A New Forward Secure Signature Scheme

Bo Gyeong Kang[1], Je Hong Park[2], and Sang Geun Hahn[1]

[1] Department of Mathematics, Korea Advanced Institute of Science and Technology,
373-1 Guseong-dong, Yuseong-gu, Daejeon, 305-701, Korea
{snubogus,sghahn}@kaist.ac.kr
[2] National Security Research Institute,
161 Gajeong-dong, Yuseong-gu, Daejeon, 305-345, Korea
jhpark@etri.re.kr

Abstract. In this paper, we present two forward secure signature schemes based on gap Diffie-Hellman groups and prove these schemes to be secure in the sense of slightly stronger security notion than that by Bellare and Miner in the random oracle model. Both schemes use the same key update strategy as the encryption scheme presented by Canetti, Halevi and Katz. Hence, our schemes outperform the previous tree-based forward secure signature scheme by Bellare and Miner in the key generation and key update time, which are only constant in the number of time periods. Specifically, we describe a straightforward scheme following from the encryption scheme, and then improve its efficiency for signature verification algorithm which needs only 3 pairing computations independent of the total time periods.

1 Introduction

Ordinary digital signatures have a fundamental limitation: if the secret key of a signer is compromised, all the signatures of that signer become worthless. As portable devices are spreading widely, it is required that a number of cryptographic computations are performed on those devices. But, these devices are usually unprotected and easily-stolen, so careless usages allow that an adversary gains access to a secret key without breaking the underlying cryptographic schemes. To mitigate the damage caused by exposure of secret keys stored on such devices, cryptographic schemes are basically required to meet the notion of *forward security*. This assumes that key exposure will inevitably occur and seeks instead to minimize the damage. The lifetime of the protocol is divided into distinct periods $0, \ldots, N-1$ and secret keys are updated from the current one at the end of each time period via a key update algorithm, while the public key remains for the lifetime of the scheme. So exposure of the secret key corresponding to a given time period does not enable an adversary to break the scheme for any *prior* time period. This notion was first proposed in the context of key-exchange protocols [12] and then adapted on signature/identification schemes.

Forward security was first formalized in the context of signature and identification scheme by Bellare and Miner [3], building on earlier ideas of Anderson [2]. Subsequently, numerous constructions of forward secure signature schemes have been proposed [1, 18, 14, 19], but a forward secure encryption scheme has been constructed recently by Canetti, Halevi and Katz [6]. It is based on the binary tree encryption scheme [17], that is a relaxed variant of the hierarchical identity-based encryption scheme of Gentry and Silverberg [11]. In this scheme, each of time periods is associated with a node in a binary tree of depth $\lceil \log N \rceil$ according to a pre-order traversal. So the scheme has all parameters at most poly-logarithmic and especially constant key generation and key update time in the number of time periods. Recently, Hu, Wu and Irwin [13] independently presented a forward secure signature scheme parallels to the encryption scheme of Katz [16][3]. However this scheme obviously uses the same strategy of binary certification tree scheme suggested by Bellare and Miner [3] which associates time periods with the leaves only.

Related works There are many alternate approaches to address the risks associated with key exposure. Especially, to obtain security for time periods following key exposure, the notions of key-insulation [4, 9, 10] and intrusion-resilience [15, 7, 8] were proposed. In these schemes, not only past but also future secret keys remain secure in case the current secret key is compromised. So the security provided by these schemes is better. However, those solutions require time synchronization and the signer's communication with a safe computing device for each time period. In the case that the user is self-sufficient and need not interact with any other device, forward secure schemes are advantageous for efficiency. Recently Malkin, Obana and Yung [20] established a hierarchy for key evolving signature schemes. They showed that above two security notions are equivalent and imply forward-security.

Our contribution First, we construct a forward secure signature scheme $\mathsf{FS.PKS}$ which use the same system parameters and key update algorithm as the encryption scheme presented by Canetti, Halevi and Katz [6]. By modifying the key update algorithm of $\mathsf{FS.PKS}$, we obtain an efficient forward secure signature scheme $\mathsf{EFS.PKS}$. It is obvious that our schemes maintain most of the advantages of the corresponding encryption scheme due to the same structure. Our schemes achieve better performance than the binary certification tree scheme by Bellare and Miner[3] with respect to the key generation and key update algorithm. One point that differentiates $\mathsf{EFS.PKS}$ from $\mathsf{FS.PKS}$ is signature verification algorithm. $\mathsf{EFS.PKS}$ needs only 3 pairing computations in each verification procedure at the cost of two field multiplications in each key update procedure, beating $\log N$ pairing computations of $\mathsf{FS.PKS}$. That is very remarkable success because the pairing computation is the most expensive procedure.

Now, we compare the full performance of our schemes with Bellare-Miner scheme in the following Table 1. We consider a supersingular curve and a modified pairing on it as a GDH group and a bilinear map, respectively. We record the cost in terms of the number of scalar multiplications and pairing computations in the group. For concrete comparison, we refer to [13] for a concrete example following the Bellare-Miner scheme.

Additionally, we prove that our concrete schemes satisfy a slightly stronger security notion rather than those in [1].

Due to recent results of [21] and [20], we can obtain several forward secure signature schemes induced from identity-based signature schemes. But these are so complex to understand their structure and so only valuable for generic construction.

[3] It is an older version of the paper [6].

	[13](BM type)	FS.PKS	EFS.PKS
Key gen time	$O(\log N)S$	$O(1)S$	$O(1)S$
Key update time	$O(\log N)S$	$O(1)S$	$O(1)S$
Signing time	$O(1)S$	$O(1)S$	$O(1)S$
Verifying time	$O(\log N)P$	$O(\log N)(S+P)$	$O(\log N)S + 3P$

Table 1. Performance (S: scalar multiplication, P: pairing computation)

Organization In Section 2, we introduce related mathematical problems, and review the formal definition of the forward secure signature scheme and its security. In Section 3, we present our new signature scheme and give a detailed security proof of it.

2 Preliminaries

In this section, we briefly introduce several mathematical backgrounds and then review the formal definition of the forward secure signature schemes [3, 1].

2.1 Cryptographic assumptions

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic groups of prime order q, where $\mathbb{G}_1$ is represented additively and $\mathbb{G}_2$ is represented multiplicatively. And let $P \in \mathbb{G}_1$ be a generator of $\mathbb{G}_1$.

- Computation Diffie-Hellman (CDH) problem: Given (P, aP, bP) where a, $b \in \mathbb{Z}_q^*$, compute abP. The advantage of an algorithm $\mathcal{A}$ in solving the CDH problem in a group $\mathbb{G}$ is

$$\mathsf{AdvCDH}_{\mathcal{A}} = \Pr[\mathcal{A}(P, aP, bP) = abP \mid a, b \xleftarrow{r} \mathbb{Z}_q^*].$$

  We say that $\mathcal{A}$ (t, ϵ)-breaks CDH in $\mathbb{G}$ if $\mathcal{A}$ runs in time at most t, and $\mathsf{AdvCDH}_A \geq \epsilon$.
- Decision Diffie-Hellman (DDH) problem: Given (P, aP, bP, cP) where a, b, $c \in \mathbb{Z}_q^*$, decide whether $c = ab$ in $\mathbb{Z}_q^*$. If so, (P, aP, bP, cP) is called a valid Diffie-Hellman tuple.

Definition 1. A prime order group $\mathbb{G}$ is a (t, ϵ)-GDH group if DDH problem can be solved in polynomial time but no probabilistic algorithm (t, ϵ)-breaks CDH in $\mathbb{G}$.

A bilinear map is an efficiently computable map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ between these two groups must satisfy the following properties:

1. Bilinear: For all P, $Q \in \mathbb{G}_1$ and all a, $b \in \mathbb{Z}$, $e(aP, bQ) = e(P, Q)^{ab}$.
2. Non-degenerate: The map does not send all pairs in $\mathbb{G}_1 \times \mathbb{G}_1$ to the identity in $\mathbb{G}_2$. Observe that since $\mathbb{G}_1$, $\mathbb{G}_2$ are groups of prime order, this implies that if P is a generator of $\mathbb{G}_1$, then $e(P, P)$ is a generator of $\mathbb{G}_2$.

Specifically, the Weil and Tate pairings are practical example of the bilinear map. Using the Weil or Tate pairing, certain elliptic curves (for example, supersingular curves) can be used as GDH groups.

A *GDH parameter generator* $\mathcal{IG}$ is a randomized algorithm that takes a security parameter $k \in \mathbb{N}$, runs in time polynomial in k, and outputs the description of two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of the same prime order q and the description of an admissible pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. We say that $\mathcal{IG}$ satisfies the *GDH assumption* if the following probability is negligible (in k) for all PPT algorithm $\mathcal{A}$:

$$\Pr[\mathcal{A}(\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, aP, bP) = abP \,|\, (\mathbb{G}_1, \mathbb{G}_2, \hat{e}) \leftarrow \mathcal{IG}(1^k), P \leftarrow \mathbb{G}_1^*, a, b \leftarrow \mathbb{Z}_q^*].$$

2.2 A key evolving signature scheme

The approach taken by forward secure signature schemes is to update the secret key periodically. Thus a forward secure signature scheme is a key-evolving signature scheme which is a regular signature scheme with the additions of time periods and the key update algorithm to vary the secret key. However, the public key is left unchanged throughout the lifetime of the scheme. The following definitions of forward security are taken from [3, 1] with slight modification.

Definition 2. A *key-evolving signature scheme* KE.PKS consists of the following four algorithms:

1. KE.GEN, the probabilistic *key generation* algorithm, takes as input a security parameter $k \in \mathbb{N}$, the total number of period N. It returns a *base public key* PK and corresponding *base secret key* SK_0.
2. KE.UPD, the probabilistic *secret key update* algorithm, takes as input the secret signing key SK_{j-1} of the previous period. It returns the secret signing key SK_j of the current period.
3. KE.SIGN, the probabilistic *signing* algorithm, takes as input the secret signing key SK_j of the current period and a message M. It returns a signature of M for period j. We write $\langle j, \sigma\rangle \stackrel{R}{\leftarrow} \mathsf{KE.SIGN}_{SK_j}(M)$.
4. KE.VRFY, the deterministic *verification* algorithm, takes as input the public key PK, a message M and a candidate signature $\langle j, \sigma\rangle$ and outputs a binary value 0 (invalid) or 1 (valid).

We say that $\langle j, \sigma\rangle$ is a *valid* signature of M for period j if $\mathsf{KE.VRFY}(M, \langle j, \sigma\rangle) = 1$. It is required that a signature of M generated via $\mathsf{KE.SIGN}_{SK_j}(M)$ be a valid signature of M for period j.

Basically, what a key evolving signature scheme is forward secure means that an adversary which obtains the current secret key is computationally infeasible to forge a signature with respect to a previous secret key. In this paper, we consider a slightly stronger security notion rather than those defined in [3]: Previously, the adversary is considered successful if it can create a valid forgery $(M, \langle b, \sigma\rangle)$ for some $b < j$ and the *new* message M, where j is the current period. We require that the adversary cannot even make a different signature on previously signed message in period b.

Furthermore, we consider the time period starting at zero. That is, even if the adversary is not allowed to get the base secret key, it can have access the signing oracle during zero period. Hence, a signature forgery for time period 0 is considered as a valid one. Formally, this adversary is modeled via Experiment 1 in the random oracle model.

Experiment 1 $\mathsf{FS.PKS}_{\mathcal{A}}(\mathsf{KE.PKS}(k, N), \mathcal{A})$

Select H at random.
$(PK, SK_0) \xleftarrow{R} \mathsf{KE.GEN}(k, N)$
$j \leftarrow 0$
$0 \leftarrow \mathcal{A}^{\mathsf{KE.SIGN}^{H(\cdot)}(SK_j, \cdot)}(PK, j)$
repeat
  $j \leftarrow j + 1$
  $SK_j \leftarrow \mathsf{KE.UPD}(SK_{j-1}, j)$
  $d \leftarrow \mathcal{A}^{\mathsf{KE.SIGN}^{H(\cdot)}(SK_j, \cdot)}(PK, j)$
until $d = \mathsf{breakin}$ or $j = N$
if $j = N$ and $d = 0$ **then**
  sets SK_{N+1} with empty strings.
end if
$(M, (b, \sigma)) \leftarrow \mathcal{A}^{H(\cdot)}(SK_j)$
if $\mathsf{KE.VRFY}(PK, M, \langle b, \sigma \rangle) = 1$ and $0 \leq b < j$ where σ is distinct from any previously given signature (if exists) on M by $\mathsf{KE.SIGN}(SK_b, \cdot)$ **then**
  return 1
else
  return 0
end if

In this model, an adversary knows the public key PK, the total number of time periods N and the current time period j. For the key-evolving signature scheme KE.PKS, the adversary $\mathcal{A}$ is viewed as functioning in three stages. Basically, we view hash function H as a random oracle. In the first phase, the chosen message attack phase (cma), the adversary has access to a signing oracle, which it can query to obtain signatures of messages of its choice with respect to the current secret key. Without loss of generality of the concept of forward security, we allow the adversary accesses to the signing oracle made by the base secret key. At the end of each time period, the adversary can choose whether to stay in the same phase or switch to the break in (breakin) phase. It cannot get access to previous oracle again. In the breakin phase, which models the possibility of a key exposure, we give the adversary the secret key SK_j for the specific time period j it decided to break in. In the last phase, the forgery phase (forge), the adversary outputs a signature forgery $\langle M, \sigma \rangle$ in period b. The adversary is considered to be successful if $\mathsf{KE.VRFY}(M, \langle b, \sigma \rangle) = 1$, $0 \leq b < j$ hold and σ is distinct from any previously given signature (if exists) on the message M in period b. So the advantage of an adversary $\mathsf{AdvKE.PKS}_{\mathcal{A}}$ is defined as the probability that $\mathcal{A}$ is successful in Experiment 1.

Definition 3. We say that an algorithm $\mathcal{A}$ (t, q_S, q_H, ϵ)-attacks KE.PKS if $\mathcal{A}$ runs in time at most t and at most q_S and q_H queries are made to the signing oracle and the random oracle H, respectively, and then $\mathsf{AdvKE.PKS}_{\mathcal{A}} \geq \epsilon$. We say that the key-evolving signature scheme

$\mathsf{KE.PKS}$ is (t, q_S, q_H, ϵ)-*forward secure* against chosen message attacks in the random oracle if there is no adversary that (t, q_S, q_H, ϵ)-attacks $\mathsf{KE.PKS}$, and denote it by $\mathsf{KE.PKS}(t, q_S, q_H, \epsilon)$.

3 A new forward secure signature scheme

3.1 Notations

Our forward secure signature scheme employs the binary tree structure which is a variant of the tree structure used in the hierarchical identity-based signature scheme [11]. If we use a full binary tree with depth l, then the number of time periods is $N = 2^{l+1} - 1$. The root of the tree is called node ε. By the well known pre-order traversal technique [6] of binary trees, we can assign nodes to time periods.

Denote the node (represented by bit string) and its secret key corresponding to the time period i by w^i and S_{w^i}, respectively. Let $w^i0\,(w^i1)$ be the left (right) child node and let $w^i|_k$ be a k-prefix of w^i. Let $w|_{\bar{k}}$ be the sibling node of $w|_k$. Pre-order traversal can be defined as follows: $w^0 = \varepsilon$ is the root node, and if w^i is an internal node, then $w^{i+1} = w^i0$. If w^i is a leaf node and $i < N - 1$, then $w^{i+1} = w'1$, where w' is the longest string such that $w'0$ is a prefix of w^i.

The public verification key PK remains fixed throughout the lifetime of the system. In the time period i, the signer generates a signature respect to the node secret key S_{w^i}, but the secret key SK_i contains secret keys of the right siblings of the nodes on the path from the root to w^i, besides the secret node key S_{w^i}. That is, whenever $w'0$ is a prefix of w^i, SK_i contains the secret key of node $w'1$. So the secret key of the time period i is expressed by $SK_i = (S_{w^i|_{\bar{1}}}, S_{w^i|_{\bar{2}}}, \cdots, S_{w^i|_{\bar{n}}}, S_{w^i})$, where $w^i = w_1 \ldots w_n$ and $S_{w^i|_{\bar{k}}} = \mathsf{Null}$ if the last bit of $w^i|_k$ is 1. At the end of the time period i, the key update algorithm can generate the secret key as the following:

1. If w^i is an internal node, then $w^{i+1} = w0$ and generates $S_{w^{i+1}}$. Additionally, it generates the secret node key S_{w^i1} for the right child node w^i1.
2. If w^i is a leaf, the secret node key $S_{w^{i+1}}$ is already contained in SK_i.

After generating the secret key of w^{i+1}, it erases the secret node key S_{w^i} in storage. This process can be easily implemented via stack. The secret key SK_i can be organized as a stack of node keys $\mathsf{ST\text{-}SK}$, with the secret node key S_{w^i} on top. When the signer runs the key update algorithm, first pops the current secret node key S_{w^i} off the stack.

1. If w^i is an internal node, generates secret keys S_{w^i0} and S_{w^i1} of w^i0 and w^i1, respectively, and pushes S_{w^i1} and then S_{w^i0} onto the stack. The new top is S_{w^i0} and indeed $w^{i+1} = w^i0$.
2. If w^i is a leaf then the next key on top of the stack is $S_{w^{i+1}}$.

In either case, it erases the secret node key S_{w^i}.

3.2 A construction of forward secure signature scheme

We now construct a forward secure signature scheme $\mathsf{FS.PKS}$ using bilinear maps. Let $\mathcal{IG}$ be a GDH parameter generator for which the GDH assumption holds.

FS.GEN : It takes as input the security parameter k, depth of the binary tree l. Then

1. Run $\mathcal{IG}(1^k)$ to groups $\mathbb{G}_1$, $\mathbb{G}_2$ of some prime order q and an admissible pairing $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$.
2. Select a random generator $P \in \mathbb{G}_1$ and a random secret $\alpha \in \mathbb{Z}_q^*$, and sets $Q = \alpha P$.
3. Chooses cryptographic hash functions $H_1 : \{0, 1\}^* \to \mathbb{G}_1$, and $H_2 : \{0, 1\}^* \times \{0, 1\}^* \times \mathbb{G}_1 \to \mathbb{G}_1$.
4. The public key is $PK = (\mathbb{G}_1, \mathbb{G}_2, e, P, Q, l, H_1, H_2)$ and the root secret key is $SN_\varepsilon = \alpha H_1(\varepsilon)$.

FS.UPD: It takes as input the tree public key PK, the time period i and the secret key $SK_i =$ ST-SK. Let w be the node corresponding to i. In general, for $w = w_1 \cdots w_n$, the secret key of the node w consists of $n+1$ group elements, $S_w = (R_{w|_1}, R_{w|_2}, \cdots, R_{w|_{n-1}}, R_w, SN_w)$. It first pops the secret node key S_w off the stack ST-SK and then updates a secret key respect to the position of node w in tree as follows.

1. If w is an internal node, then chooses random $\rho_{w0}, \rho_{w1} \in \mathbb{Z}_q^*$, and computes $R_{w0} = \rho_{w0} P$, $R_{w1} = \rho_{w1} P$, $SN_{w0} = SN_w + \rho_{w0} H_1(w0)$ and $SN_{w1} = SN_w + \rho_{w1} H_1(w1)$. Then pushes

$$S_{w1} = (R_{w|_1}, \cdots, R_{w|_{n-1}}, R_w, R_{w1}, SN_{w1}) \text{ and } S_{w0} = (R_{w|_1}, \cdots, R_{w|_{n-1}}, R_w, R_{w0}, SN_{w0})$$

   in order onto the stack, and erases S_w.
2. If w is a leaf, then only erases S_w.

FS.SIGN: It takes as input the time period i, the secret key $SK_i =$ ST-SK and a message M. The signer pops the top in the stack ST-SK and uses it to generate a signature. Let $w = w_1 \ldots w_n$ be the node corresponding to i. Then the signer selects $r \in \mathbb{Z}_q^*$ and computes $U = rP$, $P_M = H_2(M, i, U)$ and $FS = SN_w + rP_M$. The signer outputs a signature $\langle i, \sigma = (U, FS) \rangle$ and $R_{w|_m}$ where $1 \le m \le n$.

FS.VRFY: Let $w = w_1 \ldots w_n$ be the node corresponding to i. When $P_M = H_2(M, i, U)$, if

$$e(P, FS) = \prod_{m=1}^{n} e(R_{w|_m}, H_1(w|_m)) \cdot e(U, P_M) \cdot e(Q, H_1(\varepsilon)),$$

then confirms that $\langle i, \sigma = (U, FS) \rangle$ is a valid signature of $\langle M, i \rangle$.

Completeness The verification of the signature is justified by the following equations:

$$\begin{aligned}
&\prod_{m=1}^{n} e(R_{w|_m}, H_1(w|_m)) \cdot e(U, P_M) \cdot e(Q, H_1(\varepsilon)) \\
&\quad = \prod_{m=1}^{n} e(P, \rho_{w|_m} H_1(w|_m)) \cdot e(P, rP_M) \cdot e(P, \alpha H_1(\varepsilon)) \\
&\quad = e(P, \sum_{m=1}^{n} \rho_{w|_m} H_1(w|_m) + rP_M + \alpha H_1(\varepsilon)) = e(P, FS).
\end{aligned}$$

Efficiency We associate time periods with all nodes of a binary tree rather than with the leaves only as was done in prior work [16, 13]. This improves efficiency in the key-generation and key-update times from $O(\log N)$ to $O(1)$ as stated in [6]. The total signing computation is independent of the total number of the time periods N. The verification requires more computation because it involves the pairing computation and its complexity is $O(\log N)$. However, it can be optimized by precomputing $\prod_{m=1}^{n} e(R_{w|_m}, H_1(w|_m))$ and $e(Q, H_1(\varepsilon))$ only once at the time period for the given signer. Additionally, each size of the signature, the public key and the secret key is $O(\log N)$, $O(1)$ and $O(\log N)$, respectively. However, it is noted that the secret storage size is only $O(1)$ using in current period rather than $O(\log N)$.

Security The following theorem shows that the security of FS.PKS depends on the hardness of the CDH problem on $\mathbb{G}_1$.

Theorem 1. If the group $\mathbb{G}_1$ generated by $\mathsf{FS.GEN}(K, l)$ is (t', ϵ')-GDH group, then FS.PKS is $(t, q_S, q_{H_2}, \epsilon)$-forward secure, for every (t, ϵ) such that

$$t \le t' - c_{\mathbb{G}_1}(q_{H_2} + 3q_S + Tl + 4l + T + 3) \quad \text{and} \quad \epsilon \ge N \cdot \epsilon' + \frac{(q_{H_2} - 1) \cdot q_S}{q}.$$

The proof of this theorem is similar to that of the theorem 2. Due to space limitation, we will omit this proof.

4 Efficiency Improvement

Although the verification phase of FS.PKS scheme can be optimized by precomputing almost pairings once at the time period for the given signer, it requires 3 pairing computations basically. We show that the number of pairing computations in each verification procedure can be reduced only 3 (no precomputation) at the cost of two field multiplications in each key update procedure. We now describe this forward secure signature scheme EFS.PKS. Let $\mathcal{IG}$ be a GDH parameter generator for which the GDH assumption holds.

EFS.GEN : It takes as input the security parameter k, depth of the binary tree l. Then

1. Run $\mathcal{IG}(1^k)$ to groups $\mathbb{G}_1$, $\mathbb{G}_2$ of some prime order q and an admissible pairing $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$.
2. Select a random generator $P \in \mathbb{G}_1$ and a random secret $\alpha \in \mathbb{Z}_q^*$, and sets $P_{\text{pub}} = \alpha P$.
3. Chooses cryptographic hash functions $H_1 : \{0, 1\}^* \times \mathbb{G}_1 \to \mathbb{G}_1$, $H_2 : \{0, 1\}^* \times \{0, 1\}^* \times \mathbb{G}_1 \to \mathbb{G}_1$, and $H_3 : \{0, 1\}^* \times \mathbb{G}_1 \to \mathbb{Z}_q^*$.
4. The public key is $PK = (\mathbb{G}_1, \mathbb{G}_2, e, P, P_{\text{pub}}, l, H_1, H_2, H_3)$ and the root secret key is $SN_\varepsilon = \alpha H_1(\varepsilon, P_{\text{pub}})$.

EFS.UPD: It takes as input the tree public key PK, the time period i and the secret key $SK_i =$ ST-SK. Let w be the node corresponding to i. In general, for $w = w_1 \cdots w_n$, the secret key of the node w consists of $n+1$ group elements, $S_w = (R_{w|_1}, R_{w|_2}, \cdots, R_{w|_{n-1}}, R_w, SN_w)$. It first pops the secret node key S_w off the stack ST-SK and then updates a secret key respect to the position of node w in tree as follows.

1. If w is an internal node, then chooses random different $\rho_{w0}, \rho_{w1} \in \mathbb{Z}_q^*$, and computes $R_{w0} = \rho_{w0}P$, $R_{w1} = \rho_{w1}P$, $SN_{w0} = SN_w + h_{w0}\rho_{w0}H_1(\varepsilon, P_{\text{pub}})$ and $SN_{w1} = SN_w + h_{w1}\rho_{w1}H_1(\varepsilon, P_{\text{pub}})$ where $h_{w0} = H_3(w0, R_{w0})$ and $h_{w1} = H_3(w1, R_{w1})$. Then pushes

$$S_{w1} = (R_{w|_1}, \cdots, R_{w|_{n-1}}, R_w, R_{w1}, SN_{w1}) \text{ and } S_{w0} = (R_{w|_1}, \cdots, R_{w|_{n-1}}, R_w, R_{w0}, SN_{w0})$$

in order onto the stack, and erases S_w.
2. If w is a leaf, then only erases S_w.

EFS.SIGN: It takes as input the time period i, the secret key $SK_i =$ ST-SK and a message M. The signer pops the top in the stack ST-SK and uses it to generate a signature. Let $w = w_1 \dots w_n$. Then the signer selects $r \in \mathbb{Z}_q^*$ and computes $U = rP$, $P_M = H_2(M, i, U)$ and $FS = SN_w + rP_M$. The signer outputs a signature $\langle i, \sigma = (U, FS)\rangle$ and $R_{w|_m}$ where $1 \le m \le n$.

EFS.VRFY: Let $w = w_1 \dots w_n$. When $P_M = H_2(M, i, U)$ and $h_{w|_m} = H_3(w|_m, R_{w|_m})$ for $1 \le m \le n$, if

$$e(P, FS) = e(P_{pub} + \sum_{m=1}^{n} h_{w|_m} R_{w|_m}, H_1(\varepsilon, P_{\text{pub}})) \cdot e(U, P_M)$$

then, confirms that $\langle i, \sigma = (U, FS)\rangle$ is a valid signature of $\langle M, i\rangle$. Completeness can be checked easily as follows.

$$\begin{aligned} e(P_{pub} &+ \sum_{m=1}^{n} h_{w|_m} R_{w|_m}, H_1(\varepsilon, P_{\text{pub}})) \cdot e(U, P_M) \\ &= e(P, (\alpha + \sum_{m=1}^{n} h_{w|_m}\rho_{w|_m})H_1(\varepsilon, P_{\text{pub}})) \cdot e(P, rP_M)) \\ &= e(P, (\alpha + \sum_{m=1}^{n} h_{w|_m}\rho_{w|_m})H_1(\varepsilon, P_{\text{pub}}) + rP_M)) = e(P, FS). \end{aligned}$$

4.1 Security proof

The following theorem shows that the security of EFS.PKS depends on the hardness of the CDH problem on $\mathbb{G}_1$.

Theorem 2. If the group $\mathbb{G}_1$ generated by EFS.GEN(K, l) is (t', ϵ')-GDH group, then EFS.PKS is $(t, q_S, q_{H_2}, \epsilon)$-forward secure, for every (t, ϵ) such that

$$t \le t' - c_{\mathbb{G}_1}(q_{H_2} + 3q_S + Tl + 4(T-1) + 1) \quad \text{and} \quad \epsilon \ge N \cdot \epsilon' + \frac{(q_{H_2} - 1) \cdot q_S}{q}.$$

As the same method as [6], each hash function H_1 and H_3 can be replaced by 1-wise and $(l+1)$-wise independent hash functions in function family $\mathcal{H}_1$ and $\mathcal{H}_3$, respectively. To achieve chosen message security, however, H_2 needs to remain a random oracle. We will prove the security of our scheme paying due regard to such factors.

Proof. The proof of this theorem uses ideas from [3, 1] and [5]. Let $\mathcal{A}$ be an algorithm $(t, q_S, q_{H_2}, \epsilon)$-attacking EFS.PKS. Then we will construct a new PPT adversary $\mathcal{B}$ which attempts to solve the CDH problem with respect to $\mathcal{IG}$ with probability $\geq \epsilon'$ running at most t' time. The new algorithm $\mathcal{B}$ is given a challenge (P, aP, bP) and the goal of $\mathcal{B}$ is to compute abP. For that purpose, $\mathcal{B}$ runs $\mathcal{A}$ as the subroutine.

$\mathcal{B}$ selects a total time period N and guesses the time period T where $0 \leq T \leq N-1$ at which $\mathcal{A}$ will ask the breakin query. Let $w^T = w_1^* w_2^* \cdots w_s^*$ be a bit string of the node corresponding to the time period T. For preparing key exposure, $\mathcal{B}$ chooses r_{w^T}, h_{w^T} and $r_{w^T|_{\bar{i}}}$, $h_{w^T|_{\bar{i}}}$ at random in $\mathbb{Z}_q^*$ where $1 \leq i \leq s$ and $w_i^* = 0$. Afterwards, $\mathcal{B}$ randomly samples hash function H_1 and H_3 from a 1-wise $\mathcal{H}_1$ and $(l+1)$-wise $\mathcal{H}_3$ independent hash families, respectively, with the following constraints for $1 \leq i \leq s$ and $w_i^* = 0$.

$$\begin{aligned} H_1(\varepsilon, aP) &= bP = I, \\ H_3(w^T|_{\bar{i}}, R_{w^T|_{\bar{i}}}) &= h_{w^T|_{\bar{i}}} \text{ where } R_{w^T|_{\bar{i}}} = 1/h_{w^T|_{\bar{i}}}(r_{w^T|_{\bar{i}}}P - aP), \\ H_3(w^T, R_{w^T}) &= h_{w^T} \text{ where } R_{w^T} = 1/h_{w^T}(r_{w^T}P - aP) \end{aligned}$$

These result in generating the node secrets contained in the secret key SK_T. $\mathcal{B}$ gives $PK = (\mathbb{G}_1, \mathbb{G}_2, e, P, Q = aP, l, H_1, H_2, H_3)$ and N to $\mathcal{A}$ as system parameters. $\mathcal{B}$ simulates hash queries, signing queries and breakin query from $\mathcal{A}$. $\mathcal{B}$ maintains a H_2-table and a signing query table to answer the queries from $\mathcal{A}$.

$\mathcal{B}$ initializes $j = 0$. $\mathcal{A}$ first gets access to an oracle for generating signatures for SK_0. As we noted in the formal security model, there is no reason that the chosen message attack for the base signing key SK_0 is not allowed. To simplify the proof concerning the formal security model, we assume that $\mathcal{A}$ outputs $d = 0$ after the chosen message attack for period 0 without loss of generality. As long as neither $d =$ breakin nor $j = N$, $\mathcal{B}$ moves into the next chosen message phase providing $\mathcal{A}$ an oracle for signing under the next key and updating keys preparing for breakin phase. We put key updates into chosen message attack phase for convenience of proof. Let $w^j = w_1 \cdots w_t$ be the node corresponding to the time period j.

– Chosen Message Attack Phase:

Key updates. Note that this procedure is done to by $\mathcal{B}$ without any requests of $\mathcal{A}$. It is just preparing answers for queries of next time periods and breakin query. Given a current time period j, $\mathcal{B}$ simulates the key update algorithm as follows.

1. If w^j is leaf node or $j = T$, $\mathcal{B}$ skips key update procedure.
2. If $w^j 0$ is w^T, then for $h_{w^j 0}, h_{w^j 1}, r_{w^j 0}, r_{w^j 1} \in \mathbb{Z}_q^*$

$$\begin{aligned} R_{w^j 0} = 1/h_{w^j 0}(r_{w^j 0}P - Q), \quad R_{w^j 1} = 1/h_{w^j 1}(r_{w^j 1}P - Q) \\ H_3(w^j 0, R_{w^j 0}) = h_{w^j 0}, \quad H_3(w^j 1, R_{w^j 1}) = h_{w^j 1}. \end{aligned}$$

have already defined during choosing H_3 in $\mathcal{H}_3$.

3. If $w^j 0 \neq w^T$ is a prefix of w^T, then $\mathcal{B}$ computes $R_{w^j 0} = \rho_{w^j 0}P$ for randomly chosen $\rho_{w^j 0}, h_{w^j 0} \in \mathbb{Z}_q^*$ and sets $H_3(w^j 0, R_{w^j 0}) = h_{w^j 0}$. For $r_{w^j 1}, h_{w^j 1} \in \mathbb{Z}_q^*$,

$$R_{w^j 1} = 1/h_{w^j 1}(r_{w^j 1}P - Q), \quad H_3(w^j 1, R_{w^j 1}) = h_{w^j 1}$$

have already defined during choosing H_3 in $\mathcal{H}_3$.

4. Otherwise, $\mathcal{B}$ randomly chooses ρ_{w^j0}, ρ_{w^j1}, h_{w^j0}, $h_{w^j1}P \in \mathbb{Z}_q^*$, computes

$$R_{w^j0} = \rho_{w^j0}P, \quad R_{w^j1} = \rho_{w^j0}P$$
$$H_3(w^j0, R_{w^j0}) = h_{w^j0}, \quad H_3(w^j0, R_{w^j0}) = h_{w^j0}.$$

Note again that these result in generating the secret key SK_T.

H_2 **queries.** At any time $\mathcal{A}$ can query the random oracle H_2. To respond to these queries, $\mathcal{B}$ maintains H_2-table as explained below. This is initially empty. When $\mathcal{A}$ queries the oracle H_2 at a point $\langle M, j, U\rangle$, $\mathcal{B}$ responds as follows:

1. If $\langle M, j, U\rangle$ already appears on the H_2-table in a tuple $\langle M, j, U, h, \lambda, \varphi\rangle$ then $\mathcal{B}$ responds with $H_2(M, j, U) = h \in \mathbb{G}_1$.
2. Otherwise, $\mathcal{B}$ picks $\lambda \in \mathbb{Z}_q^*$ at random and sets $h \leftarrow \lambda P$, $U \leftarrow U$ and $\varphi \leftarrow *$. $\mathcal{B}$ adds the tuple $\langle M, j, U, h, \lambda, \varphi\rangle$ to the H_2-table and responds to $\mathcal{A}$ by setting $H_2(M, j, U) = h \in \mathbb{G}_1$.

Signature queries. Let $\langle M, j\rangle$ be a signature query issued by $\mathcal{A}$. $\mathcal{B}$ responds to its query as follows:

If $j \neq T$

1. $\mathcal{B}$ picks $\lambda, \varphi \in \mathbb{Z}_q$ at random and computes $h \leftarrow \lambda P - (1/\varphi)I$ and $U \leftarrow \varphi Q$. If $H_2(M, j, U)$ is defined then $\mathcal{B}$ aborts.
2. $\mathcal{B}$ adds the tuple $\langle M, j, U, h, \lambda, \varphi\rangle$ to the H_2-table.
3. Using $h_{w^j|_i}$ and $\rho_{w^j|_i}$ $(1 \le i \le t)$ generated and stored via key update algorithm, computes $FS = \sum_{i=1}^{t} h_{w^j|_i}\rho_{w^j|_i}I + \varphi\lambda Q$. Then $\mathcal{B}$ gives $\langle j, \sigma = (U, FS)\rangle$ and $R_{w^j|_i}$ $(1 \le i \le t)$ to $\mathcal{A}$. Since

$$\begin{aligned} FS &= aI + \sum_{i=1}^{t} h_{w^j|_i}\rho_{w^j|_i}I + \varphi a H_2(M, j, U) \\ &= aI + \sum_{i=1}^{t} h_{w^j|_i}\rho_{w^j|_i}I + \varphi a(\lambda P - (1/\varphi)I) = \sum_{i=1}^{t} h_{w^j|_i}\rho_{w^j|_i}I + \varphi\lambda Q, \end{aligned}$$

$\mathcal{B}$ can generate the signature of $\langle M, j, U\rangle$ though it cannot compute $aI = abP$.

If $j = T$

1. $\mathcal{B}$ picks $\lambda, \varphi \in \mathbb{Z}_q^*$ at random and computes $h \leftarrow \lambda P$ and $U \leftarrow \varphi Q$. Adds the tuple $\langle M, T, U, h, \lambda, \varphi\rangle$ to the H_2-table.
2. $\mathcal{B}$ gives $\langle T, \sigma = (U, FS)\rangle$ and $R_{w^T|_i}$ $(1 \le i \le s)$ to $\mathcal{A}$ where $FS = S_{w^T} + \lambda U$.

– Breakin Phase:

When $\mathcal{A}$ outputs a decision value d, if neither $j < T$ and $d = 0$, nor $j = T$ and $d = \mathsf{breakin}$ occurs, then $\mathcal{B}$ reports failure and terminates. If $j < T$ and $d = 0$, then increments j and $\mathcal{A}$ moves into the chosen message attack phase for period j. When $j = T$ and $d = \mathsf{breakin}$, $\mathcal{B}$ returns the current secret key SK_T to $\mathcal{A}$. This can be easily derived from simulated key update algorithm. $\mathcal{B}$ can generate the node secret keys $S_{w^T|_{\bar{i}}}$ for right siblings of the nodes $w^T|_i$ on the path from the root to w^T using $h_{w^T|_m}$, $\rho_{w^T|_m}$ for $1 \le m \le i-1$ and $r_{w^T|_{\bar{i}}}$ as follows.

$$S_{w^T|_{\bar{i}}} = r_{w^T|_{\bar{i}}}I + \sum_{m=1}^{i-1} h_{w^T|_m}\rho_{w^T|_m}I.$$

Completeness can be easily checked because for $1 \le i \le s$,

$$\begin{aligned} S_{w^T|_{\bar{i}}} &= aI + \sum_{m=1}^{i-1} h_{w^T|_m}\rho_{w^T|_m}I + h_{w^T|_{\bar{i}}}\rho_{w^T|_{\bar{i}}}I \\ &= aI + \sum_{m=1}^{i-1} h_{w^T|_m}\rho_{w^T|_m}I + h_{w^T|_{\bar{i}}} \cdot 1/h_{w^T|_{\bar{i}}}(r_{w^T|_{\bar{i}}} - a)I \\ &= r_{w^T|_{\bar{i}}}I + \sum_{m=1}^{i-1} h_{w^T|_m}\rho_{w^T|_m}I. \end{aligned}$$

$\mathcal{B}$ returns the secret key of the time period T, $SK_T = (S_{w^T|_{\bar{1}}}, S_{w^T|_{\bar{2}}}, \cdots, S_{w^T|_{\bar{s}}}, S_{w^T})$, where $w^T = w_1 \ldots w_s$ and $S_{w^T|_{\bar{k}}} = \mathsf{Null}$ if the last bit of $w^T|_k$ is 1 as the response of breakin query to $\mathcal{A}$. Note that $\mathcal{B}$ already computed S_{w^T} during signature query phase for period T.

– Output forgery phase:
After above attack process, $\mathcal{A}$ eventually outputs a signature $\langle i, \sigma = (xP, FS)\rangle$ of the message/period pair $\langle M, i\rangle$ for some $1 \le i < T$. Let $w^i = w_1w_2 \cdots w_n$ be the node corresponding to the time period i. If this signature is valid,

$$FS = abP + \sum_{m=1}^{n} h_{w^i|_m}\rho_{w^i|_m}I + xH_2(M, i, xP).$$

If it is not, $\mathcal{B}$ reports failure and terminates. The probability to output forgery without a direct H_2 query about $\langle M, i, xP\rangle$ is negligible, so $\mathcal{B}$ can find the tuple $\langle M, i, xP, h, \lambda, *\rangle$ on the H_2-table and computes $abP = FS - \sum_{m=1}^{n} h_{w^i|_m}\rho_{w^i|_m}I - \lambda U$ using $h_{w^i|_m}$, $\rho_{w^i|_m}$ stored during key update procedure and λ. This completes the description of algorithm $\mathcal{B}$.

Analysis Now, we analyze the three events need for $\mathcal{B}$ to succeed.

Event E_1. $\mathcal{B}$ does not abort as a result of any of $\mathcal{A}$'s signature queries.
Event E_2. $\mathcal{A}$ outputs $d = \mathsf{breakin}$ and $j = T$.
Event E_3. $\mathcal{A}$ generates a valid signature forgery $\langle i, \sigma\rangle$ of the message $\langle M, i\rangle$ for some $0 \le i < T$.

The success probability ϵ' is at least $(\epsilon - \Pr[E_1])/N$ where $\Pr[E_1]$ is the probability of $\mathcal{B}$'s abortion arising during signature queries.

Claim 1: $\Pr[E_1]$ is at most $\frac{(q_{H_2}-1)\cdot q_S}{q}$.

Proof. $q_{H_2} - q_s$ is the number of H_2 queries which does not include queries made by the signing and verifying algorithms of $\mathcal{B}$. When the k-th signature query is issued, in the worst case at $q_{H_2} - q_s + (k-1)$ of H_2 queries are previously defined. Hence the probability of $\mathcal{B}$ aborting on the k-th signature query ($k \in \{1, 2, \cdots q_s\}$) is at most $(q_{H_2} - q_s + (k-1))/q$ where q is the size of the domain over which U is chosen, actually φ. Note that the random choice of φ makes U random. Finally, the total abortion probability is obtained after summing the above over k and it is at most $(q_{H_2} - 1) \cdot q_S/q$ since $q_{H_2} \ge 1$.

Claim 2: $\Pr[E_2] \geq 1/N$.

Proof. Since $\mathcal{A}$ cannot distinguish the simulation given by $\mathcal{B}$ and Experiment 1, the probability that T fixed by $\mathcal{B}$ equals to the time which breakin attack occurs by $\mathcal{A}$ is at least $1/N$.

Claim 3: $\Pr[E_3] \geq \epsilon$.

Proof. $\mathcal{A}$ will produce a valid signature forgery with probability at least ϵ.

Using the bounds from the claims above, we can see that $\mathcal{B}$ produces the correct answer with probability at least $(\epsilon - (q_{H_2} - 1) \cdot q_S/q)/N \geq \epsilon'$ as required. $\mathcal{B}$'s running time is the same as $\mathcal{A}$'s running time plus the time of followings. Denote $c_{\mathbb{G}_1}$ the running time of a scalar multiplication in $\mathbb{G}_1$. In the following analysis, we don't consider addition in $\mathbb{G}_1$ and multiplication in $\mathbb{Z}_q^*$.

1. H_2-query. One scalar multiplication for the direct H_2 query and three for the indirect H_2 query arising from signature queries are required. Hence, the expected running time is $c_{\mathbb{G}_1}(q_{H_2} - q_S) + 3q_S) = c_{\mathbb{G}_1}(q_{H_2} + 2q_S)$.
2. Signature query and Key update.
   (a) For each period less than T, at least l scalar multiplications are required to compute $\sum_{i=1}^{t} h_{w^j|_i} \rho_{w^j|_i} I$. So the running time is $c_{\mathbb{G}_1} Tl$.
   (b) Signature is generated by adding $\varphi\lambda Q$ to $\sum_{i=1}^{t} h_{w^j|_i} \rho_{w^j|_i} I$. So the running time is $c_{\mathbb{G}_1} q_S$ for generating q_S signatures.
   (c) For one period, at most 4 scalar multiplications are required for key updating. So the running time for T periods is $c_{\mathbb{G}1}4(T-1)$. Note that $T-1$ is the number of key updates.
3. For the response to breakin query, $\mathcal{B}$ computes $S_{w^T|_{\bar{i}}}$ by adding $r_{w^T|_{\bar{i}}} I$ to $\sum_{m=1}^{i-1} h_{w^T|_m} \rho_{w^T|_m} I$. Thus, generating $S_{w^T|_{\bar{i}}}$ for $1 \leq i \leq s$ requires at most $c_{\mathbb{G}_1} l$.
4. To solve the CDH problem, $\mathcal{B}$ computes λU. So the running time is $c_{\mathbb{G}_1}$.

Hence, the total running time is at most

$$t + c_{\mathbb{G}_1}(q_{H_2} + 3q_S + Tl + 4(T-1) + 1)$$

as required. This completes the proof of Theorem.

References

1. M. Abdalla and L. Reyzin. A new forward-secure digital signature scheme. *Advances in Cryptology - ASIACRYPT 2000*, T. Okamoto (Ed.), Lecture Notes in Comput. Sci. **1976**, Springer-Verlag, pp. 116–129 (2000).
2. R. Anderson. Two remarks on public key cryptology. *Proc. of CCS'97*, ACM, 1997.
3. M. Bellare and S.K. Miner. A forward-secure digital signature scheme. *Advances in Cryptology - CRYPTO'99*, M. Wiener (Ed.), Lecture Notes in Comput. Sci. **1666**, Springer-Verlag, pp. 431–448 (1999).
4. M. Bellare and A. Palacio. Protecting against key exposure: strongly key-insulated encryption with optimal threshold. *Cryptology ePrint Archive*, Report **2002/64** (2002).
5. D. Boneh, B. Lynn and H. Shacham. Short signatures from the Weil pairing. *Advances in Cryptology - ASIACRYPT 2001*, C. Boyd (Ed.), Lecture Notes in Comput. Sci. **2248**, Springer-Verlag, pp. 514-532 (2001).

6. R. Canetti, S. Halevi and J. Katz. A forward-secure public-key encryption scheme. *Advances in Cryptology - EUROCRYPT 2003*, E. Biham (Ed.), Lecture Notes in Comput. Sci. **2656**, Springer-Verlag, pp. 255–271 (2003).
7. Y. Dodis, M. Franklin, J. Katz, A. Miyaji and M. Yung. Intrusion-resilient public-key encryption. *Topics in Cryptology - CT-RSA 2003*, M. Joye (Ed.), Lecture Notes in Comput. Sci. **2612**, Springer-Verlag, pp. 19–32 (2003).
8. Y. Dodis, M. Franklin, J. Katz, A. Miyaji and M. Yung. A generic construction for intrusion-resilient public-key encryption. *Topics in Cryptology - CT-RSA 2004*, T. Okamoto (Ed.), Lecture Notes in Comput. Sci. **2964**, Springer-Verlag, pp. 81–98 (2004).
9. Y. Dodis, J. Katz, S. Xu and M. Yung. Key-insulated public key cryptosystems. *Advances in Cryptology - EUROCRYPT 2002*, L.R. Knudsen (Ed.), Lecture Notes in Comput. Sci. **2332**, Springer-Verlag, pp. 65–82 (2002).
10. Y. Dodis, J. Katz, S. Xu and M. Yung. Strong key-insulated signature scheme. *Public Key Cryptography - PKC 2003*, Y.G. Desmedt (Ed.), Lecture Notes in Comput. Sci. **2567**, Springer-Verlag, pp. 130–144 (2003).
11. C. Gentry and A. Silverberg. Hierarchical ID-based cryptography, *Advances in Cryptology - ASIACRYPT 2002*, Y. Zheng (Ed.), Lecture Notes in Comput. Sci. **2501**, Springer-Verlag, pp. 548–566 (2002).
12. C.G. Günther. An identity-based key-exchange protocol. *Advances in Cryptology - EUROCRYPT'89*, J.-J. Quisquater and J. Vandewalle (Eds.), Lecture Notes in Comput. Sci. **434**, Springer-Verlag, pp. 29–37 (1990).
13. F. Hu, C.-H. Wu and J.D. Irwin. A new forward secure signature scheme using bilinear maps. *Cryptology ePrint Archive*, Report **2003/188**, 2003.
14. G. Itkis and L. Reyzin. Forward-secure signatures with optimal signing and verifying. *Advances in Cryptology - CRYPTO 2001*, J. Kilian (Ed.), Lecture Notes in Comput. Sci. **2139**, Springer-Verlag, pp. 332–354 (2001).
15. G. Itkis and L. Reyzin. SiBIR: Signer-base intrusion-resilient signatures. *Advances in Cryptology - CRYPTO 2002*, M. Yung (Ed.), Lecture Notes in Comput. Sci. **2442**, Springer-Verlag, pp. 499–514 (2002).
16. J. Katz. A forward-secure public-key encryption scheme. *Cryptology ePrint Archive*, Report **2002/060**. It is an order version of the paper [6].
17. J. Katz. Binary tree encryption: Constructions and applications. *Information Security and Cryptology - ICISC 2003*, J.I. Lim and D.H. Lee (Eds.), Lecture Notes in Comput. Sci., Springer-Verlag, to appear.
18. H. Krawczyk. Simple forward-secure signatures from any signature scheme. *Proc. of CCS'99*, pp. 108–115, 2000.
19. T. Malkin, D. Micciancio and S.K. Miner. Efficient generic forward-secure signatures with an unbounded number of time periods. *Advances in Cryptology - EUROCRYPT 2002*, L.R. Knudsen (Ed.), Lecture Notes in Comput. Sci. **2332**, Springer-Verlag, pp. 400–417 (2002).
20. T. Malkin, S. Obana and M. Yung. The hiererchy of key evolving signatures and a characterization of proxy signatures. *Advances in Cryptology - EUROCRYPT 2004*, C. Cachin (Ed.), Lecture Notes in Comput. Sci. **3027**, Springer-Verlag, pp. 306–322 (2004).
21. D.H. Yum and P.J. Lee. Efficient key updating signature schemes based on IBS. *Cryptography and Coding 2003*, K.G. Paterson (Ed.), Lecture Notes in Comput. Sci. **2898**, Springer-Verlag, pp. 167–182 (2003).