# Forward-Secure Identity-Based Signature: New Generic Constructions and Their Applications*

Noura Al Ebri, Joonsang Baek†, Abdulhadi Shoufan
*Khalifa University of Science, Technology and Research*
*Abu Dhabi, UAE*
{noura.alebri, joon.baek, abdulhadi.shoufan}@kustar.ac.ae

Quang Hieu Vu
*ETISALAT British Telecom Innovation Center*
*Abu Dhabi, UAE*
quang.vu@kustar.ac.ae

**Abstract**

As modern cryptographic schemes rely their security on the secrecy of the private keys used in them, exposing such keys results in a total loss of security. In fact, attackers have been developing various techniques to seize the secret keys rather than to cryptanalyze the underlying cryptographic primitives. Digital signature schemes, which are widely employed in many applications, are not an exception to the key exposure problem. A number of solutions for protecting signature schemes from key exposure have been proposed, and one of them is a forward-secure signature. Informally, forward-secure signature schemes can guarantee the unforgeability of the past signatures, even if the current secret signing key is exposed. In this paper, we propose an efficient generic construction of forward-secure identity-based signature (FSIBS) that retains unforgeability of past signatures in spite of the exposure of the current signing key. Our construction, supported by formal security analysis, brings about concrete FSIBS schemes which are more efficient than existing schemes in the literature. Especially, one of our instantiations of FSIBS based on discrete-log primitive turns out to be the most efficient among existing ones. We extend our generic construction employing the technique used in Merkle's tree signature to reduce the size of public parameters. Additional contribution of this paper is to refine the definition of security of FSIBS in such a way that users in the system can freely specify time periods over which their signing keys evolve.

**Keywords**: Forward-Secure Identity-Based Signature, Key Exposure, Mobile Devices

## 1 Introduction

Recent years have seen explosive growth in use of mobile devices. As this trend broadens and deepens, many people started to carry out their important tasks on mobile devices. Some of the tasks are so critical that they need high level of security. While cryptographic solutions are usually employed to provide strong protection for digital assets, their deployment in mobile devices is not an easy task. Although

*Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, volume: 4, number: 1, pp. 32-54

*This paper is an extended version of the work originally presented at the 6th International Conference on Availability, Reliability and Security (ARES'12), Prague, Chzech Republic, August 2012 , titled "Efficient Generic Construction of Forward-Secure Identity-Based Signature" [1]. This version differs from our previous paper in that it contains an extension of our generic construction of forward-secure identity-based signature using Merkle's tree signature to achieve auxiliary storage reduction; it also contains a full description of an instantiation of the generic construction from pairing-based cryptographic primitives.

†Corresponding author: Khalifa University of Science, Technology & Research (KUSTAR), P.O.Box 127788, Abu Dhabi, UAE, Tel: +971-(0)2-5018587, Fax: +971-(0)2-4472442

current advances in mobile technology make the emergence of cheap yet powerful small devices possible, it would always be preferable to reduce intensive cryptographic computations that consume energy and eventually affect battery life. In particular, in case public key cryptography is employed, managing complex public key infrastructure (PKI) could be a problem. Furthermore, since mobile devices are small, it is relatively easy for attackers to physically capture them, which can lead to *key exposure*.

Identity-based signature (IBS) [2] schemes can be a remedy to the problem of PKI management in digital signature schemes. In IBS schemes, signatures can be checked solely using signers' identities like names or email addresses instead of randomly-generated public keys which need digital certificates to be accompanied in order to map signers' identities and the public keys. Moreover, as it has been known (but has not been well iterated) since its invention, IBS schemes can be constructed without heavy pairing operations. (Pairing, however, played a crucial role in constructing identity-based encryption [3].) As exemplified in [4], very efficient IBS schemes without pairing can be constructed. The remaining problem is how to provide protection against key exposure.

In a vast majority of cryptographic schemes, including identity-based signatures, security guarantees last as long as secret keys remain unrevealed. Once the secret key is revealed, the security is compromised for past and future signatures. Frequent key revocation could be a solution to the key exposure problem, but it is inefficient and gives no guarantee on the security (unforgeability) of past signatures. Another solution based on secret sharing schemes could be considered but it turns out to be costly in terms of computational cost and communication overhead as mentioned in [5]. One of the promising solutions to this problem is to combine the concepts of forward-secure signature (FSS) [5] [6] [7] and identity-based signature (IBS) to obtain a notion of forward-secure IBS (FSIBS). To get an idea of this scheme, imagine that Alice wants to authenticate a sensitive document using an IBS scheme implemented in her mobile device. As it is an IBS scheme, once the document is signed by Alice using a signing key stored in her device, Bob will verify the validity of the signature using Alice's identity. Now, assume that Malice got a hold of Alice's signing key by stealing her mobile device. Malice can now forge signatures on any messages. If Bob, a third party, knows that Alice's signing key was stolen at some point, he will question the validity of all Alice's signatures created before the compromise because he cannot tell whether the signature is created by Alice or by Malice. FSIBS schemes provide a guarantee that Bob can be sure that Alice's signatures created before the key exposure are genuine and not forgeries.

Interestingly, the key exposure problem of IBS schemes can be an issue not only in mobile devices but also in several other applications and services. One example is the recent service model in cloud computing, "Data-as-a-Service (DaaS)", which is to provide data on demand in a so-called "data marketplace [8]" such as the ones introduced by Amazon [9] and Microsoft [10]: Assume that in DaaS, users receive some valuable documents from a data provider and these documents are authenticated with IBS schemes. But suppose that the provider's secret signing key is exposed (or stolen). In this case, the users will have a question as to whether the signatures on the data the provider generated previously are still valid and not forgeries. We can employ FSIBS schemes to address the issue.

Mobile Ad hoc Networks (MANETs) where a trusted administrator exists that will distribute initial system parameters to all nodes [11]. Also, the administrator can authenticate the identity of each node and extract an initial private key. MANETs that require such features can utilize our proposed FSIBS (e.g., sensor networks, military networks, emergency and disaster networks, etc). Since the nodes in such networks are prone to be stolen or lost (their secret key is exposed), using our FSIBS will protect the security of past communications.

## 1.1 Related Work

The concept of forward-secure signature (FSS) was first proposed by Back [6] and Anderson [7], and formalized later by Bellare and Miner [5]. Since Bellare and Miner's work, a number of FSS schemes

have been proposed. One of the notable constructions was due to Krawczyk [12], who creates a FSS scheme from any standard signature scheme and a forward-secure pseudo random generator (FSPRG). Other works include [13, 14, 15, 16], [17] and [18]. (Readers are referred to [19] and [20] for excellent surveys on FSS schemes.)

Compared with FSS, research on FSIBS seems to be relatively less active. It seems that more work needs to be done to improve the efficiency and flexibility (in terms of the ability to be instantiated by various primitives). For example, Yu et al.'s very recent FSIBS scheme [21] needs a number of bilinear pairing operations, which could be too heavy for some mobile devices with limited computational capacity and battery life. Although Yu et al.'s scheme can be proven secure without resorting to random oracles [22], it would be interesting to explore the possibility of constructing more efficient schemes that provide provable security in the standard model.

We notice that more efficient FSIBS schemes can in fact be instantiated from Galindo et al.'s generic construction of FSIBS [23] resulted from their elegant construction of IBS based on standard digital signature schemes. However, we realized that their construction incurs some redundancy that inevitably results in expansion of computation and bandwidth.

## 1.2 Our Contributions

In this paper, we make the following contributions:

- First, we propose a new efficient generic construction of FSIBS. This construction, which we call "main scheme", is based on secure IBS and standard digital signature (DS) as well as forward-secure pseudorandom generator (FSPRG). It gives rise to very efficient FSIBS schemes when it is instantiated using efficient IBS and DS schemes. Compared with Galindo et al.'s generic construction [23], our construction turns out to yield a more efficient FSIBS scheme when both IBS and DS schemes are instantiated using discrete-log-based primitive, (like Schnorr signature [24]). To our knowledge, a forward-secure IBS scheme based on discrete-log-based primitive, resulted from this construction (main scheme as described in Section 3.4.1) seems to be the most efficient FSIBS scheme in the literature. When our main scheme is instantiated using pairing-based primitive, it turns out that the majority of the sub-algorithms of the resulting instantiation are more efficient than those of Yu et al.'s FSIBS scheme. (However, due to the difference in structure, some algorithms of their scheme are inevitably more efficient. Nevertheless, our scheme is more efficient in terms of total amount of computations.)

- Our second construction is an extension of the main scheme, which we call "extended scheme" to dramatically reduce the size of "auxiliary information" required to verify every signature generated while increasing the size of signature by factor of $O(\log T)$, where $T$ is a maximum number of periods. The technique used in the extended scheme is from Merkle's signature scheme [25] that utilizes the binary certification tree.

- Another contribution of this paper is to refine the formal definition of FSIBS. We found that the previous definition given by Yu et al. [21] has a scalability issue on the number of time periods that users in the system can choose for signing key update. Our definition of FSIBS resolves this issue.

The rest of the paper is organized as follows. The next section presents a new formal definition of FSIBS. Section 3 describes our generic construction of FSIBS and its extension. It also presents two instantiations of the main scheme based on discrete-logarithm and pairing-based primitives respectively. Security proofs for our constructions are provided in Section 4. Section 5 compares computational and space efficiency of the with those of other schemes in the literature. We conclude in Section 6.

## 2 Formal Definition of Forward-Secure Identity-Based Signature

In this section, we present a formal definition of FSIBS. The FSIBS schemes function similarly as FSS schemes do. However, in FSIBS schemes, a signer needs to get a private key associated with her identity from the Private Key Generator (PKG). Once the signer gets this private key from the PKG, she will create an initial key out of it and derive a (secret) signing key for the first time period. (Note that the initial key can be the same as the private key associated with the signer's identity.) This current signing key will be used to create a new signing key for the next time period and, once that key is created, the key for the current time period will be deleted securely. The signer uses a signing key bound to each time period to sign messages. Anyone who is in the possession of a signature, a message, the signer's identity and the time-period when the signature is created should be able to verify the signature.

A formal definition of FSIBS has been given in Yu et al.'s paper [21]. However, there is an important difference between their definition and ours. In our definition, a pre-specified number of time periods $T$ over which the secret signing keys evolve, is *determined by each user (signer)* and is requested to the PKG together with the signer's identifier information to create an initial secret signing key. On the other hand, in Yu et al.'s definition, $T$ is given by the PKG as a public parameter. We note that this causes a *scalability* issue. For example, there should be a conflict if some user wants $T$ to be 365 while some other user wants $T$ to be 24. This contradicts the basic principle that the same PKG's public parameter should be shared by every party in the system. Thus, we argue that the public parameter should not contain the user-specific information $T$.

We remark that while Galindo et al. [23] proposed a generic FSIBS scheme, formal definitions of FSIBS and its security were not given explicitly. Rather, the security proof of their FSIBS scheme was given as a corollary of the security proof of their generic IBS scheme with additional property, that is, "forward-security".

The following is our formal definition of FSIBS.

**Definition 1** (FSIBS)**.** *A FSIBS scheme consists of six polynomial-time algorithms, each of which is described in the following.*

1. ***PKGSetup****: Taking a security parameter $\lambda \in N$ as input, this algorithm generates a secret master key msk and public parameters params of the Private Key Generator (PKG). (Note that params is provided as input to all the sub-algorithms in the scheme.)*

2. ***UserKeyExt****: Let $id \in \{0,1\}^*$ be an identity of the user. We assume that id consists of the user's identifier information ID and some pre-specified number of time periods $T$ over which a signing key evolves. That is, $id = ID||T$. Taking id and msk, this algorithm generates a private key $sk_{id}$ associated with the identity id, which will be sent to the user securely.*

3. ***Initialize****: Taking as input id and $sk_{id}$, this algorithm generates an initial signing key $SK_0$ which will evolve over time periods and some auxiliary parameters necessary to manage the* ***Update*** *algorithm, which is described below.*

4. ***Update****: Taking as input id, an index of the current time period $t < T$ and the signing key $SK_{t-1}$ associated with the previous time period, this algorithm generates a singing key $SK_t$ for the current time period $t$.*

5. ***Sign****: Taking as input an index of the time period $t$, id, the signing key $SK_t$ associated with the time period $t$ and a message m, this algorithm generates a signature s of m associated with id and $t$.*

| Parameter | Meaning |
|---|---|
| $\lambda$ | Security Parameter |
| $msk$ | Master secret key |
| $params$ | Public parameters |
| $ID$ | User's identifier information |
| $T$ | Max. no. of time periods |
| $id$ | User identity $id = ID\|\|T$ |
| $sk_{id}$ | $id$-based private key |
| $t$ | Current time period index |
| $SK_0$ | Initial signing key |
| $SK_t$ | Current signing key |
| $SK_{t-1}$ | Previous signing key |
| $m$ | Message |
| $s$ | Signature |

Table 1: List of Parameters

6. ***Vrfy****: Taking as input id, an index of the current time period t, a message m and a signature s, this algorithm checks whether s is a valid signature of m or not. If s is valid, this algorithm outputs valid. Otherwise, outputs invalid.*

The list of the parameters used in the above definition is summarized in Table 1. Readers are referred to Figure 1 for the general structure of FSIBS.

Note that since the identity $id$ depends on the pre-specified number of time periods $T$, $sk_{id}$ can be used only for that "$T$" periods of time. (In other words, if $T$ periods of time elapse, the user needs to get a new key that depends on "new" $T$. If the user wants to reuse $T$, we can index $T$ producing $T_1$, $T_2$ and so on. All these are determined by system policy.) Note also that how to set $T$ is a trade-off between security (e.g. vulnerability to forgery) and the performance (e.g. cost of the update operation).

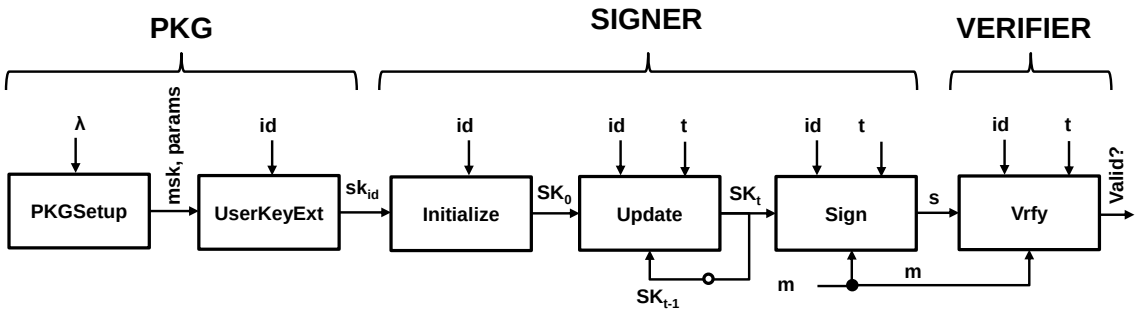



Figure 1: The General Structure of FSIBS

We remark that the time periods $t$ is merely an *index* not an absolute time, so it does not have to be synchronized between participating entities.

# 3 Proposed Constructions

## 3.1 An Idea for the Construction

A basic idea of our construction is that we use a (secure) IBS scheme to authenticate auxiliary parameters associated with time periods and use a (secure) standard signature scheme to sign messages under a signing key of each time period. On the other hand, Galindo et al. [23] constructs a FSIBS scheme by replacing one of the two (standard) signatures used in their IBS construction with a FSS scheme. Note that both Galindo et al.'s and our constructions adopt Krawczyk's idea of using the FSPRG primitive for signing key update [12]. We have discovered, however, that Galindo et al.'s FSIBS scheme constructed in this way results in some redundancy that causes increased computational overhead (especially, for signature verification) and signature size.

## 3.2 Building Blocks

For constructing our FSIBS scheme, we need the following building blocks.

The first building block we need is a standard digital signature scheme, which can be defined as follows [26] [27]. We note that even if a standard digital signature scheme is required to build our FSIBS scheme, we do not need a certificate that contains an authenticated public key for signature verification. The keys for signature generation/verification are generated for each time period and these keys are signed by the underlying identity-based signature scheme which does not require certificates. (This will be described shorlty.) As follows is a formal definition of the standard digital signature scheme.

**Definition 2** (DS)**.** *A digital signature (DS) scheme consists of three algorithms, KG, Sig and Ver. The user generates a secret signing key sk and a public key pk by running Setup. By running the Sig algorithm under her signing key sk, the user generates a signature $\sigma$ on a given message m. The validity of the signature is verified by anyone using pk via the Ver algorithm.*

The second one is an identity-based signature scheme, which can be defined formally as follows [28].

**Definition 3** (IBS)**.** *An identity-based signature (IBS) scheme consists of four algorithms, Setup, Extract, Sign and Vrfy. The Private Key Generator (PKG) generates a (secret) master key msk and a set of public parameters params by running Setup. (Note that params is shared by any interested parties.) The PKG uses its master key to generate a private key $sk_{id}$ associated with the user's identity id by running the Extract algorithm. By running the Sign algorithm under her $sk_{id}$ obtained from the PKG, the user generates a signature $\sigma$ on a given message m. The validity of the signature is verified using id via Vrfy.*

Lastly, we need a forward-secure pseudo random generator, which can be defined as follows [12] [29].

**Definition 4** (FSPRG)**.** *Let FSPRG: $\{0,1\}^{\iota} \rightarrow \{0,1\}^{m}$, where $\iota < m$, be a pseudo random generator and $k_0 \in \{0,1\}^{\iota}$ be an initial random seed. Then forward-secure pseudo random generator (FSPRG) recursively generates outputs as follows. $(r_i, k_i) \leftarrow FSPRG(k_{i-1})$ for $i = 1, \ldots, n$, where $r_i \in \{0,1\}^{m_L}$ and $k_i \in \{0,1\}^{m_R}$ with $m_L + m_R = m$, $m_L > 0$ and $m_R > 0$.*

Now we are ready to present our FSIBS construction.

## 3.3 Our Generic Constructions of FSIBS

### 3.3.1 Main Scheme

Let DS = (KG, Sig, Ver) and IBS = (Setup, Extract, Sign, Vrfy) be a standard digital signature scheme and an identity-based signature scheme as per Definition 2 and Definition 3 respectively. Also, let FSPRG

be a forward-secure pseudo random generator defined as per Definition 4. Based on these building blocks, our main FSIBS scheme, denoted by **FSIBS=** (**PKGSetup**, **UserKeyExt**, **Initialize**, **Update**, **Sign**, **Vrfy**), is constructed as follows:

- **PKGSetup**: The PKG generates a secret master key *msk* and public parameters *params* by running the Setup algorithm of the IBS. That is,

$$(msk, params) \xleftarrow{R} \text{Setup}(\lambda).$$

- **UserKeyExt**: Upon receiving $id = ID||T$ from the user, where $ID$ denotes the identifier information and $T$ denotes the pre-specified number of time periods over which a signing key evolves, the PKG generates $sk_{id}$, a private key associated with $id$, using its master key $msk$. That is,

$$sk_{id} \xleftarrow{R} \text{Extract}_{msk}(id).$$

  Then the PKG sends $sk_{id}$ to the user in a secure manner.

- **Initialize**: Having obtained $sk_{id}$, the user conducts the following steps:

  1. Select a random seed $k_0 \in \{0,1\}^{\iota}$ for FSPRG, where $\iota$ depends on the security parameter $\lambda$.
  2. For $i = 1$ till $T$ do
     2.1 Compute $(r_i, k_i) \leftarrow \text{FSPRG}(k_{i-1})$.
     2.2 Compute $(sk_i, pk_i) \leftarrow \text{KG}(\lambda, r_i)$.
     2.3 Compute $\sigma_i \leftarrow \text{Sign}_{sk_{id}}(id, i, pk_i)$.(Note that the algorithm Sign is from the IBS scheme.)
     2.4 Set $AUX_i = (id, i, pk_i, \sigma_i)$.
  3. Erase $sk_{id}$, $sk_i$, $k_i$ and $r_i$ for all $i = 1, \ldots, T$.
  4. Set $SK_0 = k_0$.
  5. Save $SK_0$ in a secure storage.
  6. Save $AUX_i$ for all $i = 1, \ldots, T$.

  Importantly, note that $AUX_i$ does not need to be stored in a secure storage. (No information about secret keys will be revealed from $AUX_i$.)

- **Update**: Given $(id, t, SK_{t-1})$, where $id = ID||T$, $t$ is an index of the current time period and $SK_{t-1}$ is the signing (or initial) key associated with the previous time period $t-1$, the user does performs the following:

  1. If $t = 1$, parse $SK_{t-1}$ into $k_{t-1}$. Otherwise, parse $SK_{t-1}$ into $(sk_{t-1}, k_{t-1})$.
  2. Compute $(r_t, k_t) \leftarrow \text{FSPRG}(k_{t-1})$.
  3. Compute $(sk_t, pk_t) \leftarrow \text{KG}(\lambda, r_t)$.
  4. Retrieve $AUX_t$ and parse it to $(A_1, A_2, A_3, A_4)$. Check if $A_1 = id$, $A_2 = t$ and $A_3 = pk_t$. If any of these tests fails, abort. If the checks succeed, test if $\text{Vrfy}_{id,params}((id, t, pk_t), A_4) = valid$. If it fails, abort, and continue otherwise. (Note that passing all of the checks implies that $A_1 = id$, $A_2 = t$, $A_3 = pk_t$ and $A_4 = \sigma_t$.)
  5. Set $SK_t = (sk_t, k_t)$.

6. Save $SK_t$ in a secure storage and erase $SK_{t-1}$. ($SK_t$ will serve as a signing key for the current period $t$.)

- **Sign**: Given $(id, t, m)$, where $id$ is an identity, $t$ is an index of current time period and $m$ is a message to be signed, the user does the following:

  1. Retrieve current values of $AUX_t$ and $SK_t$.
  2. Parse $SK_t$ into $(sk_t, k_t)$.
  3. Compute $\sigma \leftarrow \text{Sig}_{sk_t}(m)$ and output the signature $s = (AUX_t, \sigma)$ (Note that Sig is the signing algorithm of the standard signature scheme DS.)

- **Vrfy**: Given $(id, t, m, s)$, where $id$ is an identity, $t$ is an index of time period, $m$ is a message and $s$ is a signature, any party can verify the validity of $s$ as follows:

  1. Parse $s$ to $(AUX_t, \sigma)$.
  2. Parse $AUX_t$ to $(A_1, A_2, A_3, A_4)$.
  3. Check if $A_1 = id$ and $A_2 = t$.
  4. Check if $\text{Vrfy}_{id,params}((A_1, A_2, A_3), A_4) = valid$. (Note that passing the above tests implies that $A_1 = id$, $A_2 = t$, $A_3 = pk_t$ and $A_4 = \sigma_t$.)
  5. Check if $\text{Ver}_{A_3}(m, \sigma) = valid$.
  6. If all the above tests succeeded, output *valid*, otherwise output *invalid*.

### 3.3.2 Extended Scheme Based on Merkle's Signature Scheme

In our main scheme, the auxiliary values $AUX_1, \ldots, AUX_T$ need to be stored; in other words, the size of them grow linearly with $T$. To reduce such public (but non-secret) storage, we can apply use a similar technique used by Merkle's signature scheme [25]. This technique is based on building a binary certification tree where the leaves store the auxiliary value, which is $(AUX_t = (id, t, pk_t))$ denoted by $A_t$'s in Figure 2 and the nodes store a hash value computed by applying a collision resistant hash function "$Hash$" to the concatenation of its children's values. The value of the root $S$ will be signed using the identity-based signature scheme under the user's secret key $sk_{id}$. As a result, one can only store the $S$ value and the (identity-based) signature on it, thus reducing the storage. (The tree and the secret key $sk_{id}$ will be deleted.)

Notice that the signature size will be increased by a factor of $O(\log T)$ since a signature now should include a list of $O(\log T)$ (the height of tree) hash values in a path called "*authpath*". These hash values are need to reconstruct a root value from a leaf node $AUX_t$, denoted $A_t$, will be computed in a new **Update** algorithm. Notice also that the time for verification and key update will increase since, at every key update, the signer will have to recompute the partial hash tree that contains current and future auxiliary information.

In Figure 2 we present a simple example of a binary certification tree with height = 3, which has $T = 2^3 = 8$ periods.

As an example, assume that the signer will reconstruct a partial tree starting from period 5. Then $authpath_5 = (Y_6, Y_{7-8}, Y_{1-4})$ and the root value $S$ can be reconstructed as follows:

- Compute $Y_{5-6} = Hash(A_5 || A_6)$.
- Compute $Y_{5-8} = Hash(Y_{5-6} || Y_{7-8})$.

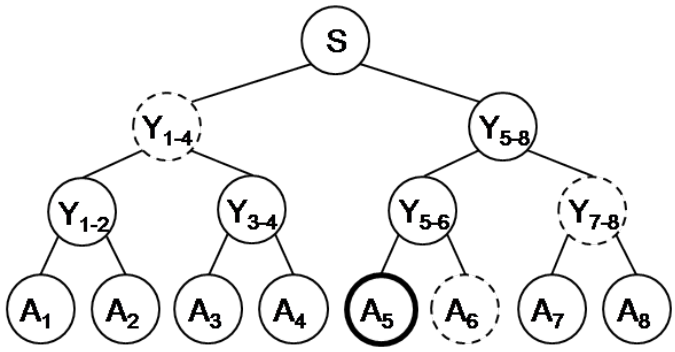



Figure 2: Storage Reduction of Auxiliary Information Using Merkle's Certification Tree

- Compute $S = Hash(Y_{1-4}||Y_{5-8})$.

In our extended scheme, a signature now becomes ($t$, $A_t(= AUX_t)$, $authpath_t$, $\psi$, $\sigma$), where $\sigma$ is a signature on a message under the private key $sk_t$ as described in the main scheme and $\psi$ is a signature on the root value $S$ under the private key $sk_{id}$.

We now describe the extended scheme more formally as follows. Let IBS = (Setup, Extract, Sign, Vrfy) and DS = (KG, Sig, Ver) be an identity-based signature scheme and a digital signature scheme respectively. Also, let FSPRG be a forward-secure pseudo random generator. Based on these building blocks, our extended FSIBS scheme, denoted by **FSIBS**$^e$= (**PKGSetup**$^e$, **UserKeyExt**$^e$, **Initialize**$^e$, **Update**$^e$, **Sign**$^e$, **Vrfy**$^e$), is constructed as follows:

- **PKGSetup**$^e$: The same as **PKGSetup** algorithm in the main scheme.
- **UserKeyExt**$^e$: The same as **UserKeyExt** algorithm in the the main scheme. For the sake of convenience, we assume that $T = 2^d$ for $d \in \mathbf{Z}^+$.
- **Initialize**$^e$: Having obtained $sk_{id}$, the user conducts the following steps:

  1. Select a random seed $k_0 \in \{0,1\}^\iota$ for FSPRG, where $\iota$ depends on the security parameter $\lambda$.
  2. For $i = 1$ till $T$ do
     2.1 Compute $(r_i, k_i) \leftarrow \text{FSPRG}(k_{i-1})$.
     2.2 Compute $(sk_i, pk_i) \leftarrow \text{KG}(\lambda, r_i)$.
     2.3 Set $AUX_i = (id, i, pk_i)$.
     2.4 Set $A_i = AUX_i$.
  3. Erase $sk_i$, $k_i$ and $r_i$ for all $i = 1, \ldots, T$.
  4. Build Merkle's binary certification tree in which leaves are $A_1, \ldots, A_T$.
     Each inner node of the tree, denoted $Y_{i-j}$ is the hash of the concatenation of its two children. More precisely

$$Y_{i-j} = \begin{cases} Hash(A_i||A_j) & \text{for} \quad j = i+1 \\ Hash(Y_{i-k}||Y_{(k+1)-j}) & \text{for} \quad k = 1, \ldots, T-1, \end{cases}$$

     where $Hash$ is a collision-resistant hash function.
  5. Compute $\psi \leftarrow \text{Sign}_{sk_{id}}(S)$, where $S = Y_{1-T}$. (Note that the algorithm Sign is from the IBS scheme.)

6. Set $SK_0 = k_0$ and erase $sk_{id}$.
7. Save $SK_0$ in a secure storage.
8. Save $\psi$.

- **Update**$^e$: Given $(id, t, SK_{t-1})$, where $id = ID||T$, $t$ is an index of the current time period and $SK_{t-1}$ is the signing (or initial) key associated with the previous time period $t-1$, the user does performs the following:

  1. If $t = 1$, parse $SK_{t-1}$ into $k_{t-1}$. Otherwise, parse $SK_{t-1}$ into $(sk_{t-1}, k_{t-1})$.
  2. Compute $(r_t, k_t) \leftarrow \text{FSPRG}(k_{t-1})$.
  3. Compute $(sk_t, pk_t) \leftarrow \text{KG}(\lambda, r_t)$.
  4. Set $A_t = AUX_t = (id_t, t, pk_t)$
  5. Build an authentication path $authpath_t = [auth_0, \ldots, auth_{d-1}]$, where $auth_i$'s are brother nodes that are needed to build parents of $A_t$ towards the root $S$.
  6. Set $SK_t = (sk_t, k_t)$.
  7. Save $SK_t$ in a secure storage and erase $SK_{t-1}$. ($SK_t$ will serve as a signing key for the current period $t$.)

  We remark that the Merkle tree traversal algorithm for computing authentication path has been improved, notably by Szydlo [30]. (A nice survey on certification tree and its traversal algorithms are given in [31]. )

- **Sign**$^e$: Given $(id, t, m)$, where $id$ is an identity, $t$ is an index of current time period and $m$ is a message to be signed, the user does the following:

  1. Retrieve current authentication path $authpath_t$, $A_t (= AUX_t)$, $\psi$ and $SK_t$.
  2. Parse $SK_t$ into $(sk_t, k_t)$.
  3. Compute $\sigma \leftarrow \text{Sig}_{sk_t}(m)$ and output the signature $s = (t, A_t, authpath_t, \sigma)$ (Note that Sig is the signing algorithm of the standard signature scheme DS.)

- **Vrfy**$^e$: Given $(id, t, m, s)$, where $id$ is an identity, $t$ is an index of time period, $m$ is a message and $s$ is a signature, any party can verify the validity of $s$ as follows:

  1. Parse $s$ to $(A_t, authpath_t, \sigma)$.
  2. Parse $A_t$ to $(W_1, W_2, W_3)$.
  3. Check if $W_1 = id$ and $W_2 = t$.
  4. Compute $S$ using $authpath_t$: $S = Hash(auth_{d-1}||Y_{l-m})$ (or $S = Hash(Y_{l-m}||auth_{d-1})$ ) where $Y_{l-m}$ is a brother of $auth_{d-1}$. Then check if $\text{Vrfy}_{id,params}(S, \psi) = valid$.
  5. Check if $\text{Ver}_{W_3}(m, \sigma) = valid$.
  6. If all the above tests succeeded, output *valid*, otherwise output *invalid*.

## 3.4 Instantiations

In this section, we provide two instantiations of our main scheme of generic FSIBS (presented in Section 3.3.1), one of which is based on "Schnorr-like lightweight IBS" [4] and the other one based on "Paterson-Schuldt IBS" [32].

### 3.4.1 An Instantiation from Schnorr-like lightweight IBS

One can instantiate our FSIBS construction using the Schnorr-like lightweight IBS scheme proposed by Galindo and Garcia [4]. The underlying DS scheme in our generic construction will be instantiated using a standard Schnorr signature scheme [24]. Note that these two schemes are built upon discrete-log (DL)-based primitives. Also, the FSPRG used in our generic construction can be instantiated by cryptographically-sound hash functions like SHA-1 and SHA-2, or a block cipher like AES as discussed in [29]. The details of the scheme are as follows:

- **PKGSetup**: Given a security parameter $\lambda$, the PKG generates a secret master key *msk* and a set of public parameters *params* as follows:

  1. Choose a group $\mathbf{G}$ of prime order $q = q(\lambda)$ with $2^{\lambda} \leq q < 2^{\lambda+1}$ . Pick a generator $g$ of $\mathbf{G}$.
  2. $G : \{0,1\}^* \rightarrow Z_q$, $H : \{0,1\}^* \rightarrow Z_q$ are descriptions of hash functions.
  3. Pick $z \xleftarrow{R} Z_q$. (Throughout this paper, $a \xleftarrow{R} S$ means that $a$ is picked uniformly at random from $S$. )
  4. Output
  $$(msk, params) = (z, (\mathbf{G}, g, q, g^z, G, H)).$$

- **UserKeyExt**: Upon receiving *id* from the user, the PKG generates a private key associated with it as follows:

  1. Pick $u \xleftarrow{R} Z_q$.
  2. Compute $y = u + zH(g^u, id)$.
  3. Output $sk_{id} = (y, g^u)$ which is sent securely to the user.

- **Initialize**: After obtaining $sk_{id}$, the user performs the following:

  1. Select a random seed $k_0$ for FSPRG.
  2. For $i = 1$ till $T$ do
     2.1 Compute $(k_i, r_i) \leftarrow \text{FSPRG}(k_{i-1})$.
     2.2 Compute $(sk_i, pk_i) \leftarrow \text{KG}(\lambda, r_i)$, where $sk_i = a_i \xleftarrow{R} Z_q$ and $pk_i = g^{a_i} = y_i$.
     2.3 Compute $\sigma_i \leftarrow Sign_{sk_{id}}(id, i, pk_i)$.
     2.4 Set $AUX_i \leftarrow (id, i, pk_i, \sigma_i)$, where $\sigma_i$ is computed using $sk_{id} = (y, g^u)$ as follows:
     (a) Pick $x \xleftarrow{R} Z_q$.
     (b) $f = x + yG(id, g^x, (id, i, pk_i))$.
     (c) Output $\sigma_i = (g^x, f, g^u)$.
  3. Set $SK_0 = k_0$.
  4. Save $SK_0$ in a secure storage.
  5. Erase $sk_{id}$ and all $sk_i, k_i, r_i$ for $i = 1, \ldots, T$.
  6. Store $AUX_i$ for all $i = 1...T$.

- **Update**: Given $(id, t, SK_{t-1})$, where *id* is an identity, $t$ is an index of time period and $SK_{t-1}$ is a singing (or initial) key associated with the previous time period, the user does the following:

1. If $t = 1$, parse $SK_{t-1}$ into $k_{t-1}$. Otherwise, parse $SK_{t-1}$ into $(sk_{t-1}, k_{t-1})$
2. Compute $(k_t, r_t) \leftarrow \text{FSPRG}(k_{t-1})$.
3. Compute $(sk_t, pk_t) \leftarrow \text{KG}(\lambda, r_t)$ where $sk_t = a_t \xleftarrow{R} Z_q$ and $pk_t = g^{a_t} = y_t$.
4. Retrieve $AUX_t$ and parse it to $(A_1, A_2, A_3, A_4)$ and check if $A_1 = id$, $A_2 = t$ and $A_3 = pk_t$. If any of these checks fails, abort.
5. Check $\text{Vrfy}_{id,params}((id, t, pk_t), A_4) = valid$, which is done as follows:
   (a) Parse $A_4$ into $(B_1, B_2, B_3)$.
   (b) Check whether or not the equation $g^{B_2} = B_1(B_3(g^z)^c)^d$ holds, where $c = H(g^u, id)$ and $d = G(id, B_1, (id, t, pk_t))$. If this check fails, abort. Otherwise continue. (Note that if $A_4$ is a valid signature, $B_1$, $B_2$ and $B_3$ correspond to $g^x$, $f$ and $g^u$ respectively.)
6. Set $SK_t = (sk_t, k_t) = (a_t, k_t)$.
7. Store $SK_t$ in a secure storage and erase $SK_{t-1}$.

- **Sign**: Given an identity $id$, an index of time period $t$ and a message $m$ to be signed, the user does the following:

1. Retrieve current values of $AUX_t$ and $SK_t$.
2. Parse $SK_t$ into $(sk_t, k_t) = (a_t, k_t)$.
3. Compute $\sigma \leftarrow \text{Sig}_{sk_t}(m)$. Here Sig is the signing algorithm of the standard Schnorr signature scheme, which can be described as follows:
   3.1 Pick $j \leftarrow Z_q$.
   3.2 Compute $l = g^j$.
   3.3 Compute $e = H(l, m)$.
   3.4 Compute $s = j + ea_t$.
   3.5 Output $\sigma = (e, s)$
4. Output the signature $s = (AUX_t, \sigma)$.

- **Verify**: Given an identity $id$, a message $m$, a time period $t$ and a signature $s$, verification is done as follows:

1. Parse $s$ into $(AUX_t, \sigma)$.
2. Parse $AUX_t$ into $(A_1, A_2, A_3, A_4)$.
3. Parse $A_4$ into $(B_1, B_2, B_3)$.
4. Check $A_1 = id$ and $A_2 = t$.
5. Check if $\text{Vrfy}_{id,params}((A_1, A_2, A_3), A_4) = valid$ which can be done by checking whether $g^{B_2} = B_1(B_3(g^z)^c)^d$ holds, where $c = H(g^u, id)$ and $d = G(id, B_1, (id, t, A_3))$. (Note that passing this test implies that $A_3 = pk_t (= y_t)$.)
6. Check if $\text{Ver}_{A_3}(m, \sigma) = valid$, done as follows:
   6.1 Parse $\sigma = (e, s)$.
   6.2 Check whether $e = H(g^s / {A_3}^e, m)$.
7. If all the above tests succeeded, output *valid*, otherwise output *invalid*.

### 3.4.2 An Instantiation from Paterson-Schuldt IBS

To create a FSIBS scheme secure in the standard model, one can instantiate our generic construction using primitives that are secure in the standard model like the Paterson-Schuldt IBS [32], and the short signature scheme proposed by Boneh and Boyen [33] and the FSPRG is a hash function or AES as mentioned before. The instantiation is as follows:

In the following instantiation, we assume that all identities and messages are bit strings of length $n_u$ and $n_m$, respectively. To construct a more flexible scheme which allows identities and messages of arbitrary lengths, collision-resistant hash functions, $H_u : \{0,1\}^* \leftarrow \{0,1\}^{n_u}$ and $H_m : \{0,1\}^* \leftarrow \{0,1\}^{n_m}$ will be used.

- **PKGSetup**:Choose groups $G$ and $G_l$ of prime order $p$ such that an admissible pairing $e : G \times G \leftarrow G_l$ can be constructed and pick a generator $g$ of $G$. Then select a secret $\alpha \xleftarrow{R} \mathbb{Z}_p$ and compute $g_1 = g^\alpha$ and pick $g_2 \xleftarrow{R} G$. Also, pick elements $u'$ and $m'$ randomly from $G$ and vectors $U = (u_i)$, $M = (m_i)$ of length $n_u$ and $n_m$, respectively, whose entries are random elements from $G$. Then, The public parameters $params = (G, G_l, e, g, g_1, g_2, u', U, m', M)$ and $msk = {g_2}^\alpha$.

- **UserKeyExt**: Let $u$ be a bit string of order $n_u$ that represent an identity and let $u[i]$ be the $i$th bit of $u$. Set $U \rightarrow 1, ..., n_u$ as the set of indices $i$ such that $u[i] = 1$. To generate the private key $sk_{id}$, choose $r_u \xleftarrow{R} \mathbb{Z}_p$ and compute:
  $sk_{id} = ({g_2}^\alpha (u' \prod_{i \in U} u_i)^{r_u}, g^{r_u})$

- **Initialize**: The user runs the KG algorithm of the standard signature scheme to get keys needed for the FSIBS scheme, together with other values needed for the scheme. The following shows the steps performed by the user before the start of period 1.

  1. Select a random seed $k_0$ for FSPRG.
  2. For $t = 1$ till $T$ do

  2.1 $(k_t, r_t) \leftarrow \text{FSPRG}(k_{t-1})$.

  2.2 $(sk_t, pk_t) \leftarrow \text{KG}(\lambda, r_t)$, done as follows:

  i. $G_1$, $G_2$ and $G_l$ are three cyclic groups of prime order $p$ and $e$ is a bilinear pairing $e : G_1 \times G_2 \rightarrow G_l$.

  ii. Select random generators $g_1 \in G_1$ and $g_2 \in G_2$, and random integers $x_t, y_t \in \mathbb{Z}_p$ that could be generated from $r_t$ or we set $x_t = r_t$.

  iii. Compute $u = {g_2}^{x_t} \in G_1$ and $v = {g_2}^{y_t} \in G_1$. Also, compute $z = e(g_1, g_2) \in G_l$.

  iv. Then, $(sk_t, pk_t) = ((g_1, x_t, y_t), (g_1, g_2, u, v, z))$.

  2.3 Compute $\sigma_t = Sign_{sk_{id}}(id, t, pk_t)$. Where $\sigma_t$ is done using the Sign of the IBS scheme as follows:

  i. let the bit strings $u$ represent the identity and $m$ represent the message.

  ii. Set $U$ as before and likewise set $M \rightarrow 1, ..., n_m$ as the set of indices $j$ such that $m[j] = 1$ and $m[j]$ be the $j$th bit of $m$.

  iii. To generate a signature choose $r_m \xleftarrow{R} \mathbb{Z}_p$.

  iv. The message $m = (id, t, pk_t)$ which means we represent them as bit strings.

  v. $\sigma_t = ({g_2}^\alpha (u' \prod_{i \in U} u_i)^{r_u} (m' \prod_{j \in M} m_j)^{r_m}, g^{r_u}, g^{r_m}) \in G^3$

  2.4 $AUX_t \leftarrow (id, t, pk_t, \sigma_t)$.

3. Set $SK_0 = k_0$.
4. Save $SK_0$ in a secure storage.
5. Erase $sk_{id}$ and all $sk_t, k_t, r_t$ for $t = 1$ till $T$.
6. Store $AUX_t$ for all $t = 1...T$.

- **Update**: At the start of each time period $t$ the user does the following:

1. If $t = 1$, parse $(sk_{t-1})$ into $k_{t-1}$. Otherwise, parse $(SK_{t-1})$ into $(sk_{t-1}, k_{t-1})$
2. Compute $(k_t, r_t) \leftarrow \text{FSPRG}(k_{t-1})$.
3. Compute $(sk_t, pk_t) \leftarrow \text{KG}(\lambda, r_t)$.
4. Retrieve $AUX_t$ and parse it to $(A_1, A_2, A_3, A_4)$ and check if $A_1 = id$, $A_2 = t$ and $A_3 = pk_t$. If any of these checks fails, abort.
5. Check $\text{Vrfy}_{id,params}((id, t, pk_t), A_4) = valid$, which is done as follows:
Verification of $A_4 = (V, R_u, R_m) \in G^3$ is done by checking if the following equality holds $e(V, g) = e(g_2, g_1)e(u' \prod_{i \in U} u_i, R_u)e(m' \prod_{j \in M} m_j, R_m)$.
6. Set $SK_t = (sk_t, k_t)$.
7. Store secretly $SK_t$ and erase $SK_{t-1}$.

- **Sign**: For a message $m$ to be signed do as follows:

1. Retrieve current values of $AUX_t$ and $SK_t$.
2. Parse $SK_t$ into $(sk_t, k_t)$.
3. Compute $\sigma \leftarrow Sig_{sk_t}(m)$, as follows:
   (a) Given a secret key $(g_1, x_t, y_t)$ and a message $m$, pick a random $r \in \mathbb{Z}_p / \{-\frac{x_t+m}{y_t}\}$.
   (b) Compute $\sigma \leftarrow {g_1}^{1/(x_t+m+y_t r)} \in G1$. Here, the inverse $1/(x_t + m + y_t r)$ is computed *modulo p*. The signature $\sigma = (\sigma_e, r)$.
4. Output the signature s = $(AUX_t, \sigma)$ where $\sigma = (\sigma_e, r)$.

- **Vrfy**: On inputs *id*, a message *m*, *params* and a signature string *s*, two verifications is needed. One for the IBS signature in $AUX_t$ and the other one is for the DS signature on the message *m*.

Also, the size of the public parameters can be reduced by using the technique by Sarkar and Chatterjee [34], where message and the identity vectors are modified to reduce the number of group elements in the public parameters to $n_u/s + n_m/t + 5$ where $n_m/n'_m = t$ and $n_u/n'_u = s$. For more details, readers are refered to [32].

# 4 Security Analysis

## 4.1 Security Definitions

First, we define unforgeability of FSIBS against chosen message attack, called "UF-FSIBS-CMA".

**Definition 5** (UF-FSIBS-CMA)**.** *Let $\mathscr{F}_{\text{FSIBS}}$ be a probabilistic polynomial-time forger that attacks a (general) FSIBS scheme, "FSIBS". (Readers are referred to Section 2.) UF-FSIBS-CMA for the scheme FSIBS is defined using the following game to which a security parameter $\lambda$ is provided as input.*

- ***Setup***: *The game runs PKGSetup providing $\lambda$ as security parameter to get the PKG's master key msk and the set of public parameters params.*
- ***Attack Phase***: *During this phase, the adversary $\mathscr{F}_{\text{FSIBS}}$ makes the following queries, each of which will be answered by the game.*
  - *UserKeyExt: On receiving this query denoted by $id(=ID||T)$, the game generates a user's private key $sk_{id}$ associated with the identity id and sends it to $\mathscr{F}_{\text{FSIBS}}$.*
  - *Sign: On receiving this query denoted by $(id,t,m)$, where $id = ID||T$, the game generates a signature s for a message m using a signing key $SK_t$ for period t, where $1 \leq t \leq T$.*
  - *Expose: On receiving this query denoted by $(id,t_e)$ where $id = ID||T$ and $1 \leq t_e \leq T$, the game returns a secret signing key $SK_{t_e}$ for period $t_e$.*
- ***Forgery Phase***: *In this phase, $\mathscr{F}_{\text{FSIBS}}$ outputs $(id^*,t^*,m^*,s^*)$, where $id^*(=ID^*||T)$, $t^*$, $m^*$ and $s^*$ denote an identity, a time period, a message and a signature respectively. A restriction here is that 1) $1 \leq t^* < t_e$ and 2) $id^*$ has not been issued as a UserKeyExt query and 3) $(id^*,t^*,m^*)$ has not been issued as a Sign query.*

*The FSIBS scheme is UF-FSIBS-CMA secure if for all $\mathscr{F}_{\text{FSIBS}}$,* $\Pr[\mathscr{F}_{\text{FSIBS}}$ *succeeds*$]$ *is negligible in $\lambda$.*

From now on, we review the security definitions of the various building blocks we used in our FSIBS construction.

First, unforgeability of a DS scheme against chosen message attack, called "UF-CMA" [35], can be defined as follows.

**Definition 6** (UF-CMA)**.** *Let $\mathscr{F}_{DS}$ be a probabilistic polynomial-time forger that attacks a generic DS scheme, "DS" as defined in Section 3.2. In the attack game which takes a security parameter $\lambda$ as input, $\mathscr{F}_{DS}$ issues a (polynomially-bounded) number of Sign queries. At forgery phase, $\mathscr{F}_{DS}$ outputs a message m which has not been issued as a Sign query and a signature $\sigma$ on m. The IBS scheme is UF-CMA secure if for all $\mathscr{F}_{DS}$,* $\Pr[\mathscr{F}_{DS}$ *succeeds*$]$ *is negligible in $\lambda$.*

Unforgeability of an IBS scheme against chosen message attack, called "UF-IBS-CMA", can be defined as follows [28].

**Definition 7** (UF-IBS-CMA)**.** *Let $\mathscr{F}_{IBS}$ be a probabilistic polynomial-time forger that attacks a generic IBS scheme, "IBS" as defined in Section 3.2. In the attack game to which the security parameter $\lambda$ is provided as input, $\mathscr{F}_{IBS}$ issues a number (but polynomially-bounded) of Extract queries, each of which is denoted by id, and Sign queries, each of which is denoted by $(id,m)$. At forgery phase, $\mathscr{F}_{IBS}$ outputs an identity/message pair $(id^*,m^*)$ such that $id^*$ has not been issued as an Extract query, $(id^*,m^*)$ has not been issued as a Sign query. The IBS scheme is UF-IBS-CMA secure if for all $\mathscr{F}_{IBS}$,* $\Pr[\mathscr{F}_{IBS}$ *succeeds*$]$ *is negligible in $\lambda$.*

Finally, security of called "ROR (Real or Random)-FSPRG", can be defined as follows [29].

**Definition 8** (ROR-FSPRG)**.** *Let "FSPRG" be a (generic) FSPRG scheme as defined in Section 3.2. Let t be an index of time period where an adversary $\mathscr{D}_{\text{FSPRG}}$ wants to break in, where $1 \leq t \leq T$ for some $T$. The game which takes a security parameter $\lambda$ as input, first computes $(r_1,k_1) \leftarrow FSPRG(k_0)$; $(r_2,k_2) \leftarrow FSPRG(k_1)$; $\cdots$; $(r_t,k_t) \leftarrow FSPRG(k_{t-1})$ and sets $\mu_1 = r_t$. Then the game picks $\mu_0 \in \{0,1\}^{m_L}$ uniformly at random, chooses $\gamma \in \{0,1\}$ at random and gives $(r_1,\ldots,r_{t-1},\mu_\gamma,k_t)$ to $\mathscr{D}_{\text{FSPRG}}$. $\mathscr{D}_{\text{FSPRG}}$ outputs $\gamma'$ which is a guess for $\gamma$. The FSPRG scheme is ROR secure if for all $\mathscr{D}_{\text{FSPRG}}$,* $\left|\Pr[\mathscr{D}_{\text{FSPRG}}\text{ succeeds}] - \frac{1}{2}\right|$ *is negligible in $\lambda$.*

## 4.2 Security Proofs

Now we prove the following theorem.

**Theorem 1.** *If the underlying DS, IBS and FSPRG schemes are UF-CMA, UF-IBS-CMA and ROR-FSPRG respectively, our main scheme **FSIBS** is UF-FSIBS-CMA-secure.*

*Proof.* We show that forgers for IBS and DS, and a distinguisher against FSPRG can be constructed using a forger against **FSIBS** as subroutine.

1. **Building a forger for DS**

   Let $\mathscr{F}_{\text{FSIBS}}$ be a forger against our **FSIBS** scheme. We build a forger $\mathscr{F}_{\text{DS}}$ against the underlying scheme DS, which will use **FSIBS** as follows:

   (a) **Setup:** Assume that $\mathscr{F}_{\text{DS}}$ is given a public key $pk$ of the DS scheme. Let $sk$ be a matching private key. $\mathscr{F}_{\text{DS}}$ now runs the PKGSetup algorithm to create $msk$ and $params$. Forger $\mathscr{F}_{\text{DS}}$ has access to a signing oracle $\text{Sig}_{sk}(.)$ which, given messages, returns corresponding signatures created under $sk$.

   (b) **Attack Phase:** Forger $\mathscr{F}_{\text{DS}}$ runs forger $\mathscr{F}_{\text{FSIBS}}$ against the **FSIBS** scheme. Upon receiving $\mathscr{F}_{\text{FSIBS}}$'s different queries (each of which can adaptively be created), $\mathscr{F}_{\text{DS}}$ answers as follows:

   - UserKeyExt: $\mathscr{F}_{\text{FSIBS}}$ queries identities, each of which is denoted by $id = ID||T$. Upon receiving these queries, $\mathscr{F}_{\text{DS}}$ will compute the corresponding $sk_{id}$ (as it knows the master key $msk$). Then for each $id = ID||T$, $\mathscr{F}_{\text{DS}}$ chooses a period number $\hat{t}$ at random between 1 and $T$ and picks $k_0 (= SK_0)$ at random, which will be used as an initial seed for FSPRG. It then generates $SK_i$ and $AUX_i$ for each period $i = 1, \ldots, \hat{t}-1, \hat{t}+1, \ldots, T$ in exactly the same way as the **Initialize** algorithm of the **FSIBS** scheme does. *For period $\hat{t}$, $\mathscr{F}_{\text{DS}}$ sets the public key $pk_{\hat{t}}$ in $AUX_{\hat{t}}$ to $pk$.*
   - Sign: $\mathscr{F}_{\text{FSIBS}}$ queries, each of which is denoted by $(id, t, m)$, where $id = ID||T$ is an identity, $m$ is a message and $t$ is a time period such that $1 \leq t \leq T$. (Note that if $t$ does not satisfy this condition, $\mathscr{F}_{\text{DS}}$ aborts.) Upon receiving $(id, t, m)$ where $t \neq \hat{t}$, $\mathscr{F}_{\text{DS}}$ retrieves matching $SK_t$ and $AUX_t$. It then uses $SK_t$ (parsed into $(sk_t, k_t)$) to generate a signature on $m$, which is denoted by $\sigma (= \text{Sig}_{sk_t}(m))$. $\mathscr{F}_{\text{DS}}$ returns $(AUX_t, \sigma)$ to $\mathscr{F}_{\text{FSIBS}}$. Upon receiving $(id, t, m)$ where $t = \hat{t}$, $\mathscr{F}_{\text{DS}}$ sends $m$ to its signing oracle $\text{Sig}_{sk}(.)$ to get the corresponding signatures and returns them to $\mathscr{F}_{\text{FSIBS}}$.
   - Expose: $\mathscr{F}_{\text{FSIBS}}$ asks for the signing key for a specified identity and time period $(id, t_e)$. Then $\mathscr{F}_{\text{DS}}$ provides $\mathscr{F}_{\text{FSIBS}}$ with the signing key $SK_{t_e}$ for that period. If $t_e = \hat{t}$ then $\mathscr{F}_{\text{DS}}$ aborts its run.

   (c) **Forgery Phase:** In this phase $\mathscr{F}_{\text{FSIBS}}$ outputs an identity $id^*$, a message $m^*$, a time period $t^*$ and a signature $s^*$, where $s$ is parsed into $(AUX_{t^*}, \sigma^*)$, where $AUX_{t^*}$ is parsed further into $(id^*, t^*, pk_{t^*}, \sigma_{t^*})$. $\mathscr{F}_{\text{FSIBS}}$ succeeds if

   i. $1 \leq t^* < t_e$.
   ii. $id^*$ has not been issued as a UserKeyExt query.
   iii. $(id^*, t^*, m^*)$ has not been issued as a Sign query.
   iv. $\text{Ver}_{pk_{t^*}}(m^*, \sigma^*) = valid$ and $\text{Vrfy}_{params, id^*}((id^*, t^*, pk_{t^*}), \sigma_{t^*}) = valid$.

   Now, if $t^* \neq \hat{t}$, $\mathscr{F}_{\text{DS}}$ aborts its run. Otherwise, $\mathscr{F}_{\text{DS}}$ outputs $(m^*, \sigma^*)$ as its forgery.

*Analysis*: Due to the restriction that $(id^*, t^*, m^*)$ has not been a Sign query, $\mathscr{F}_{\text{FSIBS}}$ needs to come up with a valid signature $\sigma^* = \text{Sig}_{sk_{t^*}}(m^*)$ by itself. By returning $(m^*, \sigma^*)$ as its signature, $\mathscr{F}_{DS}$ can succeed in forging a signature associated with the given public key $pk$ (which is equal to $pk_{t^*} = pk_{\hat{t}}$ if $\mathscr{F}_{DS}$ does not abort). However, note that because of the aborting events, $\mathscr{F}_{\text{DS}}$'s success probability is roughly $1/T$ times of that of $\mathscr{F}_{\text{FSIBS}}$.

2. **Building a forger for IBS**

Let $\mathscr{F}_{\text{FSIBS}}$ be the forger against the **FSIBS** scheme. We build a forger $\mathscr{F}_{\text{IBS}}$ against the underlying IBS scheme as follows.

(a) **Setup:** Assume that $\mathscr{F}_{\text{IBS}}$ is given the public parameters *params*. Then $\mathscr{F}_{\text{IBS}}$ gives *params* to $\mathscr{F}_{\text{FSIBS}}$. Forger $\mathscr{F}_{\text{IBS}}$ has access to an Extract and a signing oracles.

(b) **Attack Phase:** $\mathscr{F}_{\text{IBS}}$ chooses $l \in \{1, \ldots, q_{ex}\}$ at random, where $q_{ex}$ denotes the number of UseKeyExt queries.

$\mathscr{F}_{\text{IBS}}$ runs $\mathscr{F}_{\text{FSIBS}}$. Upon receiving $\mathscr{F}_{\text{FSIBS}}$'s different queries (each of which can adaptively be created), $\mathscr{F}_{\text{IBS}}$ responds to them as follows:

- UserKeyExt: $\mathscr{F}_{\text{FSIBS}}$ queries identities, each of which is denoted by $id = ID||T$. $\mathscr{F}_{\text{IBS}}$ first chooses $1 \leq \hat{t} \leq T$ at random. Upon receiving these queries, $\mathscr{F}_{\text{IBS}}$ does the following.

  If $id$ is not $l$-th query, do the following.

  Send $id$ to its Extract oracle to get a corresponding private key and return it to $\mathscr{F}_{\text{FSIBS}}$. Then pick $k_0 (= SK_0)$ at random, which will be used as an initial seed for FSPRG and generates $SK_i$ and $AUX_i$ for each period $i = 1, \ldots, T$.

  If $id$ is $l$-th query, abort.

- Sign: $\mathscr{F}_{\text{FSIBS}}$ queries, each of which is denoted by $(id, t, m)$, where $id = ID||T$ is an identity, $m$ is a message and $t$ is a time period such that $1 \leq t \leq T$. (Note that if $t$ does not satisfy this condition, $\mathscr{F}_{\text{IBS}}$ aborts.) Upon receiving $(id, t, m)$, $\mathscr{F}_{\text{IBS}}$ does the following.
  - If $id$ is not $l$-th query then do the following.
    - Retrieve matching $SK_t$ and $AUX_t$. It then uses $SK_t$ (parsed into $(sk_t, k_t)$) to generate a signature on $m$, which is denoted by $\sigma (= \text{Sig}_{sk_t}(m))$ and return $(AUX_t, \sigma)$ to$\mathscr{F}_{\text{FSIBS}}$.
  - If $id$ is $l$-th query then do the following.
    - If $t \neq \hat{t}$, pick $k_0 (= SK_0)$ at random, which will be used as an initial seed for FSPRG and generate $SK_i = (sk_t, k_t)$ and $pk_i$ for each period $i = 1, \ldots, T$. Then generate a signature on $m$, which is denoted by $\sigma (= \text{Sig}_{sk_t}(m))$. Then query $(id, t, pk_t)$ to its signing oracle to get signature $\sigma_t (= \text{Sign}_{sk_{id}}(id, t, pk_t))$. Set $AUX_t, = (id, t, pk_t, \sigma_t)$ and return $(AUX_t, \sigma)$ to$\mathscr{F}_{\text{FSIBS}}$.
    - If $t = \hat{t}$, abort.
- Expose: $\mathscr{F}_{\text{FSIBS}}$ asks for the signing key for a specified identity and time period $(id, t_e)$. Then $\mathscr{F}_{\text{IBS}}$ provides $\mathscr{F}_{\text{FSIBS}}$ with the signing key $SK_{t_e}$ for that period. If $id$ is $l$-th query and $t_e = \hat{t}$ then $\mathscr{F}_{\text{IBS}}$ aborts its run.

(c) **Forgery Phase:** In this phase $\mathscr{F}_{\text{FSIBS}}$ outputs an identity $id^*$, a message $m^*$, a time period $t^*$ and a signature $s^*$ which is parsed into $(AUX_{t^*}, \sigma^*)$, where $AUX_{t^*}$ is parsed further into $(id^*, t^*, pk_{t^*}, \sigma_{t^*})$. $\mathscr{F}_{\text{FSIBS}}$ succeeds if

i. $1 \leq t^* < t_e$.

ii. $id^*$ has not been issued as a UserKeyExt query.

iii. $(id^*, t^*, m^*)$ has not been issued as a Sign query.

iv. $\text{Ver}_{pk_{t^*}}(m^*, \sigma^*) = valid$ and $\text{Vrfy}_{params,id^*}((id^*, t^*, pk_{t^*}), \sigma_{t^*}) = valid$.

Once $\mathscr{F}_{\text{FSIBS}}$ outputs a forgery $(AUX_{t^*}, \sigma^*)$, $\mathscr{F}_{\text{IBS}}$ does the following: Check whether $id^*$ is $l$-th query and $t^* = \hat{t}$. If any of the equalities does not hold, $\mathscr{F}_{\text{IBS}}$ aborts its run. Otherwise, $\mathscr{F}_{\text{IBS}}$ outputs $(id^*, (t^*, pk_{t^*}), \sigma_{t^*})$, where $\sigma_{t^*}$ is from $AUX_{t^*}$, as its forgery.

*Analysis*: Basically, if $\mathscr{F}_{\text{IBS}}$ does not abort its run, the view of $\mathscr{F}_{\text{IBS}}$ is perfectly simulated. Note that $\sigma_{t^*}$ that $\mathscr{F}_{\text{IBS}}$ returns at the end of the forgery phase is a valid signature on $(id^*, (t^*, pk_{t^*}))$ under the key $sk_{id^*}$, which $\mathscr{F}_{\text{IBS}}$ tries to output as a forgery. We, however, lose some tightness of the reduction by the order of $O(\frac{1}{q_{ex}T})$ due to handling (ruling out) aborting events.

3. **Building an adversary for FSPRG**

Let $\mathscr{F}_{\text{FSIBS}}$ be a forger against the **FSIBS** scheme. We build an adversary $\mathscr{D}_{\text{FSPRG}}$ that attacks the FSPRG, where he will receive sequences $(r_1, \ldots, r_{\hat{t}-1}, \mu_\gamma, k_{\hat{t}})$, where $\gamma \in \{0,1\}$ is chosen at random, and must tell whether $\gamma = 1$ or 0. (Note that when $\gamma = 1$, $\mu_\gamma$ is the *next* output of FSPRG, i.e., $r_{\hat{t}}$, otherwise, $\mu_\gamma$ is a truly random value.) $\mathscr{D}_{\text{FSPRG}}$ will use the given sequence to create the view of $\mathscr{F}_{\text{FSIBS}}$ which $\mathscr{D}_{\text{FSPRG}}$ will run to decide $\gamma$ is 0 or 1.

(a) **Setup:** Given $\mu$, $\mathscr{D}_{\text{FSPRG}}$ runs the PKGSetup algorithm to create *msk* and *params*.

(b) **Attack Phase:** $\mathscr{D}_{\text{FSPRG}}$ will respond to $\mathscr{F}_{\text{FSIBS}}$'s queries as follows.

- UserKeyExt: $\mathscr{F}_{\text{FSIBS}}$ queries identities, each of which is denoted by $id = ID||T$. Upon receiving these queries, $\mathscr{D}_{\text{FSPRG}}$ will compute the corresponding $sk_{id}$ (as it knows the master key *msk*). After that, for each $id = ID||T$, $\mathscr{D}_{\text{FSPRG}}$ provides its given sequence $(r_1, \ldots, r_{\hat{t}-1}, \mu_\gamma, k_{\hat{t}})$ as input to the KG algorithm of DS scheme to generate public keys $pk_1, \ldots, pk_{\hat{t}-1}$ (and secret keys for $sk_1, \ldots, sk_{\hat{t}-1}$) for periods $1, \ldots, \hat{t}-1$. Then it generates $(sk_{\hat{t}}, pk_{\hat{t}})$ providing $\mu_\gamma$ as input to the KG algorithm of DS scheme. (Depending on whether $\gamma$ is 0 or 1, this key pair can be a right key pair for time period $\hat{t}$ or not.) Next, $\mathscr{D}_{\text{FSPRG}}$ generates $k_{\hat{t}+1}, \ldots, k_T$ for FSPRG using the given $k_{\hat{t}}$ and generates public keys $pk_{\hat{t}+1}, \ldots, pk_T$ (and secret keys for $sk_{\hat{t}+1}, \ldots, sk_T$) for periods $\hat{t}+1, \ldots, T$. It finally generates $AUX_i$ for each period $i = 1, \ldots, \hat{t}-1, \hat{t}+1, \ldots, T$ in exactly the same way as the **Initialize** algorithm of the **FSIBS** scheme does.
- Sign: $\mathscr{F}_{\text{FSIBS}}$ queries, each of which is denoted by $(id, t, m)$, where $id = ID||T$ is an identity, $m$ is a message and $t$ is a time period such that $1 \leq t \leq T$. Upon receiving $(id, t, m)$ where $t \neq \hat{t}$, $\mathscr{D}_{\text{FSPRG}}$ retrieves matching $sk_t$ and $AUX_t$. It then uses $sk_t$ to generate a signature on $m$, which is denoted by $\sigma (= \text{Sig}_{sk_t}(m))$. $\mathscr{D}_{\text{FSPRG}}$ returns $(AUX_t, \sigma)$ to $\mathscr{F}_{\text{FSIBS}}$. Upon receiving $(id, t, m)$ where $t = \hat{t}$, $\mathscr{D}_{\text{FSPRG}}$ computes $\sigma \leftarrow \text{Sig}_{sk_{\hat{t}}}(m)$ and returns it to $\mathscr{F}_{\text{FSIBS}}$.
- Expose: $\mathscr{F}_{\text{FSIBS}}$ asks for the signing key for a specified time period $t_e \geq \hat{t}$. Then $\mathscr{D}_{\text{FSPRG}}$ provides $\mathscr{F}_{\text{FSIBS}}$ with the signing key $SK_{t_e} = (sk_{t_e}, k_{t_e})$ for that period. If $t_e < \hat{t}$ then $\mathscr{F}_{\text{DS}}$ aborts its run.

(c) **Forgery Phase:** In this phase $\mathscr{F}_{\text{FSIBS}}$ outputs an identity $id^*$, a message $m^*$, a time period $t^*$ and a signature $s^*$ which is parsed into $(AUX_{t^*}, \sigma^*)$, where $AUX_{t^*}$ is parsed further into $(id^*, t^*, pk_{t^*}, \sigma_{t^*})$. $\mathscr{F}_{\text{FSIBS}}$ succeeds if

i. $1 \leq t^* < t_e$.

ii. $id^*$ has not been issued as a UserKeyExt query.

iii. $(id^*, t^*, m^*)$ has not been issued as a Sign query.

iv. $\text{Ver}_{pk_{t^*}}(m^*, \sigma^*) = valid$ and $\text{Vrfy}_{params,id^*}((id^*, t^*, pk_{t^*}), \sigma_{t^*}) = valid$.

Now, if $t^* \neq \hat{t}$, $\mathscr{D}_{\text{FSPRG}}$ aborts its run. Otherwise, $\mathscr{D}_{\text{FSPRG}}$ decides that $\gamma = 1$, that is, $\mu_\gamma$ is a real sequence generated by the FSPRG.

*Analysis*: The success probability of $\mathscr{D}_{\text{FSPRG}}$ is the same as that of $\mathscr{F}_{\text{FSIBS}}$ in forging a signature except for losing tightness of reduction by $O(1/T)$ due to handling of aborting events.

We have shown that by using a forger against the **FSIBS** scheme, forgers against DS and IBS and an adversary for FSPRG can be constructed. This means that the advantage (success probability) of the forger against the **FSIBS** scheme is bounded by the advantages of forgers and adversary against IBS, DS and FSPRG respectively. Hence, if those advantages of the forgers and adversary against DS, IBS and FSPRG are negligible, then the advantage of the forger against the **FSIBS** scheme is also negligible.

□

Our extended scheme can also be proven secure in a very similar way. We only state the theorem here:

**Theorem 2.** *If the underlying DS, IBS, FSPRG schemes and Hash used in binary certification tree are UF-CMA, UF-IBS-CMA, ROR-FSPRG and collision-resistant respectively, our extended scheme* ***FSIBS***$^e$ *is UF-FSIBS-CMA-secure.*

# 5 Performance Analysis

In this section, we compare performance of various forward-secure IBS schemes instantiated from our main scheme presented in Section 3.3.1 with that of other provably-secure FSIBS schemes. To do so, we first instantiated our generic construction (main scheme) and Galindo et al.'s [23] generic constructions using the same kind of cryptographic primitives: As described in Section 3.4.1, we instantiated our generic construction using the "Schnorr-like lightweight IBS" [4] and standard Schnorr signature schemes. We then instantiated Galindo et al.'s generic construction using the standard Schnorr signature scheme. We assume that both schemes use the same FSPRG and the same group **G** of elliptic curve points. We call our instantiation "FSIBS$_A$". Note that both schemes are DL (discrete-log)-based and the security of both schemes is proven in the random oracle model [22]. Note also that the calculation of three exponentiations (scalar multiplications on elliptic curves), which is needed in the verification process of the "Schnorr-like lightweight IBS", has a cost of about one and a half times that of a single exponentiation, according to [36].

For a pairing-based instantiation, we create a scheme called "FSIBS$_B$" which uses the Paterson-Schuldt IBS scheme [32] and the Boneh-Boyen DS scheme [33] as described in Section 3.4.2. We compare this scheme with Yu et al.'s FSIBS scheme [21] assuming that the same elliptic curve group where pairings can be efficiently computed. Note that both FSIBS$_B$ and Yu et al.'s schemes are secure in the standard model.

We summarized the computational overhead in Table 2. As can be seen from the table, all the sub-algorithms of FSIBS$_A$ are more efficient than those of Galindo et al.'s. The PKGSetup, UserKeyExt, Update and Singing algorithms of FSIBS$_B$ are more efficient than those of Yu et al.'s scheme. Although no computation for the Initialize algorithm is required in Yu et al.'s scheme due to structural difference, their Update algorithm needs significant computations proportional to $T$ (while FSIBS$_B$ needs constant amount of computations).

Table 2: Computational Overhead (DL = Discrete-Log, RO = Random Oracle, T = Total number of periods, S = Scalar multiplication on elliptic curves, P = Pairing.)

| Type | Scheme | PKGSetup | UserKeyExt | Initialize | Update | Signing | Verifying |
|---|---|---|---|---|---|---|---|
| DL-based | Our FSIBS$_A$ | 1S | 1S | T(2S) | 2.5S | 1S | 3.5S |
| with RO | Galindo et al. | 1S | 2S | 2S+T(2S) | 3S | 2S | 6S |
| Pairing-based | Our FSIBS$_B$ | 2S | 2S | T(3S) | 2S+3P | 1S | 4P+1.5S |
| without RO | Yu et al. | 2S+1P | 6S+T(4S) | None | 6S+T(4S) | 2S | 4P |

Table 3: Communication Overhead ( $|\mathbf{G}|$ = size of group in bits, $|id|$ = max. length of identity in bits, $|m|$ = max. length of message in bits. Assume that the same symmetric pairing $e : \mathbf{G} \times \mathbf{G} \rightarrow \mathbf{G}_T$ is used.)

| Type | Scheme | Sig. size | Sign. key size | AUX | *Params* |
|---|---|---|---|---|---|
| DL-based | Our FSIBS$_A$ | $6\|\mathbf{G}\|$ | $T\|\mathbf{G}\|$ | $4T\|\mathbf{G}\|$ | $2\|\mathbf{G}\|$ |
| with RO | Galindo et al. | $9\|\mathbf{G}\|$ | $T\|\mathbf{G}\|$ | $4T\|\mathbf{G}\|$ | $2\|\mathbf{G}\|$ |
| Pairing-based | Our FSIBS$_B$ | $10\|\mathbf{G}\|$ | $3T\|\mathbf{G}\|$ | $8T\|\mathbf{G}\|$ | $(\|id\|+\|m\|+5)\|\mathbf{G}\|$ |
| without RO | Yu et al. | $4\|\mathbf{G}\|$ | $\log^2 T\|\mathbf{G}\|$ | None | $(6+\|id\|+\|m\|+\|\log T\|)\|\mathbf{G}\|+2\|\mathbf{G_T}\|$ |

In Table 3, we summarized the communication overhead. As can be seen from the table, our FSIBS$_A$ scheme is more efficient than Galindo et al.'s scheme in terms of bandwidth consumption. The bandwidth cost resulted from using $AUX_t$ in our FSIBS$_B$ scheme can be reduced to be proportional to $\log T$ (not $T$) at the expense of increasing cost to compute $AUX_t$ , using the binary certification tree method suggested in [12] based on the work in [25], more details are present in the next section.

## 6 Conclusion

We proposed a provably-secure generic construction of FSIBS, which is based on secure IBS, DS and FSPRG schemes. Different from the previous construction [23] in the literature, our construction is greatly benefited from the underlying IBS scheme to yield efficient FSIBS schemes. For example, we showed that a very efficient FSIBS scheme can be instantiated by employing Schnorr-like lightweight IBS scheme [4]. We envision that especially this scheme could serve as a good security primitive for resource-limited devices such as mobile devices. We also showed that other theoretical but efficient FSIBS scheme based on pairing could emerge based on our generic construction. As an additional contribution, we presented a refinement of the definition of generic FSIBS schemes. Furthermore, we presented an extension of our main construction to reduce the size of public auxiliary information. (This is a trade-off scheme as it increases the size of signatures.)

Our future work includes a construction of efficient FSIBS schemes with unbounded number of time periods.

## Acknowledgment

The authors are grateful to the anonymous referees of JoWUA for their helpful comments.

# References


[1] N. A. Ebri, J. Baek, A. Shoufan, and Q. H. Vu, "Efficient generic construction of forward-secure identity-based signature," in *Proc. of the 7th International Conference on Availability, Reliability and Security (AReS'12), Prague, Czech Republic*. IEEE, August 2012, pp. 55–64.

[2] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Proc. of the 4th Annual International Cryptology Conference (CRYPTO'84), Santa Barbara, California, USA, LNCS*, vol. 196. Springer-Verlag, August 1985, pp. 47–53.

[3] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *Proc. of the 21st Annual International Cryptology Conference (CRYPTO'01), Santa Barbara, California, USA, LNCS*, vol. 2139. Springer-Verlag, August 2001, pp. 213–229.

[4] D. Galindo and F. D. Garcia, "A schnorr-like lightweight identity-based signature scheme," in *Proc. of the 2nd International Conference on Cryptology in Africa (AFRICACRYPT'09), Gammarth, Tunisia, LNCS*, vol. 5580. Springer-Verlag, June 2009, pp. 135–148.

[5] M. Bellare and S. K. Miner, "A forward-secure digital signature scheme," in *Proc. of the 19th Annual International Cryptology Conference (CRYPTO'99), Santa Barbara, California, USA, LNCS*, vol. 1666. Springer-Verlag, August 1999, pp. 431–448.

[6] A. Back, "Non-interactive forward secrecy," Cypherpunks Mailing List, 1996.

[7] R. Anderson, "Two remarks on public key cryptology (invited lecture)," in *Proc. of the 1997 ACM Conference on Computer and Communications Security (ACM CCS'97), Zurich, Switzerland*. ACM, April 1997, pp. 135–147.

[8] F. Data, "Buy and sell data at data marketplace," http://flowingdata.com/2010/03/22/buy-and-sell-data-at-data-marketplace/., 2010.

[9] A. Voronkov, "Amazon data sets," http://aws.amazon.com/publicdatasets/, 2012.

[10] Microsoft, "Microsoft data market," https://datamarket.azure.com/., 2012.

[11] I. Chlamtac, M. Conti, and J. J.-N. Liu, "Mobile ad hoc networking: Imperatives and challenges," *Ad Hoc Networks*, vol. 1, no. 1, pp. 13–64, July 2003.

[12] H. Krawczyk, "Simple forward-secure signatures from any signature scheme," in *Proc. of the 7th ACM conference on Computer and Communications Security (ACM CCS'00), Athens, Greece*. ACM, November 2000, pp. 108–115.

[13] T. Malkin, D. Micciancio, and S. Miner, "Efficient generic forward-secure signatures with an unbounded number of time periods," in *Proc. of the International Conference on the Theory and Applications of Cryptographic Techniques: Eurocrypt'02, Amsterdam, The Netherlands, LNCS*, vol. 2332. Springer-Verlag, April-May 2002, pp. 400–417.

[14] B. Alomair, K. Sampigethaya, and R. Poovendran, "Efficient generic forward-secure signatures and proxy signatures," in *Proc. of the 5th European PKI Workshop: Theory and Practice (EuroPKI'08), Trondheim, Norway, LNCS*, vol. 5057. Springer-Verlag, June 2008, pp. 166–181.

[15] B. G. Kang, J. H. Park, and S. G. Hahn, "A new forward secure signature scheme," Cryptology ePrint Archive, Report 2004/183, 2004.

[16] M. Abdalla and L. Reyzin, "A new forward-secure digital signature scheme," in *Proc. of the 6th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT'00), Kyoto, Japan, LNCS*, vol. 1976. Springer-Verlag, December 2000, pp. 116–129.

[17] G. Itkis and L. Reyzin, "Forward-secure signatures with pptimal signing and verifying," in *Proc. of the 21st Annual International Cryptology Conference (CRYPTO'01), Santa Barbara, California, USA, LNCS*, vol. 2139. Springer-Verlag, August 2001, pp. 332–354.

[18] A. Kozlov and L. Reyzin, "Forward-secure signatures with fast key update," in *Proc. of the 3rd international conference on Security in Communication Networks (SCN'02), Amalfi, Italy, LNCS*, vol. 2576. Springer-Verlag, September 2002, pp. 241–256.

[19] G. Itkis, "Forward security, adaptive cryptography: Time evolution," Handbook of Information Security, vol. 3, Chapter 199, H. Bidgoli (Ed), Wiley Publishers, 2006.

[20] E. Cronin, S. Jamin, T. Malkin, and P. McDaniel, "On the performance, feasibility, and use of forward-

secure signatures," in *Proc. of the 10th ACM Conference on Computer and Communications Security (ACM CCS'03), Washington D.C., USA*. ACM, October 2003, pp. 135–147.

[21] J. Yu, R. Hao, F. Kong, X. Cheng, J. Fan, and Y. Chen, "Forward-secure identity-based signature: Security notions and construction," *Information Sciences*, vol. 181, no. 3, pp. 648–660, February 2011.

[22] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in *Proc. of the 1st ACM conference on Computer and Communications Security (ACM CCS'93), Fairfax, Virginia, USA*. ACM, November 1993, pp. 62–73.

[23] D. Galindo, J. Herranz, and E. Kiltz, "On the generic construction of identity-based signatures with additional properties," in *Proc. of the 12th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT'06), Shanghai, China, LNCS*, vol. 4284. Springer-Verlag, December 2006, pp. 178–193.

[24] C. P. Schnorr, "Efficient identification and signatures for smart cards," in *Proc. of the 1989 Workshop on the Theory and Application of Cryptographic Techniques (EUROCRYPT'89), Houthalen, Belgium, LNCS*, vol. 434. Springer-Verlag, April 1989, pp. 688–689.

[25] R. C. Merkle, "A certified digital signature," in *Proc. of the 9th Annual International Cryptology Conference (CRYPTO'89), Santa Barbara, California, USA, LNCS*, vol. 435. Springer-Verlag, August 1989, pp. 218–238.

[26] Y. Lindell and J. N. Katz, *Introduction to Modern Cryptography*. Chapman & Hall CRC, 2008.

[27] W. Mao, *Modern Cryptography: Theory and Practice*. Prentice Hall PTR, 2004.

[28] E. Kiltz and G. Neven, "Identity-Based Signatures," in *Cryptology and Information Security Series on Identity-Based Cryptography*, M. Joye and G. Neven, Eds. IOS Press, 2008, pp. 31–44.

[29] M. Bellare and B. S. Yee, "Forward-security in private-key cryptography," in *Proc. of the 2003 RSA conference on The cryptographers' track (CT-RSA'03), San Francisco, California, USA, LNCS*, vol. 2612. Springer-Verlag, April 2003, pp. 1–18.

[30] M. Szydlo, "Merkle tree traversal in log space and time," in *Proc. of the 2004 International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'04), Interlaken, Switzerland, LNCS*, vol. 3027. Springer-Verlag, May 2004, pp. 541–554.

[31] B. Ederov, "Merkle tree traversal techniques," Ph.D. dissertation, Darmstadt University of Technology, April 2007.

[32] K. G. Paterson and J. C. N. Schuldt, "Efficient identity-based signatures secure in the standard model," in *Proc. of the 11th Australasian Conference on Information Security and Privacy (ACISP'06), Melbourne, Australia, LNCS*, vol. 4058. Springer-Verlag, July 2006, pp. 207–222.

[33] D. Boneh and X. Boyen, "Short signatures without random oracles and the SDH assumption in bilinear groups," *Journal of Cryptology*, vol. 21, no. 2, pp. 149–177, February 2008.

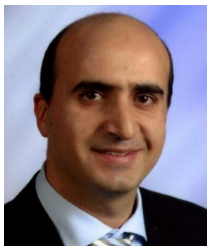
[34] P. Sarkar and S. Chatterjee., "Trading time for space: Towards an efficient IBE scheme with short(er) public parameters in the standard model," in *Proc. of the 8th International Conference on Information Security and Cryptology (ICISC'05), Seoul, Korea, LNCS*, vol. 3935. Springer-Verlag, December 2005, pp. 424–440.

[35] S. Goldwasser, S. Micali, and R. L. Rivest, "A digital signature scheme secure against adaptive chosen-message attacks," *SIAM Journal on Computing - Special issue on cryptography*, vol. 17, no. 2, pp. 281–308, April 1988.

[36] B. B. Brumley, "Efficient three-term simultaneous elliptic scalar multiplication with applications," in *Proc. of the 11th Nordic Workshop on Secure IT Systems (NordSec'06), Linköping, Sweden*. IEEE, October 2006, pp. 105–116.

**Noura Al Ebri**[1] obtained BSc in Information Systems from United Arab Emirates University in 2009 And her MSc in Information Security from Khalifa University in 2012. She is currently an IT Engineer at EAI in UAE. Her research areas include cryptographic algorithm design and analysis, computer and network security.

**Joonsang Baek** obtained his PhD from Monash University in 2004. He is currently an assistant professor in information security at Khalifa University. Before joining Khalifa University, he was a Research Fellow at Institute for Infocomm Research, ASTAR, Singapore. His research interests are in the area of applied cryptography,focusing on designing cost-efficient and highly-functional cryptographic schemes and protocols. Joonsang Baek is an internationally-recognized cryptographer and his work has presented appeared in reputable journals and conference proceedings.



**Abdulhadi Shoufan** received the Dr.-Ing. degree from the Technische Universität Darmstadt, Germany in 2007. Currently, he is an Assistant Professor for information security and ECE at Khalifa University in Abu Dhabi. His research areas include cryptographic hardware, embedded security, and engineering education. He was a manager of the project Quantum-Computer-Resistant Cryptographic Systems funded by the Federal Office of Information Security in German and leaded the hardware security group at the Center for Advanced Security Research Darmstadt.

**Quang Hieu Vu** is a Senior Researcher in the Etisalat BT Innovation Centre (EBTIC), Khalifa University for Science Technology and Research (KUSTAR), Abu Dhabi, UAE. Quang Hieu Vu obtained his PhD degree from Singapore-MIT Alliance (SMA), a collaboration program among three universities: Massachusetts Institute of Technology (MIT), National University of Singapore (NUS), and Nanyang Technological University (NTU). Prior to joining EBTIC, he worked respectively at NUS in Singapore, Imperial College London in UK, and Institute for Infocomm Research in Singapore (I2R). Quang Hieu Vu has research interests in Peer-to-Peer, Cloud Computing, Data Stream Processing, Query Optimization, and Network Security. He joined and made major contributions to several projects in these research areas. In addition to working in projects, he published several papers at top conferences and journals such as SIGMOD, VLDB, ICDE, VLDB Journal and TKDE. He wrote a book about P2P published by Springer-Verlag in 2009. Quanf Hieu Vu has received many awards including the NUS Dean's graduate award in 2005 and 2006, the prestigious Singapore President's graduate award in 2006 and 2007, and the best paper award at AP2PS conference in 2009 and 2010.

[1] No photo available