A New Key-Insulated Signature Scheme

Nicolás González-Deleito, Olivier Markowitch, and Emmanuel Dall'Olio

Université Libre de Bruxelles
Bd. du Triomphe – CP212
1050 Bruxelles
Belgium
{ngonzale,omarkow,edalloli}@ulb.ac.be

Abstract In this paper we propose a new strong and perfectly key-insulated signature scheme, more efficient than previous proposals and whose key length is constant and independent of the number of insulated time periods. Moreover, unlike previous schemes, it becomes forward-secure when all the existing secrets at a given time period are compromised. We also present a variant forward-secure scheme in which an adversary needs to compromise a user at a second time period before being able to compute future secret keys.

Keywords: key-insulation, forward-security, signature schemes

1 Introduction

Classical digital signature schemes make use of a secret signing key only held by a signer and of a related public key allowing to verify the correctness of a signer's signature on a given document. Compromise of a secret key lets an attacker produce signatures on behalf of the corresponding signer and forces the latter to revoke its public key and generate a new pair of keys. Therefore, before accepting a signature as valid, one also has to verify that the public key has not been revoked. In that case, such a verification prevents an attacker to convince a third entity that a forged signature is correct. Unfortunately, this does not always suffice as even valid signatures having been produced before the compromise become invalid, unless a time-stamping authority has attested that they were produced before the corresponding public key was revoked.

Getting rid of the revocation and time-stamping mechanisms in order to simplify key management is an active research topic. It is particularly important for online non-repudiation services, where the validity of digital signatures has to be guaranteed during a long period of time in order to resolve possible disputes.

A first approach to this problem merely aims to complicate the task of an adversary by sharing the knowledge about a secret key among a set of entities. In threshold signature schemes [5], when signing a document each entity belonging to a predetermined subset of the entities sharing the corresponding secret key computes a partial signature on that document with the help of its share. Those partial signatures are combined into one final and verifiable signature which is given to the verifier. An adversary has therefore to compromise a threshold number of entities in order to forge signatures. Proactive schemes [10] go a step further by forcing entities to periodically refresh their secret share, but without modifying the value of the secret key. These updates are done in such a way

that an adversary has to compromise a threshold number of entities before the next refresh in order to be able to produce signatures for a given secret key.

More practical approaches try to limit the damages arising when secret keys are exposed. In the forward-secure model [2, 4], secret keys are not shared among some servers, but their lifetime is divided into discrete time periods. At the beginning of each period, users compute a new secret key by applying a public one-way function to the secret key used during the previous time period, while public keys remain unchanged. An adversary compromising the secret key at a given time period will be unable to produce signatures for previous periods, but will still be able to sign messages during the current and future time periods. Unlike classical schemes, the validity of previously produced signatures is therefore assured[1], but public keys have to be revoked.

The notion of key-insulated cryptosystems, which was introduced by Dodis et al. [6], generalises the concept of forward-secure cryptography. In this model, lifetime of secret keys is also divided into discrete periods and, as in previous models, signatures are supposed to be generated by relatively insecure devices. However, the secret associated with a public key is here shared between the user and a physically secure device. At the beginning of each time period the user obtains from the device a partial secret key for the current time period. By combining this partial secret key with the secret key for the previous period, the user derives the secret key for the current time period. Exposure of the secret key at a given period will not enable an adversary to derive secret keys for the remaining time periods. More precisely, in a (t, N)-key-insulated scheme the compromise of the secret key for up to t time periods does not expose the secret key for any of the remaining $N - t$ time periods. Therefore, public keys do not need to be revoked unless t periods have been exposed.

Additionally, *strong* key-insulated schemes guarantee that the physically secure device (or an attacker compromising the partial secrets held by this device) is unable to derive the secret key for any time period. This is an extremely important property if the physically secure device serves several different users.

As Dodis et al. already noted [7], key-insulated signatures schemes can be used for signature delegation. In this context, a user grants to another user the right to sign messages on his behalf during a limited amount of time. This kind of delegation can be simply achieved by giving to this second user a secret key for the corresponding time period. Indeed, the user who receives the signing power will be unable to derive a secret key for another time period and, by using a time-stamping service, will only produce valid signatures during the delegated time period.

Finally, Itkis and Reyzin [12] introduced the notion of intrusion-resilient signatures, which strengthens the one of key-insulation by allowing an arbitrary number of non-simultaneous compromises of both the user and the device, while preserving security of prior and future time periods. However, as pointed out by Zhou et al. [16], a simple loss of synchronisation between the user and the device

[1] Time-stamping services can still be useful in some contexts to avoid backdating of documents.

is not recoverable. This forces the corresponding public key to be revoked, which seems to be to us in contradiction with the goals of key-insulation.

1.1 Contributions

In their first paper [6], Dodis et al. focused exclusively on public-key encryption schemes. Recently, they proposed [7] three different key-insulated digital signature schemes. The first one is a generic and strong $(N-1,\ N)$-key-insulated scheme with public and secret keys of constant length, but whose signatures are composed of two signatures from the scheme upon which the key-insulated scheme is built. In their second construction, which is based on the discrete logarithm assumption, the length of the public key and of the device's secret key depends linearly on the number of insulated time periods. Finally, they describe a generic and strong $(N-1,\ N)$-key-insulated scheme that can be efficiently instantiated by additively sharing a secret RSA exponent between the user and the physically secure device.

We propose in this paper a new strong $(N-1,\ N)$-key-insulated signature scheme. Before presenting this system, we describe a variant scheme (also using a physically secure device) which is forward-secure. Its main advantage with respect to forward-secure schemes is that an adversary will need to compromise a user at a second time period before being able to compute future secret keys. This offers better protection against adversaries at a very acceptable price (the physically secure device can be implemented and deployed by means of a smart card).

The new key-insulated scheme appears to be more efficient than previous proposals. Its key length is constant and independent of the number of insulated time periods. Updates mainly consist in performing modular squarings. As for [13], this confers to our scheme very fast refreshes and enables therefore more frequent key updates. Finally, signing and verifying, which are based on the Guillou-Quisquater signature scheme [9], remain also very efficient.

Unlike the schemes proposed by Dodis et al. [7], our key-insulated scheme becomes forward-secure when all the existing secrets at a given time period are compromised. In that sense, it respects one of the most interesting properties of intrusion-resilient schemes [12], without suffering from the synchronisation drawback stated above.

The remaining of this paper is organised as follows. Section 2 is devoted to formally define key-insulated signature schemes. The scheme respecting the forward-secure property is presented in section 3. By doing so, we will be able to better describe in section 4 the new key-insulated scheme. This section includes a proof of security in the random oracle model under the strong RSA assumption, that can be easily adapted to the first scheme. Finally, section 5 provides a detailed performance comparison between existing key-insulated signature schemes.

2 Definitions

In this section we formally define key-insulated signature schemes and the properties that those schemes may respect. The following definition of key-updating

signature schemes is based on the definition given by Dodis et al. [7] and tries to be more generic than theirs.

Definition 1 A *key-updating signature scheme* is a 5-tuple of polynomial time algorithms (KGen, UpdD, UpdU, Sig, Ver) such that:

- KGen, the key generation algorithm, is a probabilistic algorithm taking as input one or several security parameters sp and (possibly) the total number of periods N, and returning a public key PK, a master secret key MSK and a user's initial secret key USK_0.
- UpdD, the device key-update algorithm, is a (possibly) probabilistic algorithm which takes as input the index i of the next time period, the master secret key MSK and (possibly) the total number of periods N, and returns a partial secret key PSK_i for the i-th time period.
- UpdU, the user key-update algorithm, is a deterministic algorithm which takes as input the index i of the next time period, the user's secret key USK_{i-1} for the current time period and the partial secret key PSK_i. It returns the user's secret key USK_i and the secret key SK_i for the next time period[2].
- Sig, the signing algorithm, is a probabilistic algorithm which takes as input the index i of the current time period, a message M and the secret key SK_i for the time period i; it returns a pair $\langle i, s \rangle$ composed of the time period i and a signature s.
- Ver, the verification algorithm, is a deterministic algorithm which takes as input a message M, a candidate signature $\langle i, s \rangle$ on M, the public key PK and (possibly) the total number of periods N; it returns true if $\langle i, s \rangle$ is a valid signature on M for period i, and false otherwise.

  Moreover, the following property has to be respected:

$$\mathsf{Ver}_{PK}(M, \mathsf{Sig}_{SK_i}(i, M)) = \mathsf{true} \quad \forall\, i, M, (PK, SK_i).$$

□

The life cycle of keys in a key-updating scheme can be described as follows. A user begins by running the KGen algorithm, obtaining a public key PK, as well as the corresponding master secret key MSK and user's initial secret key USK_0. The public key PK is certified through a certification authority (CA) and made publicly available, while MSK is stored on the physically secure device and USK_0 is stored by the user himself. For each time period i, $1 \leq i \leq N$, the user is now able to obtain a partial secret key PSK_i by asking the device to run the UpdD algorithm. By executing UpdU, the user transforms, with the help of USK_{i-1}, the partial secret key received from the device into a secret key SK_i for time period i which may be used to sign messages during this time period. Furthermore, the user updates USK_{i-1} to USK_i and erases USK_{i-1} and SK_{i-1}.

As Dodis et al. [6, 7], we also assume that users authenticate themselves to the physically secure devices during key updates and that the keys used for

[2] In the schemes proposed in [7], the secret key and the user's secret key are the same key.

achieving authentication are not stored on the insecure devices used for signing documents.

The above description corresponds to the normal scenario where updates are performed sequentially, from time period i to time period $i+1$. Dodis et al. [6, 7] allow updates to be done randomly, i.e. from a given time period to any other period. However, although being an interesting feature from a theoretical point of view, we prefer to discourage users from backdating documents and only support sequential updates in our schemes.

In the same way as Dodis et al. [7], we suppose that an adversary may

- ask for signatures on adaptively chosen messages for adaptively chosen time periods;
- either expose the insecure signing device for up to t adaptively chosen time periods or expose once the physically secure device;
- compromise the insecure signing device during an update.

He succeeds if he forges a valid signature $\langle i, s\rangle$ on a message M for which he never requested a signature for time period i and if he never exposed the insecure device at this time period.

We model a signature request by giving the adversary access to a signing oracle $\mathtt{Sig}_{MSK,USK_0}(\cdot,\cdot)$, that on input $(i,\, M)$ returns the result of $\mathsf{Sig}_{SK_i}(i, M)$. Key exposures are modelled by a key exposure oracle $\mathtt{Exp}_{MSK,USK_0}(\cdot)$, which on input i returns the values USK_i and SK_i stored on the insecure device during the i-th time period.

Definition 2 [7] Let $\Pi = (\mathsf{KGen}, \mathsf{UpdD}, \mathsf{UpdU}, \mathsf{Sig}, \mathsf{Ver})$ be a key-updating signature scheme. The success probability $\mathsf{Succ}_{A,\Pi}(sp)$ of an adversary A is defined as follows:

$$P\left[\mathsf{Ver}_{PK}(M,\langle i,s\rangle)=\mathsf{true}\;\middle|\;\begin{array}{c}(PK,MSK,USK_0)\leftarrow\mathsf{KGen}(sp,N),\\ (M,\langle i,s\rangle)\leftarrow A^{\mathtt{Sig}_{MSK,USK_0}(\cdot,\cdot),\mathtt{Exp}_{MSK,USK_0}(\cdot)}(PK)\end{array}\right],$$

where $(i,\, M)$ was never submitted to the signing oracle and i was never submitted to the key exposure oracle.

Π is said to be (t, N)-*key-insulated* if for any probabilistic polynomial time adversary A submitting at most t key exposure requests, $\mathsf{Succ}_{A,\Pi}(sp)$ is negligible. When $t = N - 1$, Π is said to be *perfectly key-insulated*. □

It is also possible for an adversary to compromise the physically secure device or to have a dishonest physically secure device forging signatures on behalf of the user. The following definition deals with this problem. The adversary does not query the key exposure oracle here, but the master secret key is simply given to him.

Definition 3 [7] Let $\Pi = (\mathsf{KGen}, \mathsf{UpdD}, \mathsf{UpdU}, \mathsf{Sig}, \mathsf{Ver})$ be a $(t,\, N)$-key-insulated signature scheme. The success probability $\mathsf{Succ}_{B,\Pi}(sp)$ of an adversary B is defined as follows:

$$P\left[\mathsf{Ver}_{PK}(M,\langle i,s\rangle)=\mathsf{true}\;\middle|\;\begin{array}{c}(PK,MSK,USK_0)\leftarrow\mathsf{KGen}(sp,N),\\ (M,\langle i,s\rangle)\leftarrow B^{\mathtt{Sig}_{MSK,USK_0}(\cdot,\cdot)}(PK,MSK)\end{array}\right],$$

where $(i,\ M)$ was never submitted to the signing oracle.

Π is said to be *strong* $(t,\ N)$-key-insulated if for any probabilistic polynomial time adversary B, $\mathsf{Succ}_{B,\Pi}(sp)$ is negligible. □

Finally, we address exposures of the insecure device during an update phase. We adapt thus the definition of secure key updates of Dodis et al. [7] to the sequential updates context of our schemes.

Definition 4 A key-insulated signature scheme has *secure key updates* if the view of an adversary A making a key exposure during an update from time period i to time period $i+1$ can be perfectly simulated by an adversary A' making a key exposure at periods i and $i+1$. □

3 A First Scheme

We describe here a first signature scheme inspired from the forward-secure signature scheme of Zhou et al. [15] and respecting a slightly stronger definition of forward-security. The new scheme is not completely key-insulated because an attacker compromising the user at time periods i and j, with $i < j$, is able to deduce all the secret keys used between these two time periods. However, as an attacker will need to compromise the user at a second time period before obtaining future secret keys, a user detecting having been compromised can revoke his public key and prevent meanwhile the attacker from forging signatures for time periods comprised between the exposure and revocation instants. This differs from classical forward-secure schemes, in which valid forged signatures can be produced for the time periods comprised between exposure and revocation.

The signature and verification algorithms of our scheme are based on the Guillou-Quisquater signature scheme [9]. In the next section we will show how the scheme presented hereafter can be modified in order to achieve perfect key-insulation.

$\mathsf{KGen}(k, l)$

> k and l are two security parameters. Let $n = pq$ be a k-bit modulus, where $p = 2p' + 1$ and $q = 2p' + 1$ are safe primes numbers such that p' and q' are also safe primes. Let v be an $(l+1)$-bit prime number. And let h be a one-way hash function $h : \{0,1\}^* \rightarrow \{0,1\}^l$ (in the following we will note by $h(a, b)$ the result of applying to h the concatenation of a value a with a value b).
> The user randomly chooses $t, u \in \mathbb{Z}_n^*$, such that $s^2 \neq s^{2^{8+1}} \bmod n$ and $t^2 \neq t^{2^{8+1}} \bmod n$. The public key PK is composed of $PK_1 = t^{-v} \bmod n$ and $PK_2 = u^{-v} \bmod n$. The master secret key is $MSK = t^2 \bmod n$ and the user's initial secret key is $USK_0 = u^2 \bmod n$.

$\mathsf{UpdD}(i, N, MSK)$

> The physically secure device computes the partial secret key
>
> $$PSK_i = (MSK)^{2^{N-i}} \bmod n = t^{2^{N+1-i}} \bmod n.$$

$\mathsf{UpdU}(i, USK_{i-1}, PSK_i)$

The user computes the user's secret key for the time period i

$$USK_i = (USK_{i-1})^2 \bmod n = u^{2^{i+1}} \bmod n$$

and the corresponding secret key

$$SK_i = PSK_i \cdot USK_i \bmod n = t^{2^{N+1-i}} \cdot u^{2^{i+1}} \bmod n.$$

$\mathsf{Sig}_{SK_i}(i, M)$

In order to sign a message M during the time period i, the user randomly chooses a value $x \in \mathbb{Z}_n^*$, computes $y = x^v \bmod n$, $d = h(i, M, y)$ and $D = x \cdot (SK_i)^d \bmod n$. The signature of M for the time period i is (i, d, D).

$\mathsf{Ver}_{PK}(M, (i, d, D), N)$

For verifying if (i, d, D) is a valid signature on M for the time period i, an entity computes

$$h(i, M, D^v \cdot ((PK_1)^{2^{N+1-i}} \cdot (PK_2)^{2^{i+1}})^d \bmod n)$$

and accepts the signature only if the result is equal to d. If the signature is valid then this equality holds:

$$\begin{aligned}
&h(i, M, D^v \cdot ((PK_1)^{2^{N+1-i}} \cdot (PK_2)^{2^{i+1}})^d \bmod n) \\
&= h(i, M, (x \cdot (SK_i)^d)^v \cdot ((\frac{1}{t^v})^{2^{N+1-i}} \cdot (\frac{1}{u^v})^{2^{i+1}\ d}) \bmod n) \\
&= h(i, M, x^v \cdot (t^{2^{N+1-i}} \cdot u^{2^{i+1}})^{d \cdot v} \cdot (\frac{1}{t^{2^{N+1-i}}} \cdot \frac{1}{u^{2^{i+1}}})^{d \cdot v} \bmod n) \\
&= h(i, M, x^v \bmod n) \\
&= d
\end{aligned}$$

3.1 Analysis

In this section we show that the above scheme is forward-secure but not key-insulated. Indeed, the scheme is built in such a way that it is easy to compute past partial secret keys when knowing a current value of this secret:

$$PSK_{i-j} = (PSK_i)^{2^j} \bmod n \qquad \forall\, j > 0.$$

Moreover, the user's secret key is computed in a forward-secure fashion:

$$USK_{i+j} = (USK_i)^{2^j} \bmod n \qquad \forall\, j > 0.$$

Therefore, in the improbable case of a user's compromise at two different time periods i and j, with $i < j$, an opponent will obtain SK_i and USK_i on period i, as well as SK_j and USK_j on period j. By deriving PSK_j from the

latter two secrets and appropriately combining it with USK_i, he will be able to compute any secret signing key

$$SK_r = (USK_i)^{2^\ell} \cdot (PSK_j)^{2^{j-i-\ell}} \bmod n \qquad \forall\, \ell \in [1, j-i-1]\,,$$

comprised between time periods i and j.

However, other secret signing key values are kept secret since for $r < i$ the value of USK_r can not be easily derived and for $r > j$ the value of PSK_r can neither be easily computed.

Consequently, the scheme is not key-insulated, but following this scenario it may be considered more robust than traditional forward-secure schemes since an opponent needs to compromise the user at a second time period before being able to compute future secret keys. This robustness is achieved thanks to the physically secure update device, which is not used in classical forward-secure signature schemes. Note that this scheme can be proven forward-secure, according to its usual definition [4, 1, 11, 13], in a similar way than in the security proof for the key-insulated scheme presented in the next section.

4 A Strong and Perfectly Key-Insulated Scheme

We present now our perfectly key-insulated scheme. Basically, it remains quite similar to the scheme described above. The main difference is that the partial secret keys output by the physically secure device are based on the product between a value belonging to an increasing series of powers of 2 and its counterpart in a decreasing series of powers of 2, as the secret keys of the first scheme.

$\mathsf{KGen}(k, l)$

k and l are two security parameters. Let $n = pq$ be a k-bit modulus, where $p = 2p' + 1$ and $q = 2p' + 1$ are safe primes numbers such that p' and q' are also safe primes. Let v be an $(l+1)$-bit prime number. And let h be a one-way hash function $h : \{0,1\}^* \rightarrow \{0,1\}^l$.
The user randomly chooses $s, t, u \in \mathbb{Z}_n^*$, such that $s^2 \neq s^{2^{8+1}} \bmod n$, $t^2 \neq t^{2^{8+1}} \bmod n$ and $u^2 \neq u^{2^{8+1}} \bmod n$. The public key PK is composed of $PK_1 = s^{-v} \bmod n$, $PK_2 = t^{-v} \bmod n$ and $PK_3 = u^{-v} \bmod n$. The master secret key MSK is composed of $MSK_1 = s^2 \bmod n$ and $MSK_2 = t^2 \bmod n$, and the user's initial secret key is $USK_0 = u^2 \bmod n$.

$\mathsf{UpdD}(i, N, MSK)$

The device computes the partial secret key for the i-th time period as follows:

$$PSK_i = (MSK_1)^{2^i} \cdot (MSK_2)^{2^{N-i}} \bmod n = s^{2^{i+1}} \cdot t^{2^{N+1-i}} \bmod n.$$

Note that in order to compute the next partial secret key, the device only needs to store $(MSK_1)^{2^i}$ and MSK_2.

$\mathsf{UpdU}(i, USK_{i-1}, PSK_i)$

The user computes the user's secret key for the time period i

$$USK_i = (USK_{i-1})^2 \bmod n = u^{2^{i+1}} \bmod n$$

and the corresponding secret key

$$SK_i = PSK_i \cdot USK_i \bmod n = s^{2^{i+1}} \cdot t^{2^{N+1-i}} \cdot u^{2^{i+1}} \bmod n.$$

$\mathsf{Sig}_{SK_i}(i, M)$

In order to sign a message M during the time period i, the user randomly chooses a value $x \in \mathbb{Z}_n^*$, computes $y = x^v \bmod n$, $d = h(i, M, y)$ and $D = x \cdot (SK_i)^d \bmod n$. The signature of M for the time period i is (i, d, D).

$\mathsf{Ver}_{PK}(M, (i, d, D), N)$

For verifying if (i, d, D) is a valid signature on M for the time period i, an entity computes

$$h(i, M, D^v \cdot ((PK_1)^{2^{i+1}} \cdot (PK_2)^{2^{N+1-i}} \cdot (PK_3)^{2^{i+1}})^d \bmod n)$$

and accepts the signature only if the result is equal to d. If the signature is valid then this equality holds:

$$\begin{aligned}
&h(i, M, D^v \cdot ((PK_1)^{2^{i+1}} \cdot (PK_2)^{2^{N+1-i}} \cdot (PK_3)^{2^{i+1}})^d \bmod n) \\
&= h(i, M, (x \cdot (SK_i)^d)^v \cdot ((\frac{1}{s^v})^{2^{i+1}} \cdot (\frac{1}{t^v})^{2^{N+1-i}} \cdot (\frac{1}{u^v})^{2^{i+1}})^d \bmod n) \\
&= h(i, M, x^v \cdot (s^{2^{i+1}} \cdot t^{2^{N+1-i}} \cdot u^{2^{i+1}})^{d \cdot v} \\
&\qquad \cdot (\frac{1}{s^{2^{i+1}}} \cdot \frac{1}{t^{2^{N+1-i}}} \cdot \frac{1}{u^{2^{i+1}}})^{d \cdot v} \bmod n) \\
&= h(i, M, x^v \bmod n) \\
&= d
\end{aligned}$$

4.1 Analysis

We begin by proving that the keys take a large number of values before cycling. This avoids the possibility for an attacker to merely wait for a new occurrence of the cycle in order to obtain the secret key for a future time period.

Since n has been defined as the product of two prime numbers p and q such that:

$$\begin{cases} p = 2p' + 1, \text{ with } p' = 2p'' + 1 \text{ and } p'' \text{ prime} \\ q = 2q' + 1, \text{ with } q' = 2q'' + 1 \text{ and } q'' \text{ prime,} \end{cases}$$

we have that $\phi(n) = 4p'q'$.

The secret keys of our scheme are the product of three terms of the form a^{2^ℓ}. The length of a cycle for this product is the least common multiple of the cycle

lengths for each of these three terms. Notice that the probability to compute a previously computed secret key before cycling is negligible (less than $\phi(n)^{-1}$).

We can hence work on one of these cycles. Looking for the smallest possible cycle then reduces to find i and j such that $i < j$ and j is the smallest number such that $a^{2^i} = a^{2^j} \bmod n$. The length of a cycle is therefore $j - i$.

By testing beforehand, it is easy to prohibit cycles of length 2 or 4. We can then deduce that $2^i = 2^j \bmod z$, where z is a divisor of $4p'q'$, distinct of 2 and 4. Again, this lead us to the fact that $i = j \bmod z'$, where z' is a divisor of $\phi(z)$. The only possible values for z' which are smaller than p'' and q'' are 2, 4 and 8. As above, we can avoid these three particular cases by doing a small quantity of tests before using the key. This implies that $j - i \geq \min(p'', q'')$.

We will now prove that the scheme described above is strong and perfectly key-insulated and that it has secure key updates. Security is proven in the random oracle model and is based on the strong RSA assumption [3, 8], which states that, given a number n that is the product of two prime numbers and a value $\alpha \in \mathbb{Z}_n^*$, it is computationally infeasible to find $\beta \in \mathbb{Z}_n^*$ and $r > 1$ such that $\beta^r = \alpha \bmod n$.

Theorem 1 The scheme described in section 4 is strong and perfectly key-insulated and has secure key updates.

Proof: Suppose we are given a forger F (a probabilistic polynomial time Turing machine) that after a polynomial amount of time and after querying a finite number of times a signing oracle and, possibly, a key exposure oracle (as described in definitions 2 and 3) as well as a random oracle (allowing us to model the hash function), produces with non-negligible probability a valid signature for a message and a time period never submitted to the signing oracle and a time period never submitted to the key exposure oracle. We will show how to use F in order to solve a particular instance of the strong RSA problem.

In order to answer oracle queries, we maintain a hash query table and a signature query table. Each time F queries the random oracle on (i_j, M_j, y_j) we check if $h(i_j, M_j, y_j)$ has already been defined. If so, we answer with $h(i_j, M_j, y_j)$, otherwise we answer with a new randomly chosen value $d_j \in \{0,1\}^l$ and record (i_j, M_j, y_j, d_j) in the hash query table.

Every time F queries the signing oracle on a new (i_j, M_j) pair, we randomly chose $d_j \in \{0,1\}^l$ and $D_j \in \mathbb{Z}_n^*$ such that $h(i_j, M_j, {D_j}^v \cdot (PKi_j)^{d_j} \bmod n)$ has not already been defined, where PKi_j equals $(PK_1)^{2^{i_j+1}} \cdot (PK_2)^{2^{N+1-i_j}} \cdot (PK_3)^{2^{i_j+1}} \bmod n$. We set $h(i_j, M_j, {D_j}^v \cdot (PKi_j)^{d_j} \bmod n)$ to d_j, record (i_j, M_j, d_j, D_j) in the signature query table and output (i_j, d_j, D_j) as signature.

Finally, when answering a key exposure oracle query i_j, we simply compute USK_{i_j} and SK_{i_j} from MSK and USK_0 and give the former two keys to F.

We run F by giving it a random tape, n, v, PK and, possibly, MSK. After a polynomial amount of time, F outputs a signature (i_1, d_1, D_1) on a message M_1. Let y_1 be ${D_1}^v \cdot (PKi_1)^{d_1} \bmod n$. Note that the entry (i_1, M_1, y_1) has to be present in the hash query table and that (i_1, M_1) must not exist in the signature query table.

Furthermore, if F is given access to a key exposure oracle, we have, by definition 2, that this oracle can give to F the secret keys stored on the insecure device for up to $N-1$ time periods, obtaining therefore $\{USK_{i_j}, SK_{i_j}\}\ \forall\, 1 \leq i_j \leq N$, with $i_j \neq i_1$. From USK_{i_1-1}, F would be able to compute USK_{i_1} by simply performing a modular squaring. However, it will be unable to compute PSK_{i_1} from any other PSK_{i_j} since the factors of each PSK_{i_j} are unknown to him. (The property of secure key updates follows directly from this fact.)

On the other hand, by definition 3, we would have given to F the keys used by the physically secure device, that is to say MSK_1 and MSK_2, from which F is able to compute PSK_{i_1}. However, it will not be able to find by itself the corresponding USK_{i_1} value allowing him to compute SK_{i_1}, since it can not query now a key exposure oracle.

We reset F and run it again by giving it the same random tape as before, n, v, PK and, possibly, MSK. We give the same answers to F's oracle queries as during its first execution until (i_1, M_1, y_1) is queried to the random oracle, in which case we reply with a new randomly chosen value $d_1' \neq d_1$. From that moment we reply to F's oracle queries with new randomly chosen answers.

Again, after a polynomial amount of time F outputs a forged signature (i_2, d_2, D_2) on a message M_2. Let y_2 be ${D_2}^v \cdot (PKi_2)^{d_2} \bmod n$. With non-negligible probability [14], we have that $(i_1, M_1, y_1) = (i_2, M_2, y_2)$ and therefore $d_1' = d_2$. From this we have that ${D_1}^v \cdot (PKi_1)^{d_1} = {D_2}^v \cdot (PKi_2)^{d_2}$, with $d_1 \neq d_2$. Consequently, we have that

$$\left(\frac{D_1}{D_2}\right)^v = (PKi_1)^{d_2 - d_1}.$$

As v is a prime number, we have that v and $d_2 - d_1$ are relatively prime. Note that these two values are not equal since v is an $(l+1)$-bit number and $d_2 - d_1$ has at most l bits. By applying Bézout's theorem, we can find two integers a and b such that $av + b(d_2 - d_1) = 1$ and compute

$$\begin{aligned} PKi_1 &= (PKi_1)^{av} \cdot (PKi_1)^{b(d_2-d_1)} \bmod n \\ &= (PKi_1)^{av} \cdot \left(\frac{D_1}{D_2}\right)^{bv} \bmod n \\ &= \left((PKi_1)^a \cdot \left(\frac{D_1}{D_2}\right)^b\right)^v \bmod n \end{aligned}$$

By letting $\beta = (PKi_1)^a \cdot \left(\frac{D_1}{D_2}\right)^b$ and $r = v$, we solve the strong RSA problem for $\alpha = PKi_1$. Note that by guessing the time period i_1 during which F will forge a signature, it would be possible to compute the appropriate values for the secret and public keys in order to solve the strong RSA problem for any other value of α. As the algorithm solving the strong RSA problem that we have constructed runs in polynomial time, contradicting therefore the intractability assumption, we have that the success probability of F has to be negligible. By definitions 2 and 3 we can therefore conclude that the scheme is strong and perfectly key-insulated. □

5 Comparison

In this section we look at the performances of our key-insulated scheme and compare it with the other existing schemes, namely those described in [7].

Dodis et al. propose [7] three strong key-insulated schemes. The first one is generic and therefore not suitable for precise comparisons. Moreover, the produced signatures are composed of two signatures from the scheme upon which the key-insulated scheme is based, what implies computation and transmission overheads.

We will therefore compare our scheme with the remaining schemes they propose. Their second scheme, named DKXY2 in the following, is an interesting and practical scheme based on the Okamoto-Schnorr signature. However, it has the drawback to have keys size which grow linearly with the number of insulated time periods, whereas our scheme has key length independent of the number of time periods. For this comparison, we consider the perfectly key-insulated version of this scheme, i.e. where the number of insulated time periods equals $N - 1$.

Finally, their the third scheme, called hereafter DKXY3, is an efficient and generic perfectly key-insulated signature scheme. For the sake of comparison, we study it in its RSA-based version (for the one-way trapdoor function).

We set that n is a 512 bits modulus and that the size of v is 160 bits. For the comparison to be effective, we only consider the number of modular multiplications, which are the most time-consuming operations of those schemes, and neglect other computations such as hashing. We assume, as usual, that a modular exponentiation is equivalent to $1.5 \cdot \ell$ modular multiplications, where ℓ is the size in bits of the exponent.

The most consuming step of the key generation part of DKXY2 is the computation of $2N$ modular exponentiations with 160 bits exponents, as well as one modular multiplication. The update device needs to compute twice $N - 1$ modular multiplications during UpdD. The update step for the user is negligible.

The complete key generation part of DKXY3 cannot be detailed since no precise signature implementation is proposed (the original paper only describes the setup of RSA keys needed to implement the trapdoor function). However, the update procedures are completely described. Both updates correspond to a modular exponentiation with a 512 bits exponent. The user key-update algorithm requiring moreover a modular multiplication.

Our key generation phase implies three modular exponentiations with 160 bits exponents, three modular inverses and 27 modular squarings, 24 of them, in the best case, in order to avoid small cycles. Our updates need two modular multiplications, two modular squarings that are performed from previously computed squarings and one modular exponentiation where the size of the exponent corresponds, in the worst case, to the number of time periods.

In table 1 we indicate the complexity of the update stages for each of these three schemes in terms of modular multiplications. We can see that our scheme is at least as efficient or much more efficient than the ones of Dodis et al. for each algorithm.

Table1. Performances of the update phases

	DKXY2	DKXY3	New scheme
UpdD	$2(N-1)$	768	$2+N$
UpdU	0	769	2

The signature process of DKXY2 counts for two modular exponentiations with 160 bits exponents and three modular multiplications. Their verification process needs, in the one hand, the computation of $N-1$ modular multiplications and N modular exponentiations with small exponents, that we simplify in N modular multiplications. In the other hand, three modular exponentiations with 160 bits exponents and two modular multiplications are performed.

Our signature process is similar to the Guillou-Quisquater scheme, and our verification algorithm requires two modular squarings, two modular exponentiations with 160 bits exponents, one modular exponentiation with an exponent whose size corresponds, in the worst case, to the number of time periods and three modular multiplications.

As no practical underlying signature scheme is detailed for DKXY3, table 2 only indicates the complexity of the key generation, signature and verification algorithms for DKXY2 and our scheme in terms of modular multiplications. Anew, we can see that our scheme is more efficient than the one of Dodis et al.

Table2. Performances of the key generation, signature and verification algorithms

	DKXY2	New scheme
KGen	$481N$	750
Sig	483	481
Ver	$2N+721$	$485+N$

Our scheme's computations are partially dependent on the number of time periods, in contrary to DKXY3. However, we may consider that in our scheme the key generation step may be realised periodically (one time each year for example). This way of doing allows to keep better performances at a very acceptable price (a price which is diluted within the price of computations achieved on a long period of time).

Unlike DKXY2, where the key size depends on the number of time periods, our scheme makes use of constant length keys. The secret keys used by the user and the update device, i.e. MSK_1, MSK_2 and USK_0, counts for 1536 bits in our scheme. The public keys used when verifying signatures, i.e. PK_1, PK_2 and PK_3, count also for 1536 bits in our scheme. We may notice that the size of our scheme's public keys is not critical since it is not required to store them in a not very powerful tamper-proof device. The size of our scheme's secret keys remains reasonable, particularly when considering the efficiency of the secret

keys update procedure which may be performed regularly within a short period of time (e.g. daily).

6 Conclusion

We have proposed in this paper a new strong and perfectly key-insulated signature scheme, inspired from the forward-secure scheme of Zhou et al. [15], more efficient than other previously known key-insulated signature schemes. Its key length is constant and does not depend on the number of insulated time periods. Although being considerably fast, updates for the physically secure device consist in performing a certain number of modular squarings that depends linearly on the total number of time periods. However, this number becomes smaller as the number of elapsed time periods grows. On the other side, updates for users require only two modular multiplications.

The signature and verifications algorithms of our key-insulated signature scheme are based on the Guillou-Quisquater scheme [9]. As for the update algorithm for the physically secure device, the verification algorithm performs a certain number of modular multiplications that depends linearly on the total number of time periods, but that decreases as the number of elapsed time periods grows.

Moreover, the way how updates are done allows this scheme to become forward-secure when all the existing secrets at a given time period are compromised. This property, not respected in other key-insulated schemes, provides increased security to signatures having been produced at previous time periods without additional infrastructure.

References

[1] M. Abdalla and L. Reyzin. A new forward-secure digital signature scheme. In *Proceedings of Advances in Cryptology – ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 116–129. Springer-Verlag, Dec. 2000.

[2] R. Anderson. Invited lecture, 4th Conference on Computer and Communications Security. ACM, 1997.
`http://www.cl.cam.ac.uk/TechReports/UCAM-CL-TR-549.pdf`.

[3] N. Barić and B. Pfitzmann. Collision-free accumulators and fail-stop signatures schemes without trees. In *Proceedings of Advances in Cryptology – EUROCRYPT 97*, volume 1233 of *Lecture Notes in Computer Science*, pages 480–494. Springer-Verlag, May 1997.

[4] M. Bellare and S. K. Miner. A forward-secure digital signature scheme. In *Proceedings of Advances in Cryptology – CRYPTO 99*, volume 1666 of *Lecture Notes in Computer Science*, pages 431–448. Springer-Verlag, Aug. 1999.

[5] Y. Desmedt and Y. Frankel. Threshold cryptosystems. In *Proceedings of Advances in Cryptology – CRYPTO 89*, volume 435 of *Lecture Notes in Computer Science*, pages 307–315. Springer-Verlag, Aug. 1989.

[6] Y. Dodis, J. Katz, S. Xu, and M. Yung. Key-insulated public key cryptosystems. In *Proceedings of Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 65–82. Springer-Verlag, Apr. 2002.

[7] Y. Dodis, J. Katz, S. Xu, and M. Yung. Strong key-insulated signature schemes. In *Proceedings of the 6th International Workshop on Practice and Theory in Public Key Cryptography (PKC 2003)*, volume 2567 of *Lecture Notes in Computer Science*, pages 130–144. Springer-Verlag, Jan. 2003.

[8] E. Fujisaki and T. Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In *Proceedings of Advances in Cryptology – CRYPTO 97*, volume 1294 of *Lecture Notes in Computer Science*, pages 16–30. Springer-Verlag, Aug. 1997.

[9] L. C. Guillou and J.-J. Quisquater. A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory. In *Proceedings of Advances in Cryptology – EUROCRYPT 88*, volume 330 of *Lecture Notes in Computer Science*, pages 123–128. Springer-Verlag, May 1988.

[10] A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, and M. Yung. Proactive public key and signature systems. In *Proceedings of the 4th Conference on Computer and Communications Security*, pages 100–110. ACM, 1997.

[11] G. Itkis and L. Reyzin. Forward-secure signatures with optimal signing and verifying. In *Proceedings of Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 332–354. Springer-Verlag, Aug. 2001.

[12] G. Itkis and L. Reyzin. SiBIR: Signer-base intrusion-resilient signatures. In *Proceedings of Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 499–514. Springer-Verlag, Aug. 2002.

[13] A. Kozlov and L. Reyzin. Forward-secure signatures with fast key update. In *Proceedings of the 3rd International Conference on Security in Communication Networks (SCN 2002)*, volume 2576 of *Lecture Notes in Computer Science*, pages 241–256. Springer-Verlag, Sept. 2002.

[14] D. Pointcheval and J. Stern. Security proofs for signature schemes. In *Proceedings of Advances in Cryptology – EUROCRYPT 96*, volume 1070 of *Lecture Notes in Computer Science*, pages 387–398. Springer-Verlag, May 1996.

[15] J. Zhou, F. Bao, and R. Deng. Private communication.

[16] J. Zhou, F. Bao, and R. Deng. Validating digital signatures without TTP's time-stamping and certificate revocation. In *Proceedings of the 6th Information Security Conference (ISC 2003)*, volume 2851 of *Lecture Notes in Computer Science*, pages 96–110. Springer-Verlag, Oct. 2003.