Forward Secure Ring Signature without Random Oracles [⋆]

Joseph K. Liu[1], Tsz Hon Yuen[2], and Jianying Zhou[1]

[1] Institute for Infocomm Research, Singapore
{ksliu,jyzhou}@i2r.a-star.edu.sg
[2] University of Hong Kong, Hong Kong
thyuen@cs.hku.hk

Abstract. In this paper, we propose a forward secure ring signature scheme without random oracles. With forward security, if a secret key of a corresponding ring member is exposed, all previously signed signatures containing this member remain valid. Yet the one who has stolen the secret key cannot produce any valid signature belonged to the past time period. This is especially useful in the case of ring signature, as the exposure of a single secret key may result in the invalidity of thousands or even millions ring signatures which contain that particular user. However, most of the ring signature schemes in the literature do not provide forward security. The only one with this feature [14] relies on random oracles to prove the security. We are the *first* to construct a forward secure ring signature scheme that can be proven secure without random oracles. Our scheme can be deployed in many applications, such as wireless sensor networks and smart grid system.

1 Introduction

Ring signatures [18] allow a member of a group to sign a message on behalf of the whole group. The verifier does not know who is the real signer in the group. The group can be set dynamically by the signer and no collaboration is needed between the members of the group.

In traditional public key cryptography, the security of a cryptosystem is guaranteed under some intractability assumptions if the secret key is kept away from the adversary. However, there are many ways that a secret key may be comprised in the real world. Hackers may steal your secret key if your computer is infected with trojans, or when you use the secret key in a phishing website. Therefore, it is important to minimize the damage even if the entire secret key is lost. When the attacker has full access to your secret key, he can sign or decrypt on behalf of the victim. The

[⋆] The first and third authors are supported by the EMA project SecSG-EPD090005RFP(D).

situation is even worse for ring signatures, since the attacker can forge a message on behalf of the whole group. Moreover, the other members of the group may be completely unaware of such forgery, since they are unaware of being conscripted into the group.

Forward security for signatures [3] was designed to the key exposure problem. A forward secure signature in the past remains secure even if the current secret key is lost. The first solution was designed by Bellare and Miner [5]. The main idea is to divide the lifetime of the public key into T intervals, and in each time interval the same public key corresponds to different secret keys. A current secret key can be used to derive the secret key in the future, but not the past. Therefore, even a compromise of the current secret key does not enable the adversary to forge signatures pertaining to the past.

Forward secure ring signatures were proposed by Liu and Wong [14] to resolve the key exposure problem in ring signatures. The motivation is to reduce the damage of exposure of any secret key of users in ring signature. Even if a secret key is compromised, previously generated ring signatures remain valid and do not need to be re-generated. They proposed the security model and gave a concrete construction in the random oracle model. Here we first review some practical applications of forward secure ring signatures.

1.1 Applications

AD-HOC NETWORKS: The steadily growing importance of portable devices and mobile applications has spawned new types of groups of interacting parties: ad-hoc groups. The highly dynamic nature of such groups raises new challenges for networking. Ad-hoc networks may be described as networks with minimal infrastructure, lacking fixed routers or stable links. Wireless sensor network is a kind of ad-hoc network. Such networks inherently deal with spontaneous ad-hoc groups: a group of users who spontaneously wish to communicate sensitive data need a suite of protocols which do not involve any trusted third party or certification of any new public keys. Security goals have to be considered in this new context. Ring signatures are perfectly suited to such a setting, since no setup protocol is required. Without forward security, the compromise of a user secret key in the group may result in the invalidity of all previous ring signatures involved with this user (including this user in the ring). The consequence is very serious. The whole authentication system may be suspended if only one user secret key has been compromised. Thus forward security is an important addition to ring signature, especially in the application of ad-hoc group.

SMART GRID: Smart Grid [17] is a form of electricity network utilizing modern digital technology. The most distinctive feature in smart grid is its two-way capabilities for data communication: Not only the grid controller can issue commands to intelligent devices, consumers and devices can also send data to grid controllers. The ability to access, analyze, and respond to much more precise and detailed data from all levels of the electric grid is critical to the major benefits of the Smart Grid. As an example, Microsoft Hohm [16] provides a platform for consumers to upload energy usage data, based on which a statistical report is created. The purpose is to encourage consumers to compare their energy consumption with others (e.g., on the same street) and thus use electricity more efficiently. Data integrity is a necessary requirement in those applications since the comparison would be meaningless if the data is maliciously modified or faked. Privacy, on the other hand, is also a significant concern: Consumers may not want to give their identity information to any third-party service providers. Ring signature is a promising solution on applications (e.g., Microsoft Hohm) requiring both integrity and privacy. In ring signature, a valid signature will convince the service provider that the data is uploaded by a consumer on a certain street, without telling who exactly the consumer is. Forward-security is certainly desirable in this situation since a compromised private key within a time period will not have any negative impact on statistical reports generated previously. In other words, old statistical reports would remain valid if forward-security is satisfied.

Our Contributions. The security proofs for various cryptosystems used the random oracle model [6]. Several papers [9, 4] showed that it is possible to prove a cryptosystem secure in the random oracle while the actual construction is insecure when the random oracle is instantiated by any real-world hash function. Thus, it is desirable to design cryptosystems provably secure without requiring random oracles.

In this paper, we propose the *first* forward-secure ring signatures without random oracles. The forward secure ring signatures proposed by Liu and Wong [14] is only secure in the random oracle model. We prove the security under the CDH and subgroup decision problem.

Related Works. There are some ring signature schemes that do not rely on the random oracles. Xu et al. [23] described a ring signature scheme in the standard model, but the proof is not rigorous and is apparently flawed [7]. Chow et al. [11] proposed a ring signature scheme in the standard model, though it is based on a strong new assumption. Bender et al.

[7] gave a ring signature secure in the standard model assuming trapdoor permutations exists. Their scheme uses generic ZAPs for NP as a building block, which may not be practical. Shacham and Waters [20] presented an efficient ring signature scheme without random oracles, based on standard assumption. They rely on composite order pairing that requires a trusted setup procedure. Schäge and Schwenk [19] gave another ring signature scheme in the standard model using basic assumption. In contrast to [20], they used prime order pairing instead. However, their security model does not allow the adversary to query any private key. All the above schemes do not provide forward security.

On the other side, the concept of forward secure signatures was first proposed by Anderson [3] for traditional signatures. It was formalized by Bellare and Miner [5]. The basic idea is to extend a standard digital signature algorithm with a key update algorithm, so that the secret key can be changed frequently while the public key stays the same. The resulting scheme is forward secure if the knowledge of the secret key at some point in time does not help forge signatures relative to some previous time period. The challenge is to design an efficient scheme of this concept. In particular the size of the secret key, public key and signature should not be dependent on the number of time period during the lifetime of the public key. Several schemes [2, 13, 1, 12, 15] have been proposed by traditional signatures and threshold signatures that satisfy this efficiency property. In addition, a forward secure group signature scheme and a forward secure identity-based signature scheme are proposed in [21] and [24] respectively.

2 Preliminaries

2.1 Pairings

We make use of bilinear groups of composite order. Let N be a composite number with factorization $N = pq$. We have: (1) $\mathbb{G}$ is a multiplicative cyclic group of order N. (2) $\mathbb{G}_p$ is its cyclic order-p subgroup, and $\mathbb{G}_q$ is its cyclic order-q subgroup. (3) g is a generator of $\mathbb{G}$, while h is a generator of $\mathbb{G}_q$. (4) $\mathbb{G}_T$ is a multiplicative group of order N. (5) e is a bilinear map such that $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ with the following properties:

- *Bilinearity*: For all $u, v \in \mathbb{G}$, and $a, b \in \mathbb{Z}$, $e(u^a, v^b) = e(u, v)^{ab}$.
- *Non-degeneracy*: $\langle e(g, g)\rangle = \mathbb{G}_T$ whenever $\langle g\rangle = \mathbb{G}$.
- *Computability*: It is efficient to compute $e(u, v)$ for all $u, v \in \mathbb{G}$.

(6) $\mathbb{G}_{T,p}$ and $\mathbb{G}_{T,q}$ are the $\mathbb{G}_T$-subgroups of order p and q, respectively. (7) The group operations on $\mathbb{G}$ and $\mathbb{G}_T$ can be performed efficiently. (8) Bit strings corresponding to elements of $\mathbb{G}$ and of $\mathbb{G}_T$ can be recognized efficiently.

2.2 Mathematical Assumptions

Definition 1. ***Computational Diffie-Hellman (CDH) in*** $\mathbb{G}_p$. *Given the tuple* (r, r^a, r^b), *where* $r \in_R \mathbb{G}_p$, *and* $a, b \in_R \mathbb{Z}_p$, *compute and output* r^{ab}. *In the composite setting one is additionally given the description of the larger group* $\mathbb{G}$, *including the factorization* (p, q) *of its order* N.

The CDH assumption is formalized by measuring an adversary's success probability for computational Diffie-Hellman, that is,

$$Adv_{CDH} = \Pr[\text{ Adversary outputs } r^{ab}].$$

Definition 2. ***Subgroup Decision.*** *Given* w *selected at random either from* $\mathbb{G}$ *(with probability* $1/2$*) or from* $\mathbb{G}_q$ *(with probability* $1/2$*), decide whether* w *is in* $\mathbb{G}_q$. *For this problem one is given the description of* $\mathbb{G}$, *but not given the factorization of* N.

The assumption is formalized by measuring an adversary's guessing advantage for the subgroup decision problem. That is,

$$Adv_{SD} = \left| \Pr[\text{ Adversary guesses correctly }] - \frac{1}{2} \right|.$$

Note that if CDH in $\mathbb{G}_p$ is hard then so is CDH in $\mathbb{G}$. The assumption that the subgroup decision problem is hard is called Subgroup Hiding (SGH) assumption, and was introduced by Boneh et al [8].

3 Security Model

3.1 Syntax of Forward Secure Ring Signatures

A *forward secure ring signature* (FSRS) scheme, is a tuple of four algorithms (KeyGen, Sign, Verify and Update).

- $(sk_{i,0}, pk_i) \leftarrow \mathsf{KeyGen}(1^\lambda)$ is a PPT algorithm which, on input a security parameter $\lambda \in \mathbb{N}$, outputs a private/public key pair $(sk_{i,0}, pk_i)$ such that the private key is valid for time $t = 0$.[3] We denote by $\mathcal{SK}$ and $\mathcal{PK}$ the domains of possible secret keys and public keys, respectively.

[3] We denote $sk_{i,t}$ to be the secret key of user i at time t.

– $sk_{i,t+1} \leftarrow \mathsf{Update}(sk_{i,t}, t)$ is a PPT algorithm which, on input a private key for a certain time period t, outputs a new private key for the time period $t+1$.
– σ'_t=(n,$\mathcal{Y}$,σ)$\leftarrow$ $\mathsf{Sign}(t, n, \mathcal{Y}, sk_{i,t}, M)$ is a PPT algorithm which, on input a certain time period t, group size n, a set $\mathcal{Y}$ of n public keys in $\mathcal{PK}$, a secret key $sk_{i,t}$ whose corresponding public key $pk_i \in \mathcal{Y}$, and a message M, produces a signature σ'_t.
– $1/0 \leftarrow \mathsf{Verify}(M, \sigma'_t, t)$ is a deterministic algorithm which, on input a message-signature pair (M,σ'_t) and a time t returns 1 or 0 for accept or reject, resp. If accept, the message-signature pair is *valid*.

In some cases, we also need to define some system parameters which might be shared by all users, like the group $\mathbb{G}$, the hash function, etc. Some parameters may be generated by a trusted authority, like the group order $N = pq$ whose factorization should not be known by any user. Therefore, we also define this (optional) algorithm:

– $\mathsf{param} \leftarrow \mathsf{Global\ Setup}(1^\lambda)$ is a PPT algorithm which, on input a security parameter $\lambda \in \mathbb{N}$, outputs a system parameter param.

If the algorithm Global Setup exists, then the other algorithms (KeyGen, Sign, Verify and Update) will take param as an additional input. In the following discussion on the security model, we omit Global Setup and param for simplicity. We note that the security model in [14] includes Global Setup (which was named as Init in [14]). We believe that this algorithm is optional and may not be included. We also observed that the Update algorithm in [14] is a deterministic algorithm. We think that the Update algorithm can be probabilistic and it is reflected in our model.

Correctness. We require that $1 \leftarrow \mathsf{Verify}(M, \mathsf{Sign}(t, n, \mathcal{Y}, sk_{i,t}, M), t)$, where $(sk_{i,0}, pk_i) \leftarrow \mathsf{KeyGen}(1^\lambda)$, $pk_i \in \mathcal{Y}$ and

$$sk_{i,t} \leftarrow \underbrace{\mathsf{Update}(\mathsf{Update}(\cdots(\mathsf{Update}(sk_{i,0}, 0)\cdots), t-2), t-1)}_{t\ \mathsf{Update}}.$$

3.2 Forward-Security

The notion of forward security is similar to the unforgeability of standard ring signatures. An adversary should not be able to output a signature $\sigma^*_{t^*} = (n^*, \mathcal{Y}^*, \sigma^*)$ for a time t^* and a message M^* such that $\mathsf{Verify}(M^*, \sigma^*_{t^*}, t^*) = 1$ unless either (1) one of the public keys in $\mathcal{Y}^*$ was generated by the adversary, or (2) a user whose public key is in $\mathcal{Y}^*$ explicitly signed M^* previously (with respect to the same ring $\mathcal{Y}^*$ and time t^*).

Our model is similar to the *unforgeability w.r.t. insider corruption* in [7], which is the strongest security model for unforgeability in [7]. Our model adds the security related to forward security.

Forward-security for FSRS schemes is defined in the following game between a challenger and an adversary $\mathcal{A}$:

1. Setup. The challenger runs KeyGen for l times to obtain keypairs $(sk_{1,0}, pk_1)$, ..., $(sk_{n',0}, pk_{n'})$. The challenger gives $\mathcal{A}$ the set of public keys $\mathcal{S} = (pk_1, \dots, pk_{n'})$.
2. Query. $\mathcal{A}$ may adaptively query the following oracles.
   - $sk_{i,t} \leftarrow \mathcal{CO}(pk_i, t)$. The *Corruption Oracle*, on input a public key $pk_i \in \mathcal{S}$ and a time t, returns the corresponding secret key $sk_{i,t}$.
   - $\sigma'_t \leftarrow \mathcal{SO}(t, n, \mathcal{Y}, pk_i, M)$. The *Signing Oracle*, on input a time t, a group size n, a set $\mathcal{Y}$ of n public keys, a public key $pk_i \in \mathcal{Y}$ and a message M, returns a valid signature σ'_t for time t.
3. Output. $\mathcal{A}$ outputs a signature $\sigma^*_{t^*} = (n^*, \mathcal{Y}^*, \sigma^*)$, a time t^* and a message M^*.

$\mathcal{A}$ wins the game if: (1) Verify(M^*,$\sigma^*_{t^*}$,t^*)=1, (2) $\mathcal{Y}^* \subseteq \mathcal{S}$, (3) for all $pk^*_i \in \mathcal{Y}^*$, there is no $\mathcal{CO}(pk^*_i, t')$ query with time $t' \leq t^*$ and (4) there is no $\mathcal{SO}(t^*, n^*, \mathcal{Y}^*, \cdot, M^*)$ query. We denote by $\mathbf{Adv}^{fs}_{\mathcal{A}}(\lambda)$ the probability of $\mathcal{A}$ winning the game.

Definition 3 (forward-secure). *An FSRS scheme is forward-secure if for all PPT adversary $\mathcal{A}$, $\mathbf{Adv}^{fs}_{\mathcal{A}}(\lambda)$ is negligible.*

3.3 Anonymity

The notion of anonymity is similar to that of standard ring signatures. Simply speaking, an adversary should not be able to tell which member of a ring generated a particular ring signature. (We note that anonymity can be defined in either a computational or an unconditional sense where, informally, anonymity holds for polynomial-time adversaries in the former case, and it holds for all-powerful adversaries in the latter case. For simplicity, we only present the computational version.) In our model, the adversary is also the given the secret keys of all users at time 0, which implies the secret keys of all users in the all time intervals. Our model is similar to the *anonymity against full key exposure* in [7], which is the strongest security model for anonymity in [7].

Anonymity for FSRS schemes is defined in the following game between a challenger and an adversary $\mathcal{A}$:

1. Setup. The challenger runs KeyGen for n' times to obtain keypairs $(sk_{1,0}, pk_1)$, …, $(sk_{n',0}, pk_{n'})$. The challenger gives $\mathcal{A}$ the set of public keys $\mathcal{S} = (pk_1, \ldots, pk_{n'})$ and the set of secret keys $(sk_{1,0}, \ldots, sk_{n',0})$.
2. Query 1. $\mathcal{A}$ may query the signing oracle adaptively.
3. Challenge. $\mathcal{A}$ gives the challenger a time t^*, a group size n^*, a message M, a set $\mathcal{Y}^*$ of n^* public keys, such that two public keys $pk_{i_0}, pk_{i_1} \in \mathcal{S}$ are included in $\mathcal{Y}^*$. The challenger randomly picks a bit $b \in \{0, 1\}$ and runs $\sigma^*_{t^*} \leftarrow \mathsf{Sign}(t^*, n^*, \mathcal{Y}^*, sk_{i_b,t^*}, M^*)$. The challenger gives the signature $\sigma^*_{t^*}$ to $\mathcal{A}$.
4. Query 2. $\mathcal{A}$ may query the signing oracle adaptively.
5. Output. $\mathcal{A}$ returns his guess b'.

$\mathcal{A}$ wins the game if $b' = b$. Define the *advantage* as $\mathbf{Adv}_{\mathcal{A}}^{FS-Anon}(\lambda) = |\Pr[\mathcal{A} \text{ wins}] - 1/2|$ for security parameter λ.

Definition 4 (FS-Anonymity). *A FSRS scheme is anonymous if for any PPT adversary $\mathcal{A}$, $\mathbf{Adv}_{\mathcal{A}}^{FS-Anon}(\lambda)$ is negligible.*

4 Our Proposed Forward Secure Ring Signature Scheme

4.1 Intuition

Our construction is motivated from [20]. We add a binary tree key structure [10] to provide forward security. We first explain the intuition of our binary tree key structure here.

We use binary tree to evolve the secret key. In order to represent $T = 2^\ell$ time periods, we use a full binary tree with depth ℓ. We associate each time period with each leaf of tree from left to right. The leftmost leaf node denotes time period 0 and the rightmost leaf node denotes time period $T - 1$.

At the beginning, it stores the leftmost leaf node, and the right-child nodes ("1" node) starting from its parent node. That is, assume $\ell = 4$, the secret key for $T = 0$ contains the nodes $0000, 0001, 001, 01, 1$. (We put the current node as the first position.)

When it performs the first update, as the current node is a left-child ("0" node), we just delete the current node and move forward to its right-child ("1" node) under the same parent. In the above example, the secret key for $T = 1$ contains the nodes $0001, 001, 01, 1$.

When it performs the next update, as the current node is a right-child ("1" node), it first finds the last "0" node along the path. Extract the corresponding right-child ("1" node), generate the nodes under this node as in the beginning and delete the current node and its parent node. In the above example, the current node is 0001. The last "0" node is 000. Its corresponding right-child is 001. Thus we generate the nodes under the node 001. In this case, we just need to generate the node 0010 and 0011. Finally, the secret key for $T = 2$ contains the nodes $0010, 0011, 01, 1$.

Similarly, the next update is simple, by just deleting 0010 and replaced by 0011. The secret key for $T = 3$ contains the nodes $0011, 01, 1$.

For the next update, find the last "0" node, which is 00 in this case. Extract the corresponding right-child node, which is 01. Perform another node generation process under this node 01 and delete the current node and this node. The new nodes generated are 0100, 0101, 011. Thus the secret key for $T = 4$ contains the nodes $0100, 0101, 011, 1$.

After a few updates, assume the current time period is $T = 7$. That is, the current node is 0111. For this update, first find the last "0" node, which is 0. Extract the corresponding right-child node, which is 1. Perform a node generation process under this node 1. Thus the secret key for $T = 8$ contains the nodes $1000, 1001, 101, 11$.

We hope by presenting this example, readers may now know the intuition and concept of our binary key structure for secret key and key update process.

4.2 Construction

In our ring signature all the users keys must be defined in a group $\mathbb{G}$ of composite order. That group must be set up by a trusted authority, since the factorization of its order N must be kept secret. In addition to setting up the group $\mathbb{G}$, the setup authority must also set up some additional parameters, using a global setup algorithm we now describe.

Global Setup. Let λ, κ be a security parameter and the total number of time periods $T = 2^\ell$. The setup algorithm runs the bilinear group generator $(N = pq, \mathbb{G}, \mathbb{G}_T, e) \leftarrow \mathcal{G}(1^\lambda)$. Let $H : \{0,1\}^* \to \{0,1\}^\kappa$ be a collision resistant hash function. Suppose the group generator $\mathcal{G}$ also gives the generators $g_1, B_0, u, u_1, \ldots, u_\kappa, v, v_1, \ldots, v_\ell \in \mathbb{G}$, $h_1 \in \mathbb{G}_q$ and $\alpha \in \mathbb{Z}_N$. Set $g_2 = g_1^\alpha$ and $h_2 = h_1^\alpha$. The public parameters are $(N, \mathbb{G}, \mathbb{G}_T, e, g_1, g_2, B_0, h_1, h_2, u, u_1, \ldots, u_\kappa, v, v_1, \ldots, v_\ell, H)$. Everyone can check the validity of g_1, g_2, h_1, h_2 using pairings.

KeyGen. Choose a random $s, r_{u_0}, r_{u_1} \in \mathbb{Z}_N$. First compute

$$SK_0 = \Big(g_2^s v^{r_{u_0}}, g_1^{r_{u_0}}, v_2^{r_{u_0}}, \ldots, v_\ell^{r_{u_0}} \Big),$$
$$SK_1 = \Big(g_2^s (v v_1)^{r_{u_1}}, g_1^{r_{u_1}}, v_2^{r_{u_1}}, \ldots, v_\ell^{r_{u_1}} \Big).$$

Then for $k = 2$ to ℓ do
BEGIN

Parse

$$\Big(a_0, a_1, b_k, \ldots, b_\ell\Big) = \Big(g_2^s v^{r'}, g_1^{r'}, v_k^{r'}, \ldots, v_\ell^{r'}\Big) \leftarrow SK_{0^{k-1}},$$

for some $r' \in \mathbb{Z}_N$, where $0^{k-1} = \underbrace{0 \ldots 0}_{k-1}$.

Select $t_0, t_1 \in_R \in \mathbb{Z}_N$ and compute

$$\begin{aligned} SK_{0^k} &= \Big(a_0 v^{t_0}, a_1 g_1^{t_0}, b_{k+1} {v_{k+1}}^{t_0}, \ldots, b_\ell v_\ell^{t_0}\Big) \\ &= \Big(g_2^s v^{r_0}, g_1^{r_0}, {v_{k+1}}^{r_0}, \ldots, v_\ell^{r_0}\Big), \end{aligned}$$

where $r_0 = r' + t_0$, and

$$\begin{aligned} SK_{0^{k-1}1} &= \Big(a_0 v^{t_1} v_k^{t_1} v_k^{r'}, a_1 g_1^{t_1}, b_{k+1} {v_{k+1}}^{t_1}, \ldots, b_\ell v_\ell^{t_1}\Big) \\ &= \Big(g_2^s v^{r_1} v_k^{r_1}, g_1^{r_1}, {v_{k+1}}^{r_1}, \ldots, v_\ell^{r_1}\Big), \end{aligned}$$

where $r_1 = r' + t_1$.

END

Set $pk_i = g_1^s$ be the public key of user i and $sk_{i,0} = \Big\{SK_{0^\ell}, (SK_1, SK_{01}, \ldots, SK_{0^{\ell-1}1})\Big\}$ be the secret key of user i in time period 0.

Update. On input a secret key $sk_{i,j}$ for user i and current time period j, if $j < T$ the user updates the secret key as follow:

1. Let $\langle j \rangle = j_1 \ldots j_\ell$ be the binary representation of j and let $b_l \in \{0, 1\}$ be a bit, for $l = 1, \ldots, \ell$. We also define $j_0 = \epsilon$ and $b_0 = \epsilon$ to be empty strings. Parse

$$\Big\{SK_{\langle j \rangle}, \Big(\{SK_{b_0 \ldots b_{k-1}1}\}_{k=1 \,:\, j_k=0}^{\ell}\Big)\Big\} \leftarrow sk_{i,j}.$$

2. If $j_\ell = 0$, the new secret key is

$$sk_{i,j+1} = \Big\{SK_{j_0 \ldots j_{\ell-1}1}, \Big(\{SK_{b_0 \ldots b_{k-1}1}\}_{k=1 \,:\, j_k=0}^{\ell-1}\Big)\Big\}.$$

3. If $j_\ell = 1$, find the largest integer ϕ such that $j_\phi = 0$. Let $c_l \in \{0,1\}$ be a bit, for $l = 1, \ldots, \phi$. We define $c_0 = j_0 = \epsilon$ to be an empty string and also define $c_0 = j_0 = \epsilon$, $c_1 = j_1, \ldots, c_{\phi-1} = j_{\phi-1}$, $c_\phi = 1$. Then for $k = \phi + 1$ to ℓ do

BEGIN

Parse

$$\Big(a_0, a_1, b_k, \ldots, b_\ell\Big) = \left(g_2^s\Big(v\prod_{\delta=1}^{k-1} v_\delta^{c_\delta}\Big)^{r'}, g_1^{r'}, v_k^{r'}, \ldots, v_\ell^{r'}\right) \leftarrow SK_{c_1\ldots c_{k-1}}, \tag{1}$$

for some $r' \in \mathbb{Z}_N$.

Select $t_0, t_1 \in_R \in \mathbb{Z}_N$ and compute

$$\begin{aligned} SK_{c_1\ldots c_{k-1}0} &= \left(a_0\Big(v\prod_{\delta=1}^{k-1} v_\delta^{c_\delta}\Big)^{t_0}, a_1 g_1^{t_0}, b_{k+1}{v_{k+1}}^{t_0}, \ldots, b_\ell v_\ell^{t_0}\right) \\ &= \left(g_2^s\Big(v\prod_{\delta=1}^{k-1} v_\delta^{c_\delta}\Big)^{r_0}, g_1^{r_0}, {v_{k+1}}^{r_0}, \ldots, v_\ell^{r_0}\right), \end{aligned}$$

where $r_0 = r' + t_0$, and

$$\begin{aligned} SK_{c_1\ldots c_{k-1}1} &= \left(a_0\Big(v\prod_{\delta=1}^{k} v_\delta^{c_\delta}\Big)^{t_1} v_k^{r'}, a_1 g_1^{t_1}, b_{k+1}{v_{k+1}}^{t_1}, \ldots, b_\ell v_\ell^{t_1}\right) \\ &= \left(g_2^s\Big(v\prod_{\delta=1}^{k} v_\delta^{c_\delta}\Big)^{r_1}, g_1^{r_1}, {v_{k+1}}^{r_1}, \ldots, v_\ell^{r_1}\right), \end{aligned}$$

where $r_1 = r' + t_1$.

We define a new bit c_k and set $c_k = 0$ at this stage. (Note that currently only $c_0, \ldots, c_{k-1}$ are well defined, but not c_k. We define c_k in this stage, as in the next loop, k will be increment by 1, this bit will be used as the last bit of the subscript notation in SK in equation(1).)

END

Return

$$sk_{i,j+1} = \left\{SK_{j_0\ldots j_{\phi-1}10^{\ell-\phi}}, \Big(\{SK_{j_0\ldots j_{\phi-1}10^k1}\}_{k=0}^{\ell-\phi-1}\Big),\right.$$
$$\left.\Big(\forall\ \{SK_b\} \in sk_{i,j}\ :\ |b| \le \phi - 1\Big)\right\}.$$

where b is a binary string.

4. Erase $sk_{i,j}$.

<u>Sign.</u> To sign a message $M \in \{0,1\}^*$ in time period j, where $0 \le j < T$, let $\langle j \rangle = j_1 \dots j_\ell$ be the binary representation of j. On behalf of a list of distinct public keys $\mathcal{Y} = \{pk_1, \dots, pk_n\} = \{g_1^{s_1}, \dots, g_1^{s_n}\}$, a user with secret key $sk_{\tau,j}$, where $\tau \in \{1, \dots, n\}$ computes the follow:

1. Compute $(m_1, \dots, m_\kappa) = H(\mathcal{Y}, M, j)$.
2. Without loss of generality, suppose that τ is the index of the actual signer. Define f_i such that $f_i = 1$ if $i = \tau$. Otherwise $f_i = 0$.
3. For $i = 1, \dots, n$, choose $x_i \in_R \mathbb{Z}_N$ and set $C_i = \left(\frac{g_1^{s_i}}{B_0}\right)^{f_i} h_1^{x_i}$ and $\pi_i = \left(\left(\frac{g_1^{s_i}}{B_0}\right)^{2f_i - 1} h_1^{x_i}\right)^{x_i}$. Here the value π_i acts as a proof that C_i is well-formed. Let $C = \prod_{i=1}^n C_i$ and $x = \sum_{i=1}^n x_i$. Then we have $B_0 C = h_1^x g_1^{s_\tau}$.
4. Extract $SK_{\langle j \rangle} \leftarrow sk_{\tau,j}$ and parse
$$(a_0, a_1) \leftarrow \left(g_2^{s_\tau}\Big(v \prod_{\delta=1}^{\ell} v_\delta^{j_\delta}\Big)^{r'}, g_1^{r'}\right) \qquad \text{for some} \qquad r' \in \mathbb{Z}_N.$$
5. Choose $r_\ell, r_\kappa \in_R \mathbb{Z}_N$ and compute
$$S_1 = a_0 \cdot \Big(v \prod_{\delta=1}^{\ell} v_\delta^{j_\delta}\Big)^{r_\ell} \cdot \Big(u \prod_{\delta=1}^{\kappa} u_\delta^{m_\delta}\Big)^{r_\kappa} \cdot h_2^x, \qquad S_2 = g_1^{r_\kappa} \qquad S_3 = a_1 \cdot g_1^{r_\ell}.$$

The signature σ is $\Big(S_1, S_2, S_3, \{C_i, \pi_i\}_{i=1}^n\Big)$.

<u>Verify.</u> To verify a signature σ for a message M, a list of public keys $\mathcal{Y}$ where $|\mathcal{Y}| = n$ and the time period j, first let $\langle j \rangle = j_1 \dots j_\ell$ be the binary representation of j. Then:

1. Verify that no element is repeated in $\mathcal{Y}$ and output invalid otherwise.
2. Compute $(m_1, \dots, m_\kappa) = H(\mathcal{Y}, M, j)$.
3. For $i = 1, \dots, n$, check if
$$e\left(C_i, \frac{C_i}{\left(\frac{g_1^{s_i}}{B_0}\right)}\right) = e(h_1, \pi_i).$$
4. Compute $C = \prod_{i=1}^n C_i$ and check if:
$$e(S_1, g_1) = e\Big(S_2, u \prod_{\delta=1}^{\kappa} u_\delta^{m_\delta}\Big) \cdot e\Big(g_2, B_0 C\Big) \cdot e\Big(S_3, v \prod_{\delta=1}^{\ell} v_\delta^{j_\delta}\Big).$$

Output valid if all equalities hold. Otherwise output invalid.

Correctness:

$$
\begin{aligned}
& e\Big(S_2, u\prod_{\delta=1}^{\kappa} u_\delta^{m_\delta}\Big) \cdot e\Big(g_2, B_0 C\Big) \cdot e\Big(S_3, v\prod_{\delta=1}^{\ell} v_\delta^{j_\delta}\Big) \\
&= e\Big(S_2, u\prod_{\delta=1}^{\kappa} u_\delta^{m_\delta}\Big) \cdot e\Big(g_2, h_1^x g_1^{s_\tau}\Big) \cdot e\Big(S_3, v\prod_{\delta=1}^{\ell} v_\delta^{j_\delta}\Big) \\
&= e\Big(g_1^{r_\kappa}, u\prod_{\delta=1}^{\kappa} u_\delta^{m_\delta}\Big) \cdot e(g_2, g_1^{s_\tau}) \cdot e(g_2, h_1^x) \cdot e\Big(g_1^{r'+r_\ell}, v\prod_{\delta=1}^{\ell} v_\delta^{j_\delta}\Big) \\
&= e\left(g_1, g_2^{s_\tau} \cdot \Big(u\prod_{\delta=1}^{\kappa} u_\delta^{m_\delta}\Big)^{r_\kappa} \cdot \Big(v\prod_{\delta=1}^{\ell} v_\delta^{j_\delta}\Big)^{r'+r_\ell}\right) \cdot e(g_1, h_2^x) \\
&= e(g_1, S_1).
\end{aligned}
$$

Theorem 1. *Our scheme is forward-secure against insider corruption if the CDH assumption holds in* $\mathbb{G}_p$.

Theorem 2. *Our scheme is anonymous if the Subgroup Decision assumption holds in* $\mathbb{G}$.

Details of the security analysis of our scheme can be found in the appendix.

4.3 Comparison

We compare our scheme with some ring signature schemes in the literature in Table 1. Denote the number of users in the group as n and the number of bits of the message as κ. The time of an exponentiation in a group $\mathbb{G}$ is $\exp_{\mathbb{G}}$, and the time of a multiplication in a group $\mathbb{G}$ is $\mathrm{mul}_{\mathbb{G}}$. The time of a pairing operation is pair.

Note that the Schäge-Schwenk scheme [19] is the most efficient ring signature scheme in the standard model. However, it is only secure in the weaker chosen subring model for unforgeability. The first forward secure ring signatures by Liu-Wong [14] is only secure in random oracle model (ROM). Our scheme has a small overhead comparing with the Shacham-Waters scheme [20], while providing the extra forward security (FS) property in the standard model.

5 Conclusion

In this paper, we presented a forward secure ring signature scheme. Our construction is the first in the literature that can be proven secure without

Table 1. Comparison

Scheme	Signature Size	Sign Time	Verify Time	Model	Assumption	FS
Shacham - Waters [20]	$(2n+2)\mathbb{G}$	$3n+2$ $\text{exp}_{\mathbb{G}}$, $2n+\kappa+2$ $\text{mul}_{\mathbb{G}}$	$2n+3$ pair, $2n+\kappa+1$ $\text{mul}_{\mathbb{G}}$ 1 $\text{mul}_{\mathbb{G}_T}$.	Full	CDH, Subgp	$\times$
Schäge - Schwenk [19]	$(n+1)\mathbb{G}$	$n+2$ $\text{exp}_{\mathbb{G}}$, $n+\kappa$ $\text{mul}_{\mathbb{G}}$.	$n+2$ pair, κ $\text{mul}_{\mathbb{G}}$ n $\text{mul}_{\mathbb{G}_T}$.	Chosen Subring	CDH	$\times$
Liu - Wong [14]	$(2n+1)\mathbb{Z}_N$	$3n$ $\text{exp}_{\mathbb{Z}_N}$, n $\text{mul}_{\mathbb{Z}_N}$	$3n$ $\text{exp}_{\mathbb{Z}_N}$, n $\text{mul}_{\mathbb{Z}_N}$	ROM	Factorization	$\surd$
Our scheme	$(2n+3)\mathbb{G}$	$3n+4$ $\text{exp}_{\mathbb{G}}$, $4n+\kappa+3$ $\text{mul}_{\mathbb{G}}$.	$2n+5$ pair, $3n+\kappa+1$ $\text{mul}_{\mathbb{G}}$ 3 $\text{mul}_{\mathbb{G}_T}$.	Full	CDH, Subgp	$\surd$

random oracles. The security relies on the CDH and subgroup decision problem. We believe there are a number of useful applications of forward secure ring signature scheme, such as wireless sensor networks and smart grid system.

References

1. M. Abdalla, S. Miner, and C. Namprempre. Forward-secure threshold signature schemes. In *CT-RSA 2001*, volume 2020 of *Lecture Notes in Computer Science*, pages 441–456. Springer, 2001.
2. M. Abdalla and L. Reyzin. A new forward-secure digital signature scheme. In *ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 116–129. Springer, 2000.
3. R. Anderson. Two remarks on public key cryptology. Technical Report UCAM-CL-TR-549, University of Cambridge, Computer Laboratory, Dec. 2002. Relevant material presented by the author in an invited lecture at CCS '97.
4. M. Bellare, A. Boldyreva, and A. Palacio. An uninstantiable random-oracle-model scheme for a hybrid-encryption problem. In *EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 171–188. Springer, 2004.
5. M. Bellare and S. Miner. A forward-secure digital signature scheme. In M. J. Wiener, editor, *Crypto '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 431–448. Springer, 1999.
6. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73. ACM Press, 1993.
7. A. Bender, J. Katz, and R. Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. In *TCC*, volume 3876 of *Lecture Notes in Computer Science*, pages 60–79. Springer, 2006.
8. D. Boneh, E.-J. Goh, and K. Nissim. Evaluating 2-dnf formulas on ciphertexts. In *TCC*, volume 3378 of *Lecture Notes in Computer Science*, pages 325–341. Springer, 2005.

9. R. Canetti, O. Goldreich, and S. Halevi. The Random Oracle Methodology, Revisited. In *STOC*, pages 209–218, 1998.
10. R. Canetti, S. Halevi, and J. Katz. A forward-secure public-key encryption scheme. In *EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 255–271. Springer, 2003.
11. S. S. Chow, J. K. Liu, V. K. Wei, and T. H. Yuen. Ring signatures without random oracles. In *ASIACCS 06*, pages 297–302. ACM Press, 2006.
12. G. Itkis and L. Reyzin. Forward-secure signatures with optimal signing and verifying. In *CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 332–354. Springer, 2001.
13. H. Krawczyk. Simple forward-secure signatures from any signature scheme. In *the 7th ACM Conference on Computer and Communications Security*, pages 108–115. ACM Press, 2000.
14. J. K. Liu and D. S. Wong. Solutions to key exposure problem in ring signature. *I. J. Network Security*, 6(2):170–180, 2008.
15. T. Malkin, D. Micciancio, and S. Miner. Efficient generic forward-secure signatures with an unbounded number of time periods. In *EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 400–417. Springer, 2002.
16. Microsoft. Conserve Energy, Save Money - Microsoft Hohm, 2009. http://www.microsoft-hohm.com/.
17. N. I. of Standards and Technology. Nist ir 7628: Guidelines for smart grid cyber security. Technical report, http://csrc.nist.gov/publications/PubsNISTIRs.html.
18. R. L. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In *ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 552–565. Springer, 2001.
19. S. Schäge and J. Schwenk. A cdh-based ring signature scheme with short signatures and public keys. In *Financial Cryptography*, volume 6052 of *Lecture Notes in Computer Science*, pages 129–142. Springer, 2010.
20. H. Shacham and B. Waters. Efficient ring signatures without random oracles. In *Public Key Cryptography*, volume 4450 of *Lecture Notes in Computer Science*, pages 166–180. Springer, 2007.
21. D. X. Song. Practical forward secure group signature schemes. In *the 8th ACM Conference on Computer and Communications Security*, pages 225–234. ACM Press, 2001.
22. B. Waters. Efficient identity-based encryption without random oracles. In *EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 114–127. Springer, 2005.
23. J. Xu, Z. Zhang, and D. Feng. A ring signature scheme using bilinear pairings. In *WISA 2004*, volume 3325 of *Lecture Notes in Computer Science*, pages 163–172. Springer, 2004.
24. J. Yu, R. Hao, F. Kong, X. Cheng, J. Fan, and Y. Chen. Forward-secure identity-based signature: Security notions and construction. *Information Sciences*, 181(3):648–660, Feb. 2011.

A Security Analysis

Theorem 3. *Assume the CDH assumption holds in $\mathbb{G}_p$. If there is a PPT adversary $\mathcal{A}$ that can break the forward-security against insider corruption with non-negligible advantage ϵ' and running time t', we can construct another PPT $\mathcal{B}$ that can solve the CDH problem with probability*

$\epsilon \geq \frac{\epsilon'}{8T(\kappa+1)q_s n^*}(1 - \frac{q_c}{n'})$, using time at most $t' + \mathcal{O}\Big(T\ell^2(q_c + n'q_s)\Big)t_e + \mathcal{O}\Big(T\ell^2(q_c + n'q_s) + (\kappa+\ell)n'q_s\Big)t_m$, where q_s, q_c are the numbers of $\mathcal{SO}$ and $\mathcal{CO}$ respectively, n' is the number of public keys $\mathcal{A}$ allows to have, n^* is the number of users included in the forged signature produced by $\mathcal{A}$, T is the total number of time period, κ is the number of bits the hash function H outputs, t_e is the time for running an exponentiation and t_m is the time for running a multiplication.

Proof. Setup. The simulator $\mathcal{B}$ runs the bilinear group generator $(N = pq, \mathbb{G}, \mathbb{G}_T, e) \leftarrow \mathcal{G}(1^\lambda)$. $\mathcal{B}$ is given the CDH problem instance $(g, g^a, g^b) \in \mathbb{G}_p^3$ and is asked to output g^{ab}. $\mathcal{B}$ first sets an integer, $\mu = 4q_s$, and chooses an integer, κ', uniformly at random between 0 and κ. $\mathcal{B}$ picks $x', x_1, \ldots, x_\kappa$ uniformly at random between 0 and $\mu - 1$. $\mathcal{B}$ randomly picks a $\gamma \in \mathbb{Z}_N$ and sets $z_1 = g^{\frac{p\gamma}{q}}$. Since $g \in \mathbb{G}_p$, z_1 is in $\mathbb{G}_q$. Also z_1^b can be computed from g^b.

$\mathcal{B}$ randomly picks a generator $h_1 \in \mathbb{G}_q$. $\mathcal{B}$ randomly picks $y', y_1, \ldots, y_\kappa$, $\tilde{v}, \tilde{v}_1, \ldots, \tilde{v}_\ell$, $\alpha, \beta \in \mathbb{Z}_N$ and sets

$$g_1 = gz_1, \quad g_2 = g^a z_1^\alpha, \quad u = g_2^{N-\kappa'\mu+x'} g^{y'}, \quad u_1 = g_2^{x_1} g^{y_1}, \ldots, u_\kappa = g_2^{x_\kappa} g^{y_\kappa},$$

$$h_2 = h_1^\alpha, \quad B_0 = h_1^\beta, \qquad v = {g_1}^{\tilde{v}}, \quad v_1 = {g_1}^{\tilde{v}_1}, \quad \ldots, \quad v_\ell = {g_1}^{\tilde{v}_\ell}.$$

Note that

$$e(g_1, h_2) = e(z_1, h_1^\alpha) = e(z_1^\alpha, h_1) = e(g_2, h_1,)$$

since $e(g, h_1) = 1$. Finally, $\mathcal{B}$ randomly chooses a collision resistant hash function $H : \{0,1\}^* \to \{0,1\}^\kappa$.

Then $\mathcal{B}$ gives the public parameters $(N, \mathbb{G}, \mathbb{G}_T, e, g_1, g_2, B_0, h_1, h_2, u, u_1, \ldots, u_k, v, v_1, \ldots, v_\ell, H)]$ to the adversary $\mathcal{A}$.

We define j^* be the breakin period such that $\mathcal{A}$ is not allowed to query $\mathcal{CO}$ for any public key included in $\mathcal{Y}^*$ (the set of public keys of the forged signature output by $\mathcal{A}$), while there is no limitation for time input parameter $j' \geq j^*$. $\mathcal{A}$ is allowed to choose any $j^* \leq T$. $\mathcal{B}$ needs to guess the breakin period j^* chosen by $\mathcal{A}$. $\mathcal{B}$ randomly chooses $\hat{j}$, $1 < \hat{j} \leq T$, hoping that the breakin period falls at $\hat{j}$ or later, so that the forgery will be for a time period earlier than $\hat{j}$.

Assume $\mathcal{B}$ picks $\tau \in \{1, \ldots, n'\}$ as his guess for the challenge signer. For $i = 1, \ldots, n'$, $\mathcal{B}$ picks random $s_i \in \mathbb{Z}_N$ and sets:

$$pk_i = \begin{cases} g_1^{s_i} & \text{if } i \neq \tau, \\ g^b z_1^b & \text{if } i = \tau. \end{cases}$$

$\mathcal{B}$ stores the set of public keys $\mathcal{S} = \{pk_i\}_{i=1}^{n'}$ and gives them to $\mathcal{A}$.

For a message $m = \{m_1, \ldots, m_\kappa\}$, we define

$$F(m) = (N - \mu\kappa') + x' + \sum_{i=1}^{\kappa} x_i m_i, \quad J(m) = y' + \sum_{i=1}^{\kappa} y_i m_i.$$

Oracle Simulation. $\mathcal{B}$ simulates the oracles as follows:

– $\mathcal{CO}(pk_i, j)$: If $pk_i \notin \mathcal{S}$ or $i = \tau$, $\mathcal{B}$ declares failure and exits. Otherwise, $\mathcal{B}$ chooses random $r_{u_0}, r_{u_1} \in \mathbb{Z}_N$, computes $\tilde{\alpha} = g_2^{s_i}$ and

$$SK_0 = \left(\tilde{\alpha} v^{r_{u_0}}, g_1^{r_{u_0}}, v_2^{r_{u_0}}, \ldots, v_\ell^{r_{u_0}}\right),$$

$$SK_1 = \left(\tilde{\alpha} (v v_1)^{r_{u_1}}, g_1^{r_{u_1}}, v_2^{r_{u_1}}, \ldots, v_\ell^{r_{u_1}}\right).$$

Then $\mathcal{B}$ computes the $sk_{i,0}$ as secret key for user i in the time period 0 according to the KeyGen algorithm. If $j \neq 0$, $\mathcal{B}$ performs Update algorithm until it gets $sk_{i,j}$, the secret key for the time period j. $\mathcal{B}$ outputs $sk_{i,j}$.

– $\mathcal{SO}(j, n, \mathcal{Y}, pk_{i'}, M)$: On input a message M, a set of n public keys $\mathcal{Y} = \{pk_{i'}\}_{i'=1}^{n}$ where $\mathcal{Y} \subseteq \mathcal{S}$, and the public key of a signer $pk_{i'}$, $\mathcal{B}$ calculates $(C_{i'}, \pi_{i'})$ according to the Sign algorithm. Note that no secret key is required to generate $(C_{i'}, \pi_{i'})$. Then we have $B_0 C = h_1^x g_1^{s_i}$. Denote $m = H(\mathcal{Y}, M, j)$. We also write m into κ bits $\{m_1, \ldots, m_\kappa\}$. If

$$x' + \sum_{i=1}^{\kappa} x_i m_i \equiv 0 \mod \mu,$$

then $\mathcal{B}$ aborts. If $i' \neq \tau$, $\mathcal{B}$ calculates all $(S_{1,i}, S_{2,i}, S_{3,i})$ according to the Sign algorithm. If $i' = \tau$, $\mathcal{B}$ chooses random $r_\tau, r_\ell \in \mathbb{Z}_N$ and calculates

$$S_{1,\tau} = (g^b)^{\frac{-J(m)}{F(m)}} \left(u \prod_{\delta=1}^{\kappa} u_\delta^{m_\delta}\right)^{r_\tau} \left(v \prod_{\delta=1}^{\ell} v_\delta^{j_\delta}\right)^{r_\ell},$$

$$S_{2,\tau} = (g^b z_1^b)^{\frac{-1}{F(m)}} (g z_1)^{r_\tau}, \qquad S_{3,\tau} = g_1^{r_\ell}.$$

Let $\bar{r} = r_\tau - \frac{b}{F(m)}$, then

$$\begin{aligned} S_{1,\tau} &= (g^b)^{\frac{-J(m)}{F(m)}} \Big(u \prod_{\delta=1}^{\kappa} u_\delta^{m_\delta}\Big)^{r_\tau} \Big(v \prod_{\delta=1}^{\ell} v_\delta^{j_\delta}\Big)^{r_\ell} \\ &= g_2^b \Big(g_2^{F(m)} g^{J(m)}\Big)^{r_\tau - \frac{b}{F(m)}} \Big(v \prod_{\delta=1}^{\ell} v_\delta^{j_\delta}\Big)^{r_\ell} \\ &= g_2^b \Big(u \prod_{\delta=1}^{\kappa} u_\delta^{m_\delta}\Big)^{\bar{r}} \Big(v \prod_{\delta=1}^{\ell} v_\delta^{j_\delta}\Big)^{r_\ell}. \end{aligned}$$

The simulator will be able to perform this computation if and only if $F(m) \neq 0 \mod N$. For ease of analysis the simulator will only continue in the sufficient condition where $x' + \sum_{i=1}^{\kappa} x_i m_i \neq 0 \mod \mu$. (If we have $x' + \sum_{i=1}^{\kappa} x_i m_i \neq 0 \mod \mu$, this implies $F(m) \neq 0 \mod N$ since we can assume $N > \kappa\mu$ for any reasonable values of N, κ, and μ).

Finally, $\mathcal{B}$ calculates the rest of the signature according to the Sign algorithm.

Output. Assume $\mathcal{A}$ chooses a breakin period $j^* \leq \hat{j}$. That is, the forged signature σ^* is valid for time period $j < \hat{j}$. $\mathcal{A}$ returns $(n^*, \mathcal{Y}^*, j^*, M^*, \sigma^*)$. Denote

$$m^* = (m_1^*, \ldots, m_\kappa^*) = H(\mathcal{Y}^*, M^*, j^*) \qquad \text{and} \qquad \langle j^* \rangle = j_1^* \ldots j_\ell^*.$$

Note that this hash value is different from previous m in various $\mathcal{SO}$ queries, since $(j^*, n^*, \mathcal{Y}^*, M^*)$ cannot be the input of previous $\mathcal{SO}$ queries and H is a collision resistant hash function. If

$$pk_\tau \notin \mathcal{Y}^* \qquad \text{and} \qquad x' + \sum_{i=1}^{\kappa} x_i m_i^* \neq \mu\kappa'.$$

then $\mathcal{B}$ aborts. Otherwise, WLOG, we assume that pk_τ is at the position τ of the signature σ^*. Since σ^* is a valid signature, then

$$e(S_1^*, g_1) = e\Big(S_2^*, u \prod_{\delta=1}^{\kappa} u_\delta^{m_\delta^*}\Big) \cdot e\Big(g_2, B_0 \prod_{\delta=1}^{n^*} C_\delta^*\Big) \cdot e\Big(S_3^*, v \prod_{\delta=1}^{\ell} v_\delta^{j_\delta^*}\Big), \tag{2}$$

$$e\left(C_i^*, \frac{C_i^*}{\left(\frac{pk_i}{B_0}\right)}\right) = e(h_1, \pi_i^*). \tag{3}$$

for $i = 1, \ldots, n^*$. Since $\hat{e}(h_1, \pi_i^*)$ has order q in $\mathbb{G}_T$, therefore either C_i^* or $\frac{B_0 C_i^*}{pk_i}$ has order q from equation (3). $\mathcal{B}$ checks if $(C_i^*)^q = 0$. If it is true, then C_i^* has order q and then $\mathcal{B}$ sets $f_i = 0$. Otherwise, $\frac{B_0 C_i^*}{pk_i}$ has order q and then $\mathcal{B}$ sets $f_i = 1$. It follows that $C_i^* = (\frac{pk_i}{B_0})^{f_i} z_1^{r'_i}$ for some unknown r'_i, no matter $f_i = 0/1$. If $f_\tau = 0$, $\mathcal{B}$ aborts.

Let $\delta' \in \mathbb{Z}_N$ such that $\delta' = 0 \mod q$ and $\delta' = 1 \mod p$. If we raise equation (2) to the δ'-th power, then we have

$$e(S_1^*, g_1)^{\delta'} = e\Big(S_2^*, u \prod_{\delta=1}^{\kappa} u_\delta^{m_\delta^*}\Big)^{\delta'} \cdot e\Big(g_2, B_0 \prod_{i|pk_i \in \mathcal{Y}^*} C_i^*\Big)^{\delta'} \cdot e\Big(S_3^*, v \prod_{\delta=1}^{\ell} v_\delta^{j_\delta^*}\Big)^{\delta'},$$

$$e(S_1^*, gz_1)^{\delta'} = e\Big(S_2^*, g^{J(m^*)}\Big)^{\delta'} \cdot e\Big(g^a z_1^\alpha, B_0 \prod_{i|pk_i \in \mathcal{Y}^*} (\frac{pk_i}{B_0})^{f_i} z_1^{r'_i}\Big)^{\delta'} \cdot e\Big(S_3^*, (gz_1)^{\tilde{v} + \sum_{\delta=1}^{\ell} \tilde{v}_\delta j_\delta^*}\Big)^{\delta'}, \tag{4}$$

$$e(S_1^*, g)^{\delta'} = e\Big(S_2^*, g^{J(m^*)}\Big)^{\delta'} \cdot e\Big(g^a, B_0 \prod_{i|pk_i \in \mathcal{Y}^*} (\frac{pk_i}{B_0})^{f_i}\Big)^{\delta'} \cdot e\Big(S_3^*, g^{\tilde{v} + \sum_{\delta=1}^{\ell} \tilde{v}_\delta j_\delta^*}\Big)^{\delta'}, \tag{5}$$

$$e(S_1^*, g)^{\delta'} = e\Big(S_2^*, g^{J(m^*)}\Big)^{\delta'} \cdot e\Big(g^a, B_0 (\frac{pk_\tau}{B_0}) \prod_{i|pk_i \in \mathcal{Y}^*, i \neq \tau} (g^{s_i})^{f_i}\Big)^{\delta'} \cdot e\Big(S_3^*, g^{\tilde{v} + \sum_{\delta=1}^{\ell} \tilde{v}_\delta j_\delta^*}\Big)^{\delta'}, \tag{6}$$

$$e(S_1^*, g)^{\delta'} = e\Big(S_2^*, g^{J(m^*)}\Big)^{\delta'} \cdot e\Big(g^a, g^b\Big)^{\delta'} \cdot e\Big(S_3^*, g^{\tilde{v} + \sum_{\delta=1}^{\ell} \tilde{v}_\delta j_\delta^*}\Big)^{\delta'}. \tag{7}$$

For equation (4), note that

$$u \prod_{j=1}^{\kappa} u_j^{m_j^*} = g_2^{F(m^*)} g^{J(m^*)} = g^{J(m^*)}, \qquad \text{since} \qquad x' + \sum_{i=1}^{\kappa} x_i m_i^* = \mu \kappa'.$$

For equation (5), as $\delta' = 0 \mod q$, all terms which are in $\mathbb{G}_q$ are cancelled. Thus

$$C_i^{*\delta'} = \left(\Big(\frac{pk_i}{B_0}\Big)^{f_i} z_1^{r'_i}\right)^{\delta'} = \Big(\frac{pk_i}{B_0}\Big)^{f_i \delta'}, \qquad \text{since } z_1 \in \mathbb{G}_q.$$

From equation (7), we can see that

$$S_1^* = S_2^{*J(m^*)} \cdot g^{ab} \cdot S_3^{*\tilde{v} + \sum_{\delta=1}^{\ell} \tilde{v}_\delta j_\delta^*}.$$

Therefore $\mathcal{B}$ can output

$$A = S_1^* \cdot (S_2^*)^{-J(m^*)} \cdot (S_3^*)^{-(\tilde{v} + \sum_{\delta=1}^{\ell} \tilde{v}_\delta j_\delta^*)},$$

as the solution to the CDH problem.

Probability Analysis. Following the probability analysis of Waters signature [22], the probability of $F(m) \neq 0 \mod N$ during signing oracle query and $x + \sum_{i=1}^{\kappa} x_i m_i^* = \mu\kappa'$ is at least $\frac{1}{8(\kappa+1)q_s}$. The probability of not asking $pk_\mathcal{T}$ in the corruption oracle is $1 - \frac{q_c}{n'}$. The probability of $f_\mathcal{T} = 1$ in the output phase is $\frac{1}{n^*}$. In additional, $\mathcal{S}$ also needs to guess the breaking period correctly. The probability is at least $\frac{1}{T}$. Therefore if $\mathcal{A}$ outputs a forged signature with probability ϵ', $\mathcal{B}$ solves the CDH problem with probability $\epsilon \geq \frac{\epsilon'}{8T(\kappa+1)q_s n^*}(1 - \frac{q_c}{n'})$, where q_s, q_c are the numbers of $\mathcal{SO}$ and $\mathcal{CO}$ respectively, n' is the number of public keys $\mathcal{A}$ allows to have, n^* is the number of users included in the forged signature produced by $\mathcal{A}$, T is the total number of time period and κ is the number of bits the hash function H outputs.

Time Analysis. The running time is dominated by the operations of exponentiation and multiplication in the corruption oracle and the signing oracle. There are at most $\mathcal{O}(T\ell^2 q_c)$ exponentiations and $\mathcal{O}(T\ell^2 q_c)$ multiplications in all corruption oracle queries. There at at most $\mathcal{O}(T\ell^2 n' q_s)$ exponentiations and $\mathcal{O}((T\ell^2 + \kappa + \ell)n' q_s)$ multiplications in all signing oracle queries. As the total running time for $\mathcal{A}$ is t', the running time for $\mathcal{B}$ is at most $t' + \mathcal{O}\Big(T\ell^2(q_c + n'q_s)\Big)t_e + \mathcal{O}\Big(T\ell^2(q_c + n'q_s) + (\kappa + \ell)n'q_s\Big)t_m$. where q_s, q_c are the numbers of $\mathcal{SO}$ and $\mathcal{CO}$ respectively, n' is the number of public keys $\mathcal{A}$ allows to have, t_e is the time for running an exponentiation and t_m is the time for running a multiplication. □

Theorem 4. *Assume the Subgroup Decision assumption holds in $\mathbb{G}$. If there is a PPT adversary $\mathcal{A}$ that can break the anonymity with non-negligible advantage ϵ' and running time t', we can construct another PPT $\mathcal{B}$ that can solve the Subgroup Decision problem with advantage $\epsilon \geq \epsilon'$ and running time at most $t' + \mathcal{O}(T\ell^2 q_s)t_e + \mathcal{O}(T\ell^2 n' q_s + (\kappa + \ell)n' q_s)t_m$. where q_s is the numbers of $\mathcal{SO}$ respectively, n' is the number of public keys $\mathcal{A}$ allows to have, T is the total number of time period, κ is the number of bits the hash function H outputs, t_e is the time for running an exponentiation and t_m is the time for running a multiplication.*

Proof. Setup. The simulator $\mathcal{B}$ is given the subgroup decision problem instance $(N, \mathbb{G}, \mathbb{G}_T, e, g, h)$. $\mathcal{B}$ is asked to determine whether $h \in \mathbb{G}$ or

$h \in \mathbb{G}_q$. $\mathcal{B}$ randomly picks the generators $u, u_1, \ldots u_\kappa, v, v_1, \ldots, v_\ell, B_0 \in \mathbb{G}$ and $\alpha \in \mathbb{Z}_N$. $\mathcal{B}$ sets

$$g_1 = g, \quad g_2 = g_1^\alpha, \quad h_1 = h, \quad h_2 = h^\alpha.$$

Finally, $\mathcal{B}$ randomly chooses a collision resistant hash function $H : \{0,1\}^* \to \{0,1\}^\kappa$. Then $\mathcal{B}$ gives the public parameters $(N, \mathbb{G}, \mathbb{G}_T, e, g_1, g_2, B_0, h_1, h_2, u, u_1, \ldots, u_k, v, v_1, \ldots, v_\ell, H)$ to the adversary $\mathcal{A}$.

For $i = 1, \ldots, n'$, $\mathcal{B}$ generates the public key and secret key of each user according to the KeyGen algorithm. $\mathcal{B}$ stores the set of public keys and secret keys $\{pk_i, sk_{i,0}\}_{i=1}^{n'}$ and sends them to $\mathcal{A}$.

Oracle Simulation. $\mathcal{B}$ simulates the $\mathcal{SO}(n, j, \mathcal{Y}, pk_i, M)$ by running the Sign algorithm honestly.

Challenge. At some point, $\mathcal{A}$ outputs a message M^*, a set of n^* public keys $\mathcal{Y}^*$, two indexes $i_0, i_1 \in \{1, \ldots, n\}$ such that $pk_{i_0}, pk_{i_1} \in \mathcal{Y}^*$ and a time j^*. $\mathcal{B}$ picks a random bit $b \in \{0,1\}$ and uses the secret keys of sk_{i_b,j^*} to run the Sign algorithm to obtain the signature σ^*. $\mathcal{B}$ gives σ^* to $\mathcal{A}$.

Output. $\mathcal{A}$ continues to query and finally outputs a bit b'. If $b' = b$, then $\mathcal{B}$ outputs $h \in \mathbb{G}_q$. Otherwise, $\mathcal{B}$ outputs $h \in \mathbb{G}$.

Analysis. Suppose the challenge signature is $(S_1^*, S_2^*, S_3^*, \{C_i^*, \pi_i^*\}_{i=1}^{n^*})$ and $\mathcal{Y}^* = \{pk_1^*, \ldots, pk_{n^*}^*\}$. If h_1 is a generator of $\mathbb{G}$, there exist $x_i, \bar{x}_i \in \mathbb{Z}_N$ such that $C_i^* = (\frac{pk_i^*}{B_0})h_1^{x_i} = h_1^{\bar{x}_i}$. Then $x_i, \bar{x}_i$ correspond to the case $f_i^* = 0$ or 1 respectively. Denote by $(\pi_i^* | f_i^* = b)$ the value of π_i^* if f_i is set to $b \in \{0,1\}$. Then

$$\begin{aligned}(\pi_i^* | f_i^* = 0) &= \big((\frac{pk_i^*}{B_0})h_1^{x_i}\big)^{x_i} = (h_1^{\bar{x}_i})^{x_i} = (h_1^{x_i})^{\bar{x}_i} \\ &= \big((\frac{pk_i^*}{B_0})^{-1}h_1^{\bar{x}_i}\big)^{\bar{x}_i} = (\pi_i^* | f_i^* = 1).\end{aligned}$$

Therefore $\{C_i^*, \pi_i^*\}_{i=1}^{n^*}$ has no information about the real signer if $h \in \mathbb{G}$.

On the other hand, S_2^* and S_3^* are computed by random numbers only and do not have information about the real signer. Finally, S_1^* is determined by the verification equation

$$e(S_1^*, g_1) = e\Big(S_2^*, u \prod_{\delta=1}^{\kappa} u_\delta^{m_\delta^*}\Big) \cdot e\Big(g_2, B_0 \prod_{i=1}^{n^*} C_i^*\Big) \cdot e\Big(S_3^*, v \prod_{\delta=1}^{\ell} v_\delta^{j_\delta^*}\Big).$$

Hence, it leaks no useful information about the public key pk_{i_b}. Therefore if $\mathcal{A}$ wins the game, $\mathcal{B}$ outputs $h \in \mathbb{G}_q$. The advantage of $\mathcal{B}$ is at least ϵ'. The running time of $\mathcal{B}$ should be similar to the proof of Theorem 3 and we skip here. □