ or $i = i'$ and $j < j'$ holds.

5.2 Using a Public Archive

The idea here is to replace the A-scheme in the previous solution with a public archive. We assume that it's not possible to delete messages from the archive and that messages are published together with the exact time they were received by the archive.

Given such an archive, one can realize a fine-grained forward-secure signature scheme using only one instance of our one-period flat-tree scheme as follows (one could use our basic scheme or any other one-period fine-grained forward secure signature).

A signature on m is made with our one-period flat-tree scheme using the current index. After having signed, the secret key is updated.

At the end of each time period, the user signs a pre-determined message (e.g., "last index used in time period T_i") with the one-period scheme using the current index, say j, and then updates the secret key and sends this *index signature* to the public archive. The public archive posts the message along with the time it received the signature.

Now, whenever a signer wants to revoke her key w.r.t. index j' and time-period $T_{i'}$, she sends the TTP a message indicating this. The TTP checks whether either $T_{i'}$ is the current time period or less than Δ_R time has passed since $T_{i'}$ has ended and whether j' is not smaller than the index j of the index signature the signer put in the public archive for the time period $T_{i'-1}$. If all these checks succeed, the TTP publishes j' and $T_{i'}$.

In this solution, a user's signature with index j is considered valid if no revocation happened, or if revocation happened, if $j < j'$, where j' is the smallest index of any revocation signatures published by the TTP.

Compared to the previous solution, this one has the advantage that we can have an almost arbitrary number of signatures per time period.

5.3 Allowing s Signatures Per Time-Period

In this solution a single instance of a one-period fine-grained signature scheme is sufficient as well. The idea is to bind the signature indices to time-periods by allowing exactly s signatures per time-period. The parameter s (together with t_0, t_Δ) need to be published as part of the public key. Thus in time-period T_i, the indices $is, (i+1)s-1$ can be used to sign.

To revoke her public key w.r.t. index j' the user sends a message indicating this to the TTP. The TTP publishes the index j' if it matches the current time-period or if it matches the prior time-period and less time that Δ_R has passed since the end of it. A signature with index j is considered valid if no revocation happened or, in case a revocation w.r.t. index j' was done, if $j < j'$ holds.

The rational behind this proposal is the main work of signing a message consists actually of the secret-key update. Thus one could calculate how many signatures one can possibly issue during a time period given the computational power one has and then set s to this number. Then, one would constantly perform the secret key update, even if no message was signed. This approach would not change the response behavior of the system very much, but does not require a public archive and the signatures are smaller than in the first solution.

References

[1] M. Abdalla and L. Reyzin. A new forward-secure digital signature scheme. In *ASIACRYPT 2000*, vol. 1976 of *LNCS*, pages 116–129. Springer-Verlag, 2000.

[2] R. J. Anderson. Invited lecture. ACM CCS, 1997.

[3] M. Bellare and S. K. Miner. A forward-secure digitial signature scheme. In *CRYPTO '99*, vol. 1666 of *LNCS*, pp. 431–448. Springer Verlag, 1999.

[4] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. In *13th ACM STOC*, pp. 209–218, 1998.

[5] R. Cramer and I. Damgård. Secure signature schemes based on interactive protocols. In *CRYPTO '95*, vol. 963 of *LNCS*, pp. 297–310, 1995.

[6] R. Cramer and I. Damgård. New generation of secure and practical RSA-based signatures. In *CRYPTO '96*, vol. 1109 of *LNCS*, pp. 173–185. Springer Verlag, 1996.

[7] R. Cramer and V. Shoup. Signature schemes based on the strong RSA assumption. In *ACM CCS*, pp. 46–51, 1999.

[8] I. Damgård and M. Koprowski. Generic lower bounds for root extraction and signature schemes in general groups. In *EUROCRYPT 2002*, vol. 2332 of *LNCS*, pp. 256–271.

[9] Y. Dodis, J. Katz, S. Xu, and M. Yung. Strong key-insulated signature schemes. In *PKC 2003*, vol. 2567 of *LNCS*, 2003.

[10] M. Fischlin. The Cramer-Shoup Strong-RSA signature scheme revisited. In *PKC 2003*, vol. 2567 of *LNCS*, 2003.

[11] S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.

[12] G. Itkis and L. Reyzin. Forward-secure signatures with optimal signing and verifying. In *CRYPTO 2001*, vol. 2139 of *LNCS*, pp. 332–354. Springer Verlag, 2001.

[13] G. Itkis and L. Reyzin. SiBIR: Signer-base intrusion-resilient signatures. In *CRYPTO 2002*, 2002.

[14] A. Kozlov and L. Reyzin. Forward-secure signatures with fast key update. In *SCN*, vol. 2576 of *LNCS*, 2002.

[15] T. Malkin, D. Micciancio, and S. Miner. Efficient forward-secure signatures with an unbounded number of time periods. In *EUROCRYPT 2002*, vol. 2332 of *LNCS*, pp. 400–417. Springer Verlag, 2002.

[16] D. Micciancio. Private communication, August 2002.

[17] J. B. Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In *CRYPTO 2002*, 2002.

[18] C. P. Schnorr. Efficient signature generation for smart cards. *Journal of Cryptology*, 4(3):239–252, 1991.