Efficient Intrusion-Resilient Signatures Without Random Oracles

Benoît Libert[1] [⋆], Jean-Jacques Quisquater[1], and Moti Yung[2]

[1] UCL, Microelectronics Laboratory, Crypto Group (Belgium)
[2] RSA Labs and Columbia University (USA)

Abstract. Intrusion-resilient signatures are key-evolving protocols that extend the concepts of forward-secure and key-insulated signatures. As in the latter schemes, time is divided into distinct periods where private keys are periodically updated while public keys remain fixed. Private keys are stored in both a user and a base; signature operations are performed by the user while the base is involved in periodic updates. Such a system remains secure after arbitrarily many compromises of both modules as long as break-ins are not simultaneous. Besides, when they simultaneously occur within some time period, past periods remain safe. In this work, we propose the first intrusion-resilient signature in the standard model (i.e. without random oracles) which provides both short signatures and at most log-squared private storage in the number of time periods.

Keywords. Intrusion-resilience, standard model, signatures, pairings.

1 Introduction

Key exposures seem to be inevitable and containing their damage is an extremely important issue in cryptography. The late nineties and the past recent years witnessed the exploration of various approaches to address the problem.

Among them, the concept of *intrusion-resilient* security [26] strives to combine the benefits of forward-security [3, 5], key-insulated [17, 18] and proactive [36, 23] security paradigms. As in [3, 5, 17, 18], intrusion-resilient systems involve public keys that remain unchanged throughout the lifetime of the protocol while private keys evolve at the beginning of discrete time intervals. Like key-insulated schemes, they involve a physically-secure (but computationally-limited) device called *base* where certain keys are stored. These keys are used to periodically update the user's short-term secret which is used to sign or decrypt messages. As in [17, 18], this is accomplished so as to preserve the security of past and future time periods when the signer is compromised (unlike forward-secure cryptosystems [3, 5, 13] that only protect past time periods). Besides, the intrusion-resilient model preserves the security in case of compromise of both the signer and the base as long as they are not simultaneously broken into. Moreover, the security of past (but not future) periods is retained after such a simultaneous attack.

[⋆] This author acknowledges the DGTRE's First Europe Program of the Walloon Region in Belgium for its financial support.

Security in this strong adversarial model is achieved via frequent refreshes (akin to proactive mechanisms [36, 23]) of base and user keys within each time period: the base changes its key and sends a refresh message to the user who in turn refreshes his key accordingly. Refreshes may take place at arbitrary times and are transparent to signature verifiers or message senders. They are meant to prevent attackers compromising the base and the user without a refresh in between to threaten the security of other time periods. Besides, when an adversary learns simultaneous base and user keys, the scheme "becomes" forward-secure in that past time periods remain safe.

In [26], Ikis and Reyzin proposed the first intrusion-resilient signature which is based on the GQ signature scheme [21] and resorts to the idealized random oracle model [6] which is known [12] to *only* provide heuristic arguments. Promptly later, Itkis [27] described a generic construction of intrusion-resilient signature using any ordinary digital signature and without employing random oracles. In 2003, Dodis *et al.* put forward the first intrusion-resilient public key encryption scheme [15] built on the forward-secure cryptosystem of Canetti *et al.* [13] which itself stems from the hierarchical identity-based encryption (HIBE) scheme of [19], the latter being an extension of [9]. They subsequently explained [16] how to generally obtain such a primitive from forward-secure cryptosystems with suitable properties. In 2004, Malkin, Obana and Yung [33] showed an equivalence relation between key-insulated, intrusion-resilient and proxy signatures [34]. They proved that all these primitives imply forward-secure signatures. Their results imply the existence of all these kinds of signatures in the standard model since key-insulated signatures are known [18] to be implied by identity-based signatures [38] which exist in the standard model [37]. However, in intrusion-resilient and forward-secure signatures obtained by applying the generic constructions of [33] to some key-insulated scheme, private keys have linear length in the number of time periods. Hence, we may hope for more efficient non-generic constructions.

In this paper, we propose an intrusion-resilient signature which is secure in the standard model and has at most poly-logarithmic complexity (in the number of stages) in all parameters. It utilizes bilinear maps and features constant-size signatures. For practical numbers of periods, public keys are not significantly longer than in a scheme derived from [37] using generic conversions of [33]. Our method combines Waters's signature [39] with a hierarchical key derivation technique borrowed from [8]. We first construct a special kind of forward-secure signature with suitable "homomorphic" properties which is of independent interest[1]. We then achieve an intrusion-resilient scheme by applying a generic conversion suggested by Dodis *et al.* [16] in the setting of public key encryption and which is easily seen to apply for signature schemes as well. The resulting system yields much shorter signatures than Itkis's generic method [27]: in the latter implemented over N stages, each signature contains a sequence of $\log N$

[1] After the completion of this work, we were informed that our forward-secure signature was independently discovered in [10] where it was provided with an additional property.

one-time signatures[2] [31] and their public keys (that are typically very long).

In the following, section 2.1 defines a proper model for special forward-secure signatures that serve our purposes. Section 2.2 recalls definitions and security notions for intrusion-resilient signatures. Our forward-secure scheme and its intrusion-resilient variant are respectively analyzed in sections 3 and 4.

2 Preliminaries

2.1 Key-evolving signatures

A *key-evolving signature* is a forward-secure signature [3, 5] where the user's secret key can be "divided" into a *local key*, only used in signing operation, and an *update key* which is involved in key updates and not in signature generation.

Definition 1. *A key-evolving signature is specified by the following algorithms.*

Keygen: *takes as input a security parameter λ and a number of time periods N. It returns a public key $\overline{pk}$ and an initial user update key $\overline{sk}_0$.*

Update: *takes as input a period number i and the corresponding update key $\overline{sk}_i$. It returns $SK_{i+1} = (\overline{lsk}_{i+1}, \overline{sk}_{i+1})$ where $\overline{sk}_{i+1}$ is the next user update key and $\overline{lsk}_{i+1}$ is the next user local key.*

Sign: *takes as input a message M, a period number i and the matching user local key $\overline{lsk}_i$. It returns a signature σ.*

Verify: *takes as input $\overline{pk}$, a period number i and a message M bearing some purported signature σ. It outputs either 0 or 1.*

The usual completeness requirement imposes $\mathsf{Verify}(\overline{pk}, i, M, \sigma) = 1$ *whenever* $\sigma = \mathsf{Sign}(M, i, \overline{lsk}_i)$ *and* $SK_i = (\overline{lsk}_i, \overline{sk}_i) = \mathsf{Update}(\overline{sk}_{i-1})$.

Definition 2. *A key-evolving signature over N stages is secure against chosen-message attacks if no PPT adversary has non-negligible advantage in this game.*

1. *The challenger $\mathcal{C}$ runs the key generation algorithm to obtain $(\overline{pk}, \overline{sk}_0)$ and gives $\overline{pk}$ to the forger $\mathcal{F}$.*
2. *$\mathcal{F}$ interacts with the following oracles.*
   - *An update-key oracle $O_{ukey}(\overline{sk}_0, .)$ which, on input of $i \in \{0, \ldots, N-1\}$, returns $\overline{sk}_i$ (which is appropriately derived from $\overline{sk}_0$).*
   - *A local-key oracle $O_{lkey}(\overline{sk}_0, .)$ which, on input of i, returns $\overline{lsk}_i$ (which is again appropriately derived from $\overline{sk}_0$).*
   - *A signing oracle $O_{sig}(\overline{sk}_0, .)$ taking as inputs a period number i and a message M. From $\overline{sk}_0$, it derives $\overline{lsk}_i$ that is used to output a signature σ on M for period i.*
3. *$\mathcal{F}$ comes up with a message M and a signature σ for some period $i^\star$. She wins if* $\mathsf{Verify}(\overline{pk}, i^\star, M, \sigma) = 1$ *with these restrictions: M was not queried to O_{sig}; queries i to O_{ukey} satisfy $i > i^\star$ and queries i' to O_{lkey} satisfy $i' \neq i^\star$.*

[2] This can be reduced to $\log i$, where i is the period number, as shown in [27].

$\mathcal{F}$'s advantage is her probability of victory taken over coin tosses of $\mathcal{A}$ and $\mathcal{C}$. We say that she (t, q_s, ε)-breaks the scheme if she has advantage ε within running in time t and after q_s signing queries.

As its counterpart for encryption [16], this model is stronger than the standard security model [5] of forward-secure signatures in that adversaries are allowed to obtain local keys for any period but $i^\star$ (and not just for periods $i > i^\star$).

2.2 Intrusion-resilient signatures

An intrusion-resilient signature [26] scheme consists of the following algorithms.

Keygen: takes a security parameter λ, a number of periods N and a maximal number of refreshes R in each period. It returns an initial user key $SKS_{0.0}$, an initial base key $SKB_{0.0}$ and a public key PK.

Base Update: takes as input a current base key $SKB_{i.r}$ (for period i, after r refreshes) and outputs the next base key $SKB_{i+1.0}$ together with a key update message SKU_i.

User Update: is given a current signer key $SKS_{i.r}$ (for period i, after r refreshes) and an update message SKU_i. It outputs the next user key $SKS_{i+1.0}$.

Base Refresh: takes as input a current base key $SKB_{i.r}$ and outputs a refreshed base key $SKB_{i.r+1}$ along with a key refresh message $SKR_{i.r}$.

User Refresh: takes a current signer key $SKS_{i.r}$ and a refresh message $SKR_{i.r}$ to return a refreshed signing key $SKS_{i.r+1}$.

Sign: given a message M, period/refresh numbers (i, r) and the matching signing key $SKS_{i.r}$, this algorithm generates a signature σ.

Verify: takes as input PK, a period number i and a message M with an alleged signature σ. It outputs either 0 or 1.

Syntactically, in such a scheme, private keys are generated as follows:

```
Set $(SKS_{0.0}, SKB_{0.0}, PK) \leftarrow \mathsf{Keygen}(\lambda, N, R)$.
For $i = 0$ to $N - 1$:
    Set $(SKB_{i+1.0}, SKU_i) \leftarrow \mathsf{Base\ Update}(SKB_{i.r})$
        $SKS_{i+1.0} \leftarrow \mathsf{User\ Update}(SKS_{i.r}, SKU_i)$.
    For $r = 0$ to $R - 1$
        Set $(SKB_{i+1.r+1}, SKR_{i+1.r}) \leftarrow \mathsf{Base\ Refresh}(SKB_{i+1.r})$
            $SKS_{i+1.r+1} \leftarrow \mathsf{User\ Refresh}(SKS_{i+1.r}, SKR_{i+1.r})$.
```

Keys $SKS_{i.0}$ and $SKB_{i.0}$ for $0 \leq i \leq N$ are never actually used or stored as key generation is immediately followed by an update. Besides, each update is followed by a refresh. Hence, adversaries may potentially access these keys:

$$
\begin{aligned}
SKS^* &= \{SKS_{i.r} | 1 \leq i \leq N, 1 \leq r \leq R\},\\
SKB^* &= \{SKB_{i.r} | 1 \leq i \leq N, 1 \leq r \leq R\},\\
SKU^* &= \{SKU_i | 1 \leq i \leq N - 1\},\\
SKR^* &= \{SKR_{i.r} | 1 \leq i \leq N - 1, 0 \leq r \leq R - 1\} \backslash \{SKR_{1.0}\}
\end{aligned}
$$

For $i > 1$, $SKR_{i.0}$ is sent together with SKU_{i-1}. That is why it is accessible.

To define security, we provide a forger $\mathcal{F}$ with the following oracles:

- O_{sig}, the signing oracles which, on input of period/refresh numbers (i, r) and a message M, returns a signature σ.
- O_{sec}, the key exposure oracle which
  1. on input of ("s",$i.r$) for $1 \leq i \leq N$, $1 \leq r, \leq R$, outputs $SKS_{i.r}$.
  2. on input of ("b",$i.r$) for $1 \leq i \leq N$, $1 \leq r, \leq R$, outputs $SKB_{i.r}$.
  3. on input of ("u",i) for $1 \leq i \leq N-1$, outputs SKU_i and $SKR_{i+1.0}$.
  4. on input of ("r",$i.r$) for $1 \leq i \leq N$, $1 \leq r, \leq R$, outputs $SKR_{i.r}$.

It is reasonable to impose adversaries to "respect erasures" in that values that should have been erased may not be queried or used in signature generation:

- ("s", $i.r$) must be queried before ("s", $i'.r'$) if $(i'.r') > (i.r)$;[3]
- ("b", $i.r$) must be queried before ("b", $i'.r'$) if $(i'.r') > (i.r)$;
- ("b", $i.r$) must be queried before ("r", $i'.r'$) if $(i'.r') > (i.r)$;
- ("b", $i.r$) must be queried before ("u", i') if $i' > i$.

For a set Q of key exposure queries, a signing key $SKS_{i.r}$ is said to be Q-exposed if one of the following is true:

- ("s", $i.r$) $\in Q$;
- $r > 1$, ("r", $i.r-1$) $\in Q$ and $SKS_{i.r-1}$ is Q-exposed;
- $r = 1$, ("u", $i-1$) $\in Q$ and $SKS_{i-1.R}$ is Q-exposed;
- $r < R$, ("r", $i.r$) $\in Q$ and $SKS_{i.r+1}$ is Q-exposed;

A completely analogous definition is given for Q-exposure of a base key $SKB_{i.r}$. The scheme is said $(i^\star, Q)$-compromised if $SKS_{i^\star.r}$ is Q-exposed, for some r, or if $SKS_{i'.r}$ and $SKB_{i'.r}$ are both Q-exposed for some $i' < i^\star$. We say that an adversary is successful if, after polynomially-many queries to the above oracles, she produces a message-signature pair $(M^\star, \sigma^\star)$ for some period $i^\star$ provided $M^\star$ was not queried to O_{sig} for period $i^\star$ and the scheme is not $(i^\star, Q)$-compromised.

2.3 Bilinear maps

Groups $(\mathbb{G}, \mathbb{G}_T)$ of prime order p are called *bilinear map groups* if there is a mapping $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ with the following properties:

1. bilinearity: $e(g^a, h^b) = e(g, h)^{ab}$ for any $(g, h) \in \mathbb{G} \times \mathbb{G}$ and $a, b \in \mathbb{Z}$;
2. efficient computability for any input pair;
3. non-degeneracy: $e(g, h) \neq 1_{\mathbb{G}_T}$ whenever $g, h \neq 1_{\mathbb{G}}$.

We require the intractability of the following problem in bilinear map groups.

Definition 3. *The* $\ell + 1$**-Diffie-Hellman Problem** *($\ell+1$-DH) in a group $\mathbb{G}$ generated by g is to compute $g^{(a^{\ell+1})} \in \mathbb{G}$ given $(g, g^a, g^{a^2}, \ldots, g^{(a^\ell)}) \in \mathbb{G}^{\ell+1}$.*

For the applications we have in mind, the strength of the $\ell + 1$-DH assumption does not depend on the number of adversarial queries. Instead, it mildly depends on the number of time periods in the lifetime of the protocol as the parameter ℓ is logarithmic in this number of periods. Hence, this assumption is quite reasonable for any realistic number of periods (such as $N \leq 2^{15}$). However, it sounds much weaker than related assumptions used in [7, 40] where the counterpart of parameter ℓ may be as large as 2^{30} (instead of $\ell \approx 15$ here).

[3] We write $(i'.r') > (i.r)$ when $i' > i$ or $i' = i$ and $r' > r$.

3 A new key-evolving signature without random oracles

Intuitively, the scheme uses the hierarchical key derivation method of [8] in the context of signatures. It can be thought of as using the signature analogue of the concept of Binary Tree Encryption suggested by Canetti, Halevi and Katz [13]. As in [30, 15], we associate time periods with leaves of the tree. To achieve a key-evolving signature in the standard model, we only need a "binary tree signature" which is selective-node [13] and adaptive-message secure (i.e. the adversary has to choose the node to attack ahead of time but can adaptively choose her target message). That is why we implement our hierarchical signature with Waters's signature [39] (which is secure against adaptive chosen-message attacks [20] in the standard model) at the lowest level.

In the description below, we imagine binary tree of height ℓ where the root (at depth 0) has label ε. When a node at depth $\leq \ell$ has label w, its children are labeled with $w0$ and $w1$. Besides, $\langle i \rangle$ stands for the ℓ-bit representation of integer i. The leaves of the tree correspond to successive time periods in the obvious way, stage i being associated with the leaf labeled by $\langle i \rangle$. Periods are indexed from 0 to $N-1$ with $N = 2^{\ell-1}$. As in [30, 15], signatures are generated using the private key of node $\langle i \rangle$ at stage i where the full private key also includes node keys for all right siblings for nodes on the path from $\langle i \rangle$ to the root. The latter key material allows for key updates from period i to the next one.

Keygen: given security parameters $\lambda, n \in \mathbb{N}$ and a number of stages $N = 2^{\ell-1}$,

1. choose bilinear map groups $(\mathbb{G}, \mathbb{G}_T)$ of order $p > 2^\lambda$ and $g \in \mathbb{G}$.
2. Compute $g_1 = g^\alpha$ for a random $\alpha \xleftarrow{R} \mathbb{Z}_p^*$. Choose $g_2, g_3, h_1, \ldots, h_\ell \xleftarrow{R} \mathbb{G}$, $u', u_1, \ldots, u_n \xleftarrow{R} \mathbb{G}$ and compute $Z = e(g_1, g_2)$.
3. Select a collision-resistant hash function $h : \{0,1\}^* \rightarrow \{0,1\}^n$.
4. Define functions $F : \{0,1\}^{\leq \ell} \rightarrow \mathbb{G}$, $G : \{0,1\}^n \rightarrow \mathbb{G}$ as
$$F(w) = g_3 \cdot \prod_{j=1}^{k} h_j^{w_j} \qquad G(m) = u' \cdot \prod_{j=1}^{n} u_j^{m_j}$$
where $w = w_1 \ldots w_k$ and $m = m_1 \ldots m_n$ ($w_i, m_j \in \{0,1\}$ for all i, j). The public key is
$$\overline{pk} = \{g, Z, g_1, g_2, g_3, h_1, \ldots, h_\ell, u', u_1, \ldots, u_n\}.$$
The matching root secret key $\mathsf{sk}_\varepsilon = g_2^\alpha$ is not stored but is directly used to derive lower level node keys.
5. Set $\mathsf{sk}_0 = \big(g_2^\alpha g_3^{r_0}, g^{r_0}, h_2^{r_0}, \ldots h_\ell^{r_0}\big)$ and $\mathsf{sk}_1 = \big(g_2^\alpha (g_3 h_1)^{r_1}, g^{r_1}, h_2^{r_1}, \ldots h_\ell^{r_1}\big)$ with $r_0, r_1 \xleftarrow{R} \mathbb{Z}_p^*$. Using sk_0, recursively apply algorithm $\mathsf{Extract}$ (defined below) to obtain node keys sk_{01}, sk_{001}, $\ldots \mathsf{sk}_{0^{\ell-1}1}$.
6. The initial private update key is $\overline{sk}_0 = \{\mathsf{sk}_1, \mathsf{sk}_{01}, \mathsf{sk}_{001}, \ldots, \mathsf{sk}_{0^{\ell-1}1}\}$.

Extract $(\mathsf{sk}_{w_1\dots w_{k-1}})$: to generate private keys for its children, a node at level $k-1$ parses its private key into

$$\mathsf{sk}_{w_1\dots w_{k-1}} = (a_0, a_1, b_k, \dots, b_\ell) = \left(g_2^\alpha \cdot F(w_1 \dots w_{k-1})^{r'}, g^{r'}, h_k^{r'}, \dots, h_\ell^{r'}\right).$$

For $j = 0, 1$, it chooses a random $t_j \xleftarrow{R} \mathbb{Z}_p^*$ and computes

$$\begin{aligned}\mathsf{sk}_{w_1\dots w_{k-1}j} &= \left(a_0 \cdot b_k^j \cdot F(w_1 \dots w_{k-1}j)^{t_j}, a_1 \cdot g^{t_j}, b_{k+1} \cdot h_{k+1}^{t_j}, \dots, b_\ell \cdot h_\ell^{t_j}\right) \\ &= \left(g_2^\alpha \cdot F(w_1 \dots w_{k-1}j)^{r_j}, g^{r_j}, h_{k+1}^{r_j}, \dots, h_\ell^{r_j}\right)\end{aligned}$$

where $r_j = r' + t_j$.

Update $(SK_i, i+1)$: (where $i < N-1$)

1. Parse $\langle i \rangle$ as $i_0 i_1 \dots i_\ell$ with $i_0 = \varepsilon$. Parse SK_i into

$$(\overline{lsk}_i, \overline{sk}_i) = \left(\mathsf{sk}_{\langle i \rangle}, \{\mathsf{sk}_{i_0 \dots i_{k-1}1}\}_{i_k=0}\right)$$

(where $\overline{lsk}_0$ is undefined if $i = 0$) and erase $\mathsf{sk}_{\langle i \rangle}$.

2. If $i_\ell = 0$, SK_{i+1} simply consists of remaining node keys:

$$SK_{i+1} = \left(\overline{lsk}_{i+1}, \overline{sk}_{i+1}\right) = \left(\mathsf{sk}_{i_0 \dots i_{\ell-1}1}, \{\mathsf{sk}_{i_0 \dots i_{k-1}1}\}_{i_k=0, k<\ell}\right).$$

Otherwise, let $\tilde{k} < \ell$ be the largest index such that $i_{\tilde{k}} = 0$. Let $i' = i_0 \dots i_{\tilde{k}-1}1$. Using $\mathsf{sk}_{i'}$ (which is available as part of $\overline{sk}_i$), recursively apply $\mathsf{Extract}$ to obtain node keys $\mathsf{sk}_{i'1}, \mathsf{sk}_{i'01}, \dots, \mathsf{sk}_{i'0^{\ell-\tilde{k}-1}1}$ and finally $\mathsf{sk}_{\langle i+1 \rangle} = \mathsf{sk}_{i'0^{\ell-\tilde{k}}}$. Erase $\mathsf{sk}_{i'}$ and return remaining keys as SK_{i+1}.

Sign (i, SK_i, M): let $\langle i \rangle = i_1 \dots i_\ell$. Parse SK_i into $\left(\mathsf{sk}_{\langle i \rangle}, \{\mathsf{sk}_{i_0 \dots i_{k-1}1}\}_{i_k=0}\right)$ and $\overline{lsk}_i = \mathsf{sk}_{\langle i \rangle}$ into $(a_0, a_1) = (g_2^\alpha \cdot F(i_1 \dots i_\ell)^r, g^r)$ (which is a key of level ℓ). This algorithm computes $m = h(M)$, chooses $s \xleftarrow{R} \mathbb{Z}_p^*$ and returns

$$(\sigma_0, \sigma_1, \sigma_2) = (a_0 \cdot G(m)^s, a_1, g^s) = (g_2^\alpha \cdot F(i_1 \dots i_\ell)^r \cdot G(m)^s, g^r, g^s).$$

Verify (M, i, PK, σ) : let $\langle i \rangle = i_1 \dots i_\ell$. The purported signature $\sigma = (\sigma_0, \sigma_1, \sigma_2)$ on $m = h(M) \in \{0,1\}^n$ is accepted if and only if

$$\frac{e(\sigma_0, g)}{e(F(i_1 \dots i_\ell), \sigma_1) e(G(m), \sigma_2)} = Z.$$

The completeness is checked by noting that

$$e(\sigma_0, g) = e(g_1, g_2) \cdot e\left(F(i_1 \dots i_\ell), g^r\right) \cdot e\left(G(m), g^s\right).$$

From an efficiency point of view, signature and verification take constant time while key update requires $O(\ell^2)$ exponentiations. The verification cost amounts to a product of three pairings (which is much faster to compute than 3 sequential pairings as discussed in [22]). Combining the hierarchical key derivation of [8] with Waters's signature at the lower level allows for constant-size signatures.

With $\ell = 15$ and for a typical parameter $n = 160$, the public key consists of 179 elements of $\mathbb{G}$ and one element of $\mathbb{G}_T$. If we instantiate the scheme with asymmetric pairings $e : \mathbb{G} \times \mathbb{G}' \rightarrow \mathbb{G}_T$ (over elliptic curves such as those of [4] or [35]), we may obtain signatures of $3 \times 160 = 480$ bits while public keys can be stored within 40 Kb (which remains acceptable in many settings).

Since Waters's signature has a large public key, it is fairly cheap to turn it into a forward-secure signature as we did. Indeed, our scheme performs interestingly w.r.t. the one obtained by applying the MMM [32] construction to Waters's scheme with the motivation to obtain forward-secure signatures in the standard model under soft computational assumptions (details are given in the full version of the paper). However, it answers open questions raised in section 9.1 of [28] as it offers constant-size signatures and at most log-squared complexity in all parameters in the absence of random oracles. It also improves on many previous non-generic schemes [5, 2, 11, 25] which all have some linear cost in N (at least in key generation). The only exceptions are [14, 24, 29] where signatures have logarithmic size.

In the random oracle model, we can even get constant-size public keys as elements $g_2, g_3, h_1, \ldots, h_\ell$ can be derived from a random oracle (as noticed in [8]) and the function G may be also replaced by an independent random oracle. In this case, verification becomes logarithmic since evaluating $F(i_1 \ldots i_\ell)$ requires to compute ℓ hash values on $\mathbb{G}$ (which has non-negligible cost unlike hash operations over $\mathbb{Z}_p^*$). To retain constant-time verification, we can keep $g_2, g_3, h_1, \ldots, h_\ell$ (but not $u', u_1, \ldots, u_n$) in the public key. In summary, the random oracle model yields either constant-size public keys or constant-time verification.

We also observe that ideas from [32] can be applied to support an arbitrary polynomial number of time periods: the number of stages does not have to be known when initializing the scheme.

Theorem 1. *Assuming that a forger $\mathcal{F}$ can (t, q_s, ε)-break the scheme, there is an algorithm $\mathcal{B}$ that (t', ε')-breaks the $\ell + 1$-Diffie-Hellman assumption where*

$$\varepsilon' \geq \frac{\varepsilon}{4Nq_s(n+1)} \qquad t' \leq t + O(q_s t_{exp}),$$

t_{exp} denoting the time complexity of an exponentiation in $\mathbb{G}$.

Proof. We outline an algorithm $\mathcal{B}$ using the forger $\mathcal{F}$ to find $z_{\ell+1} = g^{(a^{\ell+1})}$ given $(z_1, z_2, \ldots, z_\ell) = (g^a, g^{(a^2)}, \ldots, g^{(a^\ell)})$.

At the outset of the game, the simulator $\mathcal{B}$ chooses $i^\star \stackrel{R}{\leftarrow} \{1, \ldots, N-1\}$ as a guess for the time period to be attacked by $\mathcal{F}$. It parses $i^\star$ into $\langle i^\star \rangle = i_1^\star \ldots i_\ell^\star$ and prepares public parameters so as to handle all adversarial queries.

Preparation: to generate the public key, $\mathcal{B}$ picks $\gamma \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ and sets $g_1 = z_1 = g^a$, $g_2 = z_\ell \cdot g^\gamma = g^{\gamma + (a^\ell)}$. The (unknown) root secret key is thereby implicitly set to $\mathsf{sk}_\varepsilon = g_2^a = g^{a\gamma + (a^{\ell+1})} = z_1^\gamma \cdot z_{\ell+1}$. Then, $\mathcal{B}$ chooses $\gamma_1, \ldots, \gamma_\ell, \delta \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ and defines $g_3 = g^\delta \prod_{j=1}^{\ell} z_{\ell-j+1}^{i_j^\star}$ and $h_j = g^{\gamma_j} / z_{\ell-j+1}$ for $j = 1, \ldots, \ell$.

Next, $\mathcal{B}$ picks $\kappa \in \{0, \dots, n\}$ and defines $\tau = 2q_s$. We assume[4] $\tau(n+1) < p$ which implies $0 \leq \kappa\tau < p$. Algorithm $\mathcal{B}$ also selects $x' \xleftarrow{R} \mathbb{Z}_\tau$ and a vector $(x_1, \dots, x_n)$ of elements with $x_i \in \mathbb{Z}_\tau$ for all i. It also chooses at random an integer $y' \xleftarrow{R} \mathbb{Z}_p$ and a vector $(y_1, \dots, y_n)$ with $y_j \in \mathbb{Z}_p$ for all j. For ease of analysis, we consider functions

$$J(m) = x' + \sum_{i=1}^{n} m_i x_i - \kappa\tau \quad \text{and} \quad K(m) = y' + \sum_{i=1}^{n} m_i y_i.$$

taking as input strings $m \in \{0,1\}^n$. Remaining public parameters are chosen as

$$u' = g_2^{x' - \kappa\tau} g^{y'} \qquad u_i = g_2^{x_i} g^{y_i} \text{ for } 1 \leq i \leq n$$

which means that, for any $m \in \{0,1\}^n$, $G(m) = u' \cdot \prod_{i=1}^n u_i^{m_i} = g_2^{J(m)} \cdot g^{K(m)}$. $\mathcal{F}$ is challenged on $(g, Z, g_1, g_2, g_3, h_1, \dots, h_\ell, u', u_1, \dots, u_n)$ with $Z = e(g_1, g_2)$.

Deriving node keys: $\mathcal{B}$ has to compute node private keys for right siblings (whenever they exist) of all nodes on the path from the root to $\langle i^\star \rangle$ (we call this path "crucial path"). For all indexes $k \in \{1, \dots, \ell\}$ such that $i_k^\star = 0$, to generate a private key for node $i^\star|_{\overline{k}} = i_1^\star \dots i_{k-1}^\star 1$, $\mathcal{B}$ first picks $\tilde{r} \xleftarrow{R} \mathbb{Z}_p^*$. If we define $r = \tilde{r} + a^k \in \mathbb{Z}_p^*$, $\mathcal{B}$ is able to compute

$$\mathsf{sk}_{i^\star|_{\overline{k}}} = \left(g_2^a \cdot \left(g_3 \cdot h_1^{i_1^\star} \dots h_{k-1}^{i_{k-1}^\star} h_k\right)^r, g^r, h_{k+1}^r, \dots, h_\ell^r \right).$$

We indeed observe that

$$\left(g_3 \cdot h_1^{i_1^\star} \dots h_{k-1}^{i_{k-1}^\star} h_k\right)^r = \left(g^{\delta + \sum_{j=1}^{k-1} \gamma_j i_j^\star + \gamma_k} \cdot z_{\ell-k+1}^{-1} \cdot \prod_{j=k+1}^{\ell} z_{\ell-j+1}^{i_j^\star} \right)^r$$

where the second term in the product of the right hand side member equals

$$z_{\ell-k+1}^{-r} = z_{\ell-k+1}^{-\tilde{r}} \cdot z_{\ell-k+1}^{-a^k} = z_{\ell-k+1}^{-\tilde{r}} / z_{\ell+1}$$

so that $g_2^a \cdot \left(g_3 \cdot h_1^{i_1^\star} \dots h_{k-1}^{i_{k-1}^\star} h_k\right)^r$ can be computed as

$$z_1^\gamma \cdot \left(g^{\tilde{r}} \cdot z_k\right)^{\delta + \sum_{j=1}^{k-1} \gamma_j i_j^\star + \gamma_k} \cdot z_{\ell-k+1}^{-\tilde{r}} \cdot \prod_{j=k+1}^{\ell} \left(z_{\ell-j+1}^{\tilde{r}} \cdot z_{\ell-j+k+1}\right)^{i_j^\star}.$$

Besides, elements $g^r = g^{\tilde{r}} \cdot z_k$ and $h_j^r = \left(g^{\tilde{r}} \cdot z_k\right)^{\gamma_j} / \left(z_{\ell-j+1}^{\tilde{r}} \cdot z_{\ell-j+k+1}\right)$ (for $j = k+1, \dots, \ell$ and when $k < \ell$) are all computable from available values.

Private keys for right siblings of nodes in the crucial path yield update and local keys for stages $i > i^\star$. In the same way, $\mathcal{B}$ can compute private keys for left siblings whenever they exist. Namely, for all indexes $k \in \{1, \dots, \ell\}$ s.t. $i_k^\star = 1$, a private key for node $i^\star|_{\overline{k}} = i_1^\star \dots i_{k-1}^\star 0$ is computable as

$$\left(g_3 \cdot h_1^{i_1^\star} \dots h_{k-1}^{i_{k-1}^\star}\right)^r = \left(g^{\delta + \sum_{j=1}^{k-1} \gamma_j i_j^\star} \cdot z_{\ell-k+1} \cdot \prod_{j=k+1}^{\ell} z_{\ell-j+1}^{i_j^\star} \right)^r$$

[4] This is a realistic requirement as parameters should be chosen s.t. $n \geq 160$, $p > 2^{160}$ and it is common to suppose $q_s < 2^{30}$.

and $z_{\ell-k+1}^{r} = z_{\ell-k+1}^{\tilde{r}} \cdot z_{\ell-k+1}^{a^k} = z_{\ell-k+1}^{\tilde{r}}/z_{\ell+1}$. Once generated, those keys allow extracting local keys for periods $i < i^\star - 1$ which correspond to leaves at the left of $\langle i^\star \rangle$ in the tree. Hence, $\mathcal{B}$ can compute all local keys for periods $i \neq i^\star$ and not only those for which $i > i^\star$. We note that $\mathcal{B}$ fails if $\mathcal{F}$ ever queries an update key for some period $i < i^\star$. According the rules of definition 2, this can only happen if $\mathcal{B}$ wrongly guesses which period will be attacked by $\mathcal{F}$.

Signing queries: at any time, $\mathcal{F}$ may ask for signatures on messages for any period number i. Those queries are answered using relevant private keys whenever they are computable. To answer signing queries for period $i^\star$, $\mathcal{B}$ uses the same strategy as in the security proof of Waters's signature [39]. At the first message M queried for stage $i^\star$, $\mathcal{B}$ chooses a random $r_{i^\star} \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ that will be used to answer all subsequent queries for period $i^\star$. If $J(m) = 0 \bmod p$, $\mathcal{B}$ aborts. Otherwise, it picks $s \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ and returns

$$\sigma = (\sigma_0, \sigma_1, \sigma_2) = \left(g_1^{-\frac{K(m)}{J(m)}} \cdot F(i_1^\star \ldots i_\ell^\star)^{r_{i^\star}} \cdot G(m)^s, g^{r_{i^\star}}, g_1^{-\frac{1}{J(m)}} \cdot g^s \right).$$

If we define $\tilde{s} = s - a/J(m)$, we indeed have $g_1^{-\frac{1}{J(m)}} \cdot g^s = g^{\tilde{s}}$ and

$$\begin{aligned} g_1^{-\frac{K(m)}{J(m)}} \cdot F(i_1^\star \ldots i_\ell^\star)^{r_{i^\star}} \cdot G(m)^s &= g_1^{-\frac{K(m)}{J(m)}} \cdot F(i_1^\star \ldots i_\ell^\star)^{r_{i^\star}} \cdot G(m)^{\tilde{s}} \cdot \left(g_2^a \cdot g_1^{\frac{K(m)}{J(m)}}\right) \\ &= g_2^a \cdot F(i_1^\star \ldots i_\ell^\star)^{r_{i^\star}} \cdot G(m)^{\tilde{s}}. \end{aligned}$$

Forgery: if $\mathcal{B}$ does not abort and luckily guesses $i^\star$, $\mathcal{F}$ comes up with a forgery $\sigma^\star = (\sigma_0^\star, \sigma_1^\star, \sigma_2^\star)$ on some new message $m^\star = h(M^\star)$ for stage $i^\star$. At that point, $\mathcal{B}$ reports "failure" if $J(m^\star) \neq 0 \bmod p$. Otherwise, $G(m^\star) = g^{K(m^\star)}$ and

$$\sigma_0^\star = g_2^a \cdot F(i_1^\star \ldots i_\ell^\star)^r \cdot g^{sK(m)},\ \sigma_1^\star = g^r,\ \sigma_2^\star = g^s$$

for some $r, s \in \mathbb{Z}_p^*$. Since $F(i_1^\star \ldots i_\ell^\star) = g^{\delta + \sum_{j=1}^{\ell} \gamma_j i_j^\star}$, $\mathcal{B}$ can extract

$$g^{a^{(\ell+1)}} = \frac{\sigma_0^\star}{z_1^\gamma \cdot {\sigma_1^\star}^{\delta + \sum_{j=1}^{\ell} \gamma_j i_j^\star} \cdot {\sigma_2^\star}^{K(m^\star)}}.$$

When analyzing $\mathcal{B}$'s probability of success, we observe that it terminates without aborting if, $J(m) \neq 0 \bmod p$ for all signing queries m. As $0 \leq \kappa\tau < p$ and $x' + \sum_{i=1}^{n} m_i x_i < \tau(n+1) < p$, we note that $J(m) = 0 \bmod p$ implies $J(m) = 0 \bmod \tau$ (and thus $J(m) \neq 0 \bmod \tau$ implies $J(m) \neq 0 \bmod p$). Hence, to simplify the analysis, we force $\mathcal{B}$ to abort whenever $J(m) = 0 \bmod \tau$ in a signing query. Besides, $\mathcal{B}$ is successful if the target message satisfies $J(m^\star) = 0 \bmod p$.

More formally, let $m_1', \ldots, m_{q_s}'$ be messages appearing in signing queries and let us define events $A_i : J(m_i') \neq 0 \bmod \tau$ and $A^\star : J(m^\star) = 0 \bmod p$, the probability that $\mathcal{B}$ does not fail is

$$\Pr[\neg\mathsf{abort}] \geq \Pr[\bigwedge_{i=1}^{q_s} A_i \wedge A^*].$$

As $J(m^\star) = 0 \bmod p$ implies $J(m^\star) = 0 \bmod \tau$ and given that, if $J(m^\star) = 0 \bmod \tau$, there is a unique $\kappa \in \{0, \ldots, n\}$ that yields $J(m^\star) = 0 \bmod p$, we have

$$\Pr[A^\star] = \Pr[J(m^\star) = 0 \bmod \tau]\Pr[J(m^*) = 0 \bmod p | J(m^\star) = 0 \bmod \tau] = \frac{1}{\tau}\frac{1}{n+1}.$$

Moreover,
$$\Pr[\bigwedge_{i=1}^{q_s} A_i | A^\star] = 1 - \sum_{i=1}^{q_s} \Pr[\neg A_i | A^\star] = 1 - \frac{q_s}{\tau},$$

where the rightmost equality stems from the independence of A_i and $A^\star$ for any i (hence $\Pr[\neg A_i | A^\star] = 1/\tau$). Putting the above together, we obtain

$$\Pr[\neg \mathsf{abort}] = \Pr[A^\star]\Pr[\bigwedge_{i=1}^{q_s} A_i | A^*] = \frac{1}{\tau(n+1)}\Big(1 - \frac{q_s}{\tau}\Big) = \frac{1}{4q_s(n+1)}$$

thanks to the choice of $\tau = 2q_s$. Since $\mathcal{B}$ correctly guesses the index $i^\star$ of the attacked stage with probability higher than $1/N$, the claimed bound follows. □

4 Intrusion-resilient signatures without random oracles

In [16], Dodis *et al.* showed how to generically obtain intrusion-resilient schemes from key-evolving cryptosystems where the key update algorithm satisfies a suitable homomorphic property. Although they proved the security of their conversion in the context of encryption schemes, their proof simply goes through for digital signatures (as detailed in the full version of the paper).

At a high level, the idea is to share the update key of a key-evolving signature between the signer and the base so that the sharing for period $i+1$ can be derived from that of period i. At stage i, the signer stores the local key $\overline{lsk_i}$ (used in signing operations) and his share of the update key $\overline{sks_i}$ while the base stores the other share $\overline{skb_i}$. This sharing ensures security against multiple compromises of base and user keys. When each share of the update key is independently evolved (using the update algorithm of the key-evolving scheme) at period i, shares of update and local keys for period $i+1$ are obtained. The update message SKU_i sent by the base is its evolved share of the local key. Thanks to the homomorphic property of the update algorithm, the signer can combine $\overline{SKU_i}$ with his own share of the evolved local key to reconstruct the local key $\overline{lsk_{i+1}}$.

In our notation, when two vectors $\mathsf{sk}_{s.w} = (a_0, a_1, b_{k+1}, \ldots, b_\ell)$ and $\mathsf{sk}_{b.w} = (a'_0, a'_1, b'_{k+1}, \ldots, b'_\ell)$ are node keys at level k in the hierarchy, we denote by $\mathsf{sk}_{s.w} \odot \mathsf{sk}_{b.w}$ the component-wise product $(a_0 \cdot a'_0, a_1 \cdot a'_1, b_{k+1} \cdot b'_{k+1}, \ldots, b_\ell \cdot b'_\ell)$.

Keygen: given security parameters $\lambda, n \in \mathbb{N}$ and a number of periods $N = 2^{\ell-1}$,

1. choose bilinear map groups $(\mathbb{G}, \mathbb{G}_T)$ and $g \in \mathbb{G}$. Set $g_1 = g^\alpha$ with $\alpha \xleftarrow{R} \mathbb{Z}_p^*$.
2. Pick $g_2, g_3, h_1, \ldots, h_\ell \xleftarrow{R} \mathbb{G}$, $u', u_1, \ldots, u_n \xleftarrow{R} \mathbb{G}$ and set $Z = e(g_1, g_2)$.
3. Select a collision-resistant hash function $h : \{0,1\}^* \to \{0,1\}^n$.
4. Define functions $F : \{0,1\}^{\leq \ell} \to \mathbb{G}$, $G : \{0,1\}^n \to \mathbb{G}$ as
$$F(w) = g_3 \cdot \prod_{j=1}^{k} h_j^{w_j} \qquad G(m) = u' \cdot \prod_{j=1}^{n} u_j^{m_j}$$
where $w = w_1 \ldots w_k$ and $m = m_1 \ldots m_n$ ($w_i, m_j \in \{0,1\}$ for all i, j). The public key is

$$PK = \{g, Z, g_1, g_2, g_3, h_1, \ldots, h_\ell, u', u_1, \ldots, u_n\}.$$

The root secret $\mathsf{sk}_\varepsilon = g_2^\alpha$ is used to derive lower level node keys.

5. Extract node keys for labels 0 and 1 at level 1. Namely, compute $\mathsf{sk}_0 = \big(g_2^\alpha g_3^{r_0}, g^{r_0}, h_2^{r_0}, \ldots h_\ell^{r_0}\big)$, $\mathsf{sk}_1 = \big(g_2^\alpha (g_3 h_1)^{r_1}, g^{r_1}, h_2^{r_1}, \ldots h_\ell^{r_1}\big)$ for random $r_0, r_1 \overset{R}{\leftarrow} \mathbb{Z}_p^*$. Using sk_0, recursively apply algorithm $\mathsf{Extract}$ (defined below) to generate keys sk_{01}, sk_{001}, $\ldots \mathsf{sk}_{0^{\ell-1}1}$. Consider the key set $S_{\langle 0\rangle} = \{\mathsf{sk}_1, \mathsf{sk}_{01}, \mathsf{sk}_{001}, \ldots, \mathsf{sk}_{0^{\ell-1}1}\}$. For each $\mathsf{sk}_w \in S_{\langle 0\rangle}$, define $\mathsf{sk}_{b.w} = \mathsf{sk}_w \odot (R_w^{-1}, 1, 1, ..., 1)$ and $\mathsf{sk}_{s.w} = \mathsf{sk}_w \odot (R_w, 1, 1, ..., 1)$ with $R_w \overset{R}{\leftarrow} \mathbb{G}$.
6. Base and signer keys are $SKB_{0.0} = \{\mathsf{sk}_{b.1}, \mathsf{sk}_{b.01}, \mathsf{sk}_{b.001}, \ldots, \mathsf{sk}_{b.0^{\ell-1}1}\}$ and $SKS_{0.0} = (-, \{\mathsf{sk}_{s.1}, \mathsf{sk}_{s.01}, \mathsf{sk}_{s.001}, \ldots, \mathsf{sk}_{s.0^{\ell-1}1}\})$.

Extract $(\mathsf{sk}_{\star.w_1\ldots w_{k-1}})$: (where $\star = s$ or $\star = b$) is an auxiliary algorithm used by the base and the signer to derive node private keys. A node at level $k-1$ parses its private key into $\mathsf{sk}_{\star.w_1\ldots w_{k-1}} = (a_0, a_1, b_k, \ldots, b_\ell)$. For $j = 0, 1$, it chooses a random $t_j \overset{R}{\leftarrow} \mathbb{Z}_p^*$ and computes

$$\mathsf{sk}_{\star.w_1\ldots w_{k-1}j} = \Big(a_0 \cdot b_k^j \cdot F(w_1 \ldots w_{k-1}j)^{t_j}, a_1 \cdot g^{t_j}, b_{k+1} \cdot h_{k+1}^{t_j}, \ldots, b_\ell \cdot h_\ell^{t_j}\Big).$$

Base Update $(SKB_{i.r}, i+1)$: (where $i < N-1$)

1. Parse $\langle i\rangle$ as $i_0 i_1 \ldots i_\ell$ with $i_0 = \varepsilon$. Let $SKB_{i.r} = \{\mathsf{sk}_{b.i_0\ldots i_{k-1}1}\}_{i_k=0}$.
2. If $i_\ell = 0$, the update message SKU_i is the share $\mathsf{sk}_{b.i_0\ldots i_{\ell-1}1}$ and the updated base key is $SKB_{i+1.0} = \{\mathsf{sk}_{b.i_0\ldots i_{k-1}1}\}_{i_k=0, k<\ell}$. Otherwise, let $\tilde{k} < \ell$ be the largest index s.t. $i_{\tilde{k}} = 0$. Let $i' = i_0 \ldots i_{\tilde{k}-1}1$. Using $\mathsf{sk}_{b.i'}$ (available as part of $SKB_{i.r}$), recursively apply $\mathsf{Extract}$ to obtain $\mathsf{sk}_{b.i'1}, \mathsf{sk}_{b.i'01}, \ldots, \mathsf{sk}_{b.i'0^{\ell-\tilde{k}-1}1}$ and $\mathsf{sk}_{b.i'0^{\ell-\tilde{k}}}$ which is the update message SKU_i. Erase $\mathsf{sk}_{b.i'}$ and return $SKU_i = \mathsf{sk}_{b.\langle i+1\rangle}$ and

$$SKB_{i+1.0} = \{\{\mathsf{sk}_{b.i_0\ldots i_{k-1}1}\}_{i_k=0,\ k<\tilde{k}}, \mathsf{sk}_{b.i'1}, \mathsf{sk}_{b.i'01}, \ldots, \mathsf{sk}_{b.i'0^{\ell-\tilde{k}-1}1}\}.$$

User Update $(SKS_{i.r}, SKU_i, i+1)$: (with $i < N-1$)

1. Let $\langle i\rangle = i_0 i_1 \ldots i_\ell$ with $i_0 = \varepsilon$. Parse $SKS_{i.r}$ as $\big(\mathsf{sk}_{\langle i\rangle}, \{\mathsf{sk}_{s.i_0\ldots i_{k-1}1}\}_{i_k=0}\big)$ and erase $\mathsf{sk}_{\langle i\rangle}$.
2. If $i_\ell = 0$, then set $SKS_{i+1.0} = \big(\mathsf{sk}_{\langle i+1\rangle}, \{\mathsf{sk}_{s.i_0\ldots i_{k-1}1}\}_{i_k=0,k<\ell}\big)$ with $\mathsf{sk}_{\langle i+1\rangle} = \mathsf{sk}_{s.i_0\ldots i_{\ell-1}1} \odot SKU_i$. Otherwise, let $\tilde{k}$ be the largest value with $i_{\tilde{k}} = 0$ and let $i' = i_0 \ldots i_{\tilde{k}-1}1$. Using $\mathsf{sk}_{s.i'}$ (available in $SKS_{i.r}$), apply $\mathsf{Extract}$ to obtain $\mathsf{sk}_{s.i'1}, \mathsf{sk}_{s.i'01}, \ldots, \mathsf{sk}_{s.i'0^{\ell-\tilde{k}-1}1}$ and $\mathsf{sk}_{s.i'0^{\ell-\tilde{k}}}$ which allows reconstructing $\mathsf{sk}_{\langle i+1\rangle} = \mathsf{sk}_{s.i'0^{\ell-\tilde{k}}} \odot SKU_i$. Set $SKS_{i+1.0}$ as

$$\big(\mathsf{sk}_{\langle i+1\rangle}, \{\{\mathsf{sk}_{s.i_0\ldots i_{k-1}1}\}_{i_k=0,\ k<\tilde{k}}, \mathsf{sk}_{s.i'1}, \mathsf{sk}_{s.i'01}, \ldots, \mathsf{sk}_{s.i'0^{\ell-\tilde{k}-1}1}\}\big)$$

and erase $\mathsf{sk}_{s.i'}$.

Base Refresh $(SKB_{i.r})$: let $\langle i\rangle = i_0, \ldots i_\ell$ and $SKB_{i.r} = \{\mathsf{sk}_{b.i_0\ldots i_{k-1}1}\}_{i_k=0}$. For all keys $\mathsf{sk}_{b.w}$ in $SKB_{i.r}$, let $\mathsf{sk}'_{b.w} = \mathsf{sk}_{b.w} \odot (R_w^{-1}, 1, \ldots, 1)$ for a random $R_w \overset{R}{\leftarrow} \mathbb{G}$. Return $SKB_{i.r+1} = \{\mathsf{sk}_{b.w'} | \mathsf{sk}_{b.w} \in SKB_{i.r}\}$ together with the refresh message $SKR_{i.r} = \{R_w | \mathsf{sk}_{b.w} \in SKB_{i.r}\}$.

User Refresh $(SKS_{i.r}, SKR_{i.r})$: let $SKS_{i.r} = \big(\mathsf{sk}_{\langle i \rangle}, \{\mathsf{sk}_{s.i_0 \dots i_{k-1}1}\}_{i_k=0}\big)$ and $SKR_{i.r} = \{R_w | \mathsf{sk}_{s.w} \in \{\mathsf{sk}_{s.i_0 \dots i_{k-1}1}\}_{i_k=0}\}$ with $\langle i \rangle = i_0, \dots i_\ell$. For all shares of node keys $\mathsf{sk}_{s.w} \in \{\mathsf{sk}_{s.i_0 \dots i_{k-1}1}\}_{i_k=0}$, set $\mathsf{sk}_{s.w'} = \mathsf{sk}_{s.w} \odot (R_w, 1, \dots, 1)$ and return $SKS_{i.r+1} = \big(\mathsf{sk}_{\langle i \rangle}, \{\mathsf{sk}_{s.w'} | \mathsf{sk}_{s.w} \in \{\mathsf{sk}_{s.i_0 \dots i_{k-1}1}\}_{i_k=0}\}\big)$.

Sign $(i, SKS_{i.r}, M)$: let $\langle i \rangle = i_1 \dots i_\ell$. Parse $SKS_{i.r}$ into $\big(\mathsf{sk}_{\langle i \rangle}, \{\mathsf{sk}_{i_0 \dots i_{k-1}1}\}_{i_k=0}\big)$ and $\mathsf{sk}_{\langle i \rangle}$ as (a_0, a_1). Compute $m = h(M)$, choose $s \overset{R}{\leftarrow} \mathbb{Z}_p^*$ and return

$$(\sigma_0, \sigma_1, \sigma_2) = (a_0 \cdot G(m)^s, a_1, g^s) = (g_2^\alpha \cdot F(i_1 \dots i_\ell)^r \cdot G(m)^s, g^r, g^s)\,.$$

Verify (M, i, PK, σ) : let $\langle i \rangle = i_1 \dots i_\ell$ and $m = h(M) \in \{0,1\}^n$. The signature $\sigma = (\sigma_0, \sigma_1, \sigma_2)$ is accepted if and only if

$$\frac{e(\sigma_0, g)}{e(F(i_1 \dots i_\ell), \sigma_1) e(G(m), \sigma_2)} = Z.$$

This scheme is as efficient as the key-evolving scheme of section 3 in terms of computational cost as well as signature/key sizes. In particular, it features private keys of size $O(\ell^2)$. Public keys also have logarithmic size but, for realistic values of ℓ, the bulk of their length is the string $u', u_1, \dots, u_n$ borrowed from Waters's signature scheme.

It is natural to compare our scheme with the one obtained by applying the generic construction of [27] to Waters's signature since assumptions of comparable strength are needed for the security of both schemes. It turns out that the generic method allows for faster verification but results in signatures of prohibitive size (up to 10^6 bits for $N \leq 2^{10}$) for many practical applications. Besides, it does not yield shorter public keys as the large string $u', u_1, \dots, u_n$ should remain part of the public key to avoid including it within each signature. We concede that [27] allows for faster key updates (which take constant time if we neglect the time for verifying $\log N$ one-time signatures). However, our construction should be preferred for all applications where signature sizes are primary concern.

5 Conclusion

We proposed the first random oracle-free intrusion-resilient signature with short constant-size signatures and at most log-squared complexity in all other parameters. It was built on Waters's signature by a relative public key lengthening which is quite moderate for any realistic number of time periods.

As an intermediate result, we described a new forward-secure signature without random oracles which is of independent interest. Thanks to the "homomorphic" properties of Waters's signature, this new efficient scheme can be extended into a very efficient forward-secure threshold signature (following ideas of Abdalla *et al.* [1]) where no communication is required between servers during the signing protocol. To the best of our knowledge, such a protocol did not previously exist in the standard model.

References

1. M. Abdalla, S. K. Miner, C. Namprempre. Forward-Secure Threshold Signature Schemes. In *CT-RSA'01*, volume 2020 of *LNCS*, pages 441–456, Springer, 2001.
2. M. Abdalla, L. Reyzin. A New Forward-Secure Digital Signature Scheme. In *Asiacrypt'00*, *LNCS* 1666, pp. 116–129. Springer, 1999.
3. R. Anderson. Two Remarks on Public Key Cryptology. Invited lecture, *ACM Conference on Computer and Communications Security*, 1997.
4. P. S. L. M. Barreto, M. Naehrig. Pairing-friendly elliptic curves of prime order. In *SAC'05*. volume 3897 of *LNCS*, pages 319–331. Springer, 2006.
5. M. Bellare, S. Miner. A Forward-Secure Digital Signature Scheme. In *Crypto'99*, *LNCS* 1666, pp. 431–448. Springer, 1999.
6. M. Bellare, P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *1st ACM Conference on Computer and Communications Security*, pages 62–73, ACM Press, 1993.
7. D. Boneh, X. Boyen. Short signatures without random oracles. In *Eurocrypt'04*, volume 3027 of *LNCS*, pages 56–73. Springer, 2004.
8. D. Boneh, X. Boyen, E.-J. Goh. Hierarchical Identity Based Encryption with Constant Size Ciphertext. In *Eurocrypt'05*, *LNCS* 3494, pp. 440–456. Springer, 2005.
9. D. Boneh, M. Franklin. Identity-based encryption from the Weil pairing. In *Crypto'01*, *LNCS* 2139, pp. 213–229. Springer, 2001.
10. X. Boyen, H. Shacham, E. Shen, B. Waters. Forward-Secure Signatures with Untrusted Update. In *ACM CCS'06*, ACM Press, 2006.
11. J. Camenisch, M. Koprowski. Fine-grained forward-secure signature schemes without random oracles. Discrete Applied Mathematics 154(2), pages 175–188, 2006.
12. R. Canetti, O. Goldreich, S. Halevi. The random oracle methodology, revisited. Journal of the ACM 51(4), pages 557–594, 2004.
13. R. Canetti, S. Halevi, J. Katz. A forward secure public key encryption scheme. In *Eurocrypt'03*, volume 2656 of *LNCS*, pages 254–271. Springer, 2003.
14. S. S. Chow, L. C. Kwong Hui, S. M. Yiu, K. P. Chow. Secure Hierarchical Identity Based Signature and Its Application. In *ICICS'04*, volume 3269 of *LNCS*, pages 480–494, Springer, 2004.
15. Y. Dodis, M. Franklin, J. Katz, A. Miyaji, M. Yung. Intrusion-Resilient Public-Key Encryption. In *CT-RSA'03*, volume 2612 of *LNCS*, pages 19–32. Springer, 2003.
16. Y. Dodis, M. Franklin, J. Katz, A. Miyaji, M. Yung. A Generic Construction for Intrusion-Resilient Public-Key Encryption. In *CT-RSA'04*, volume 2964 of *LNCS*, pages 81–98. Springer, 2004.
17. Y. Dodis, J. Katz, S. Xu, M. Yung. Key-Insulated Public Key Cryptosystems. In *Eurocrypt'02*, volume 2332 of *LNCS*, pages 65–82. Springer, 2002.
18. Y. Dodis, J. Katz, S. Xu, M. Yung. Strong key-insulated signature schemes. In *PKC'03*, volume 2567 of *LNCS*, pages 130–144. Springer, 2003.
19. C. Gentry, A. Silverberg. Hierarchical ID-based cryptography. In *Asiacrypt'02*, volume 2501 of *LNCS*, pages 548–566. Springer, 2002.
20. S. Goldwasser, S. Micali, R. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIAM J. Comput.* 17(2), pages 281–308, 1988.
21. L. Guillou, J.-J. Quisquater. A "paradoxical" identity-based signature scheme resulting from zero-knowledge. In *Crypto'88*, volume 0403 of *LNCS*, pages 216–231. Springer, 1988.

22. R. Granger, N. P. Smart. On Computing Products of Pairings. Cryptology ePrint Archive: Report 2006/172, 2006.
23. A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, M. Yung. Proactive Public Key and Signature Systems. In 4th *ACM Conference on Computer and Communication Security*, pages 100-110, *ACM* Press, 1997.
24. F. Hu, Ch.-H. Wu, J. D. Irwin. A New Forward Secure Signature Scheme using Bilinear Maps. Cryptology ePrint Archive: Report 2003/188, 2003.
25. G. Itkis, L. Reyzin. Forward-Secure Signatures with Optimal Signing and Verifying. In *Crypto'01*, volume 2139 of *LNCS*, pages 332–354, Springer, 2001.
26. G. Itkis, L. Reyzin. SiBIR: Signer-Base Intrusion-Resilient Signatures. In *Crypto'02*, volume 2442 of *LNCS*, pages 499–514, Springer, 2002.
27. G. Itkis. Intrusion-Resilient Signatures: Generic Constructions, or Defeating Strong Adversary with Minimal Assumptions. In *SCN'02*, volume 2576 of *LNCS*, pages 102–118, Springer, 2003.
28. G. Itkis. Forward Security: Adaptive Cryptography - Time Evolution. Invited chapter for the Handbook of Information Security, John Wiley and Sons, 2004.
29. B. G. Kang, J. H. Park, S. G. Hahn. A New Forward Secure Signature Scheme. Cryptology ePrint Archive: Report 2004/183, 2004.
30. J. Katz. A Forward-Secure Public-Key Encryption Scheme. Cryptology ePrint Archive: Report 2002/060, 2002.
31. L. Lamport. Constructing Digital Signatures from a One-Way Function. Technical Report CSL-98. Sri Internation, 1979.
32. T. Malkin, D. Micciancio, S. K. Miner. Efficient Generic Forward-Secure Signatures with an Unbounded Number Of Time Periods. In *Eurocrypt'02*, volume 2332 of *LNCS*, pages 400–417, Springer, 2002.
33. T. Malkin, S. Obana, M. Yung. The Hierarchy of Key Evolving Signatures and a Characterization of Proxy Signatures. In *Eurocrypt'04*, volume 3027 of *LNCS*, pages 306–322, Springer, 2004.
34. M. Mambo, K. Usuda, E. Okamoto. Proxy signatures for delegating signing operation. In 3rd *ACM Conference on Computer and Communications Security*, pages 48–57, ACM Press, 1996.
35. A. Miyaji, M. Nakabayashi, S. Takano. New explicit conditions of elliptic curve traces for FR-reduction. *IEICE Transactions on Fundamentals*, E84-A(5):1234–1243, 2001.
36. R. Ostrovsky, M. Yung. How to Withstand Mobile Virus Attacks. In 10th *ACM Symp. on Principles of Distributed Computing*, pages 51–59, 1991.
37. K. G. Paterson, J. C. N. Schuldt. Efficient Identity-based Signatures Secure in the Standard Model. In *ACISP'06*, volume 4058 of *LNCS*, pages 207–222, Springer, 2006.
38. A. Shamir. Identity based cryptosystems and signature schemes. In *Crypto'84*, volume 196 of *LNCS*, pages 47–53. Springer, 1984.
39. B. Waters. Efficient Identity-Based Encryption Without Random Oracles. In *Eurocrypt'05*, volume 3494 of *LNCS*, pages 114–127. Springer 2005.
40. F. Zhang, R. Safavi-Naini, W. Susilo. An efficient signature scheme from bilinear pairings and its applications. In *PKC'04*, volume 2947 of *LNCS*, pages 277–290. Springer, 2004.