2020

Lattice Blind Signatures with Forward Security

Quoc Huy Le
University of Wollongong, qhl576@uowmail.edu.au

Dung Hoang Duong
University of Wollongong, hduong@uow.edu.au

Willy Susilo
University of Wollongong, wsusilo@uow.edu.au

Ha Tran

Viet Cuong Trinh

See next page for additional authors

Lattice Blind Signatures with Forward Security

Abstract

© 2020, Springer Nature Switzerland AG. Blind signatures play an important role in both electronic cash and electronic voting systems. Blind signatures should be secure against various attacks (such as signature forgeries). The work puts a special attention to secret key exposure attacks, which totally break digital signatures. Signatures that resist secret key exposure attacks are called forward secure in the sense that disclosure of a current secret key does not compromise past secret keys. This means that forward-secure signatures must include a mechanism for secret-key evolution over time periods. This paper gives a construction of the first blind signature that is forward secure. The construction is based on the SIS assumption in the lattice setting. The core techniques applied are the binary tree data structure for the time periods and the trapdoor delegation for the key-evolution mechanism.

Publication Details

Le, Q., Duong, H., Susilo, W., Tran, H., Trinh, V., Pieprzyk, J. & Plantard, T. (2020). Lattice Blind Signatures with Forward Security. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 12248 LNCS 3-22.

Authors

Quoc Huy Le, Dung Hoang Duong, Willy Susilo, Ha Tran, Viet Cuong Trinh, Josef Pieprzyk, and Thomas Plantard

Lattice Blind Signatures with Forward Security

Huy Quoc Le[1,4(✉)], Dung Hoang Duong[1(✉)], Willy Susilo[1], Ha Thanh Nguyen Tran[2], Viet Cuong Trinh[3], Josef Pieprzyk[4,5], and Thomas Plantard[1]

[1] Institute of Cybersecurity and Cryptology, School of Computing and Information Technology, University of Wollongong
Northfields Avenue, Wollongong NSW 2522, Australia
qhl576@uowmail.edu.au, {hduong, wsusilo, thomaspl}@uow.edu.au
[2] Department of Mathematical and Physical Sciences, Concordia University of Edmonton, 7128 Ada Blvd NW, Edmonton, AB T5B 4E4, Canada
hatran1104@gmail.com
[3] Faculty of Information and Communication Technology, Hong Duc University, 565 Quang Trung, Thanh Hoa, Vietnam
trinhvietcuong@hdu.edu.vn
[4] CSIRO Data61, Sydney, NSW, Australia.
[5] Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland.
Josef.Pieprzyk@data61.csiro.au

Abstract. Blind signatures play an important role in both electronic cash and electronic voting systems. Blind signatures should be secure against various attacks (such as signature forgeries). The work puts a special attention to secret key exposure attacks, which totally break digital signatures. Signatures that resist secret key exposure attacks are called forward secure in the sense that disclosure of a current secret key does not compromise past secret keys. This means that forward-secure signatures must include a mechanism for secret-key evolution over time periods.
This paper gives a construction of the *first* blind signature that is forward secure. The construction is based on the SIS assumption in the lattice setting. The core techniques applied are the binary tree data structure for the time periods and the trapdoor delegation for the key-evolution mechanism.

Key words: Key Exposure, Forward Security, Blind Signatures, Lattice-based Cryptography, SIS Assumption

1 Introduction

Key exposure is one of most serious dangers for both secret and public key cryptography. When secret keys are disclosed, cryptographic systems using them are completely broken. Fortunately, there are some solutions that can be used to mitigate secret-key exposure. They are summarized in [6]. Among many possible solutions, forward security seems to be the most promising when trying to minimize a damage caused by secret-key disclosure.

For cryptographic protocols, forward security guarantees that even if the current session key is compromised by an adversary, she gets no information about previous session keys. This means that past sessions are still secure. The notion of *forward security* has been coined by Günther in [15] and later used in [12] to evaluate security of authenticated key-exchange protocols. Note that the authors of [12] and [15] call it *forward secrecy*. Ross Anderson in [5] extends the notion for digital signatures.

Blind signatures, introduced by Chaum [10], allow users to obtain message signatures from a signer without leaking information about message contents. Blind signatures are indispensable in many applications such as electronic cash [23, Section 1] and electronic voting protocols [18]. For such security-critical applications, one would expect blind signatures to be resistant against key disclosure. An obvious solution is to incorporate forward security into blind signatures. There are many works such as [11,13,17] that follow this line of investigation. All solutions published so far rely on number-theoretic assumptions and consequently are insecure against quantum adversaries.

Related Works. Bellare and Milner investigate secret-key exposure of digital signatures in their Crypto99 paper [6]. They formulate a security model and define forward-secure digital signatures. They also design their forward-secure signature assuming intractability of integer factorization. Abdalla and Reyzin [1], and Itkis and Reyzin [16] improve efficiency the Bellare-Miner signature. The work of Duc et al. [13] is the first, which investigates forward security in the context of blind signatures. The authors of [13] adopt the definition and security model from [6] to forward-secure blind signatures. Their blind signature provides forward-secure unforgeability assuming intractability of the strong RSA problem and access to random oracle. Their security proof exploits the forking lemma by Pointcheval and Stern [23]. Later, Chow et al. [11] design forward-secure blind signature using bilinear pairings. Jia et al. [17] describe a forward-secure blind signature that is also based on bilinear pairings. Boyd and Gellert [7] give a comprehensive survey of methods of incorporating forward security to different cryptographic primitives. They also unify different approaches to forward security by generalising the notion and its terminology.

Our Contributions and Approach. Thanks to its quantum resistance, lattice-based cryptography is attracting more and more attention from the research community. However, there is no lattice-based construction of forward-secure blind signatures. Our work fills the gap. We construct *the first forward-secure blind signature in the lattice setting*. Forward security is proven in the random oracle model assuming intractability of the average case of short integer solution (SIS). We also use the rewinding (forking lemma) argument.

Inspired by the works [19,24,25], our signature is designed using the 3-move Fiat-Shamir transformation. To achieve blindness, the rejection sampling technique is applied (see Section 2). Thus, an extra move is needed to ensure that a final signature is valid. In order to achieve forward security, we exploit both a

binary tree structure for lattice-based schemes introduced in [9] and a trapdoor delegation from [2,9].

To obtain forward-secure signature, we need a mechanism that permits for a secret-key update between two time intervals. For this purpose, we use a binary tree of the depth ℓ, whose leaves are labelled from left to right by consecutive time intervals $t = 0$ up to $t = \tau - 1$, where $\tau = 2^\ell$ is the total number of time intervals. To generate the public key and the initial secret key, we choose random matrices $A_j^{(0)}$, $A_j^{(1)}$ for $j \in [\ell]$ together with a matrix/trapdoor pair (A_0, T_{A_0}). Now, for any node $w^{(i)} = (w_1, \cdots, w_i) \in \{0,1\}^i$, we build up a concatenated matrix of form $F_{w^{(i)}} = [A_0 \| A_1^{(w_1)} \| A_i^{(w_i)}]$. Then, we can compute a trapdoor for $\Lambda_q^\perp(F_{w^{(i)}})$ using T_{A_0}. If the node $w^{(k)}$ is the ancestor of the node $w^{(i)}$, then we can obtain a trapdoor for $\Lambda_q^\perp(F_{w^{(i)}})$ from a trapdoor for $\Lambda_q^\perp(F_{w^{(k)}})$. However, one cannot get a trapdoor for $\Lambda_q^\perp(F_{w^{(k)}})$ from a trapdoor of $\Lambda_q^\perp(F_{w^{(i)}})$. This is the main idea behind the key evolution (key update) mechanism.

2 Preliminaries

For a positive integer ℓ, $[\ell]$ stands for the set $\{1, \cdots, \ell\}$. For a vector $\mathbf{c}$ and a matrix S, $\mathbf{c}[i]$ and $S[i]$ represent the i-th element of $\mathbf{c}$ and the i-th column of S, respectively.

Lattices. Integer lattices are discrete subgroups of $\mathbb{Z}^m$. Formally, a lattice $\mathcal{L}$ in $\mathbb{Z}^m$ is defined as $\mathcal{L} = \mathcal{L}(B) := \{\sum_{i=1}^n \mathbf{b}_i x_i : x_i \in \mathbb{Z}, \forall i = 1, \cdots, n\} \subseteq \mathbb{Z}^m$, where $B = [\mathbf{b}_1, \cdots, \mathbf{b}_n] \in \mathbb{Z}^{m \times n}$ is called a basis of $\mathcal{L}$, and $\mathbf{b}_i$'s are column vectors. We call n the rank of $\mathcal{L}$. We say $\mathcal{L}$ is a full rank lattice if $n = m$.

Given a matrix $A \in \mathbb{Z}^{n \times m}$ and a vector $\mathbf{u} \in \mathbb{Z}_q^n$, we define two lattices:

$$\Lambda_q^\perp(A) := \{\mathbf{e} \in \mathbb{Z}^m \text{ s.t. } A\mathbf{e} = \mathbf{0} \mod q\},$$
$$\Lambda_q^{\mathbf{u}}(A) := \{\mathbf{e} \in \mathbb{Z}^m \text{ s.t. } A\mathbf{e} = \mathbf{u} \mod q\}.$$

They all are full rank lattices containing $q\mathbb{Z}^m$ and are called q-ary lattices. Note that if $\mathbf{v} \in \Lambda_q^{\mathbf{u}}(A)$, then $\Lambda_q^{\mathbf{u}}(A) = \Lambda_q^\perp(A) + \mathbf{v}$.

For a set of vectors $S = \{\mathbf{s}_1, \cdots, \mathbf{s}_k\}$ in $\mathbb{R}^m$, we denote $\|S\| := \max_i \|\mathbf{s}_i\|$. Also, $\widetilde{S} := \{\widetilde{\mathbf{s}}_1, \cdots, \widetilde{\mathbf{s}}_k\}$ stands for the Gram-Schmidt orthogonalization of the vectors $\mathbf{s}_1, \cdots, \mathbf{s}_k$ in that order. The Gram-Schmidt norm of S is denoted by $\|\widetilde{S}\|$. A basis of a lattice is called *short* if its Gram-Schmidt norm is short.

We recall the shortest independent vectors problem (SIVP), which is the worst case of approximation problem on lattices. Note that the *i-th minimum* of a n-dimensional lattice $\mathcal{L}$ is defined as $\lambda_i(\mathcal{L}) := \min\{r : \dim(\text{span}(\mathcal{L} \cap \mathcal{B}_n(0, r))) \geq i\}$, where $\mathcal{B}_n(0, r) = \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\| \leq r\}$.

Definition 1 (SIVP). *Given a full-rank basis B of an n-dimensional lattice $\mathcal{L}$. SIVP_γ requires to output a set of n linearly independent lattice vectors $S \subset \mathcal{L}(B)$ such that $\|S\| \leq \gamma(n) \cdot \lambda_n(\mathcal{L}(B))$.*

Below we define discrete Gaussian distribution over an integer lattice.

Definition 2 (Gaussian Distribution). *Let $\Lambda \subseteq \mathbb{Z}^m$ be a lattice. For a vector $\mathbf{v} \in \mathbb{R}^m$ and a positive parameter $s \in \mathbb{R}$, define $\rho_{s,\mathbf{v}}(\mathbf{x}) = \exp\left(-\frac{\pi\|\mathbf{x}-\mathbf{v}\|^2}{s^2}\right)$ and $\rho_{s,\mathbf{v}}(\Lambda) = \sum_{\mathbf{x}\in\Lambda} \rho_{s,\mathbf{v}}(\mathbf{x})$. The discrete Gaussian distribution over Λ with center $\mathbf{v}$ and parameter σ is $\forall \mathbf{y} \in \Lambda, \mathcal{D}_{\Lambda,s,\mathbf{v}}(\mathbf{y}) = \frac{\rho_{s,\mathbf{v}}(\mathbf{y})}{\rho_{s,\mathbf{v}}(\Lambda)}$.*

For convenience, ρ_s and $\mathcal{D}_{\Lambda,s}$ denote $\rho_{\mathbf{0},s}$ and $\mathcal{D}_{\Lambda,s,\mathbf{0}}$, respectively. When $s = 1$, we will write ρ instead of ρ_1. Also, $\mathcal{D}^m_{s,\mathbf{v}}$ and $\mathcal{D}^m_s$ stand for $\mathcal{D}_{\mathbb{Z}^m,s,\mathbf{v}}$ and $\mathcal{D}_{\mathbb{Z}^m,s}$, respectively.

Lemma 1 ([21, Lemma 4.5]). *For any $\mathbf{v} \in \mathbb{Z}^m$, if $s = \alpha \cdot \|\mathbf{v}\|$, where $\alpha > 0$, we have $\Pr\left[\mathcal{D}^m_s(\mathbf{x})/\mathcal{D}^m_{s,\mathbf{v}}(\mathbf{x}) \le e^{12/\alpha+1/(2\alpha^2)} : \mathbf{x} \leftarrow \mathcal{D}^m_s\right] \ge 1 - 2^{-100}$.*

Remark 1. In Lemma 1, if $\alpha = 12$, i.e., $s = 12\|\mathbf{v}\|$ then $\mathcal{D}^m_s(\mathbf{x})/\mathcal{D}^m_{s,\mathbf{v}}(\mathbf{x}) \le e^{1+1/288}$ with probability not smaller than $1 - 2^{-100}$.

Trapdoors and Trapdoor Delegation. Alwen and Peikert [4] give an algorithm for sampling a uniform matrix $A \in \mathbb{Z}_q^{n\times m}$ together with a short basis T_A for $\Lambda_q^\perp(A)$. It is an improvement of the algorithm published by Ajtai in [3]. We call T_A an *associated trapdoor* for A or for $\Lambda_q^\perp(A)$.

Theorem 1 ([4]). *Let $q \ge 3$ be odd and $m := \lceil 6n \log q \rceil$. There is a probabilistic polynomial-time (PPT) algorithm $\mathsf{TrapGen}(q, n)$ that outputs a pair $(A \in \mathbb{Z}_q^{n\times m}, T_A \in \mathbb{Z}^{m\times m})$ such that A is statistically close to a uniform matrix in $\mathbb{Z}_q^{n\times m}$ and T_A is a basis for $\Lambda_q^\perp(A)$ satisfying $\|\widetilde{T_A}\| \le O(\sqrt{n \log q})$ and $\|T_A\| \le O(n \log q)$ with all but negligible probability in n.*

Regarding Gaussian distribution, q-ary lattices and trapdoors, some useful results are presented in the following lemma and theorem.

Lemma 2 ([14, Corollary 5.4]). *Let m, n, q be positive integers such that q is prime and $m \ge 2n \log q$. Then for all but $2q^{-n}$ fraction of all matrix $A \in \mathbb{Z}_q^{n\times m}$ and for any $s \ge \omega(\sqrt{\log m})$, the distribution of $\mathbf{u} := A\mathbf{e} \pmod q$ is statistically close to uniform over $\mathbb{Z}_q^n$, where $\mathbf{e} \leftarrow \mathcal{D}_{\mathbb{Z}^m,s}$. Furthermore, the conditional distribution of $\mathbf{e} \leftarrow \mathcal{D}_{\mathbb{Z}^m,s}$, given $A\mathbf{e} = \mathbf{u} \pmod q$, is exactly $\mathcal{D}_{\Lambda^{\mathbf{u}}_q(A),s}$.*

Theorem 2. *Let $q > 2$ and let A, B be a matrix in $\mathbb{Z}_q^{n\times m}$ with $m > n$. Let T_A, T_B be a basis for $\Lambda_q^\perp(A)$ and $\Lambda_q^\perp(B)$, respectively. Then the following statements are true.*

1. *[22, Lemma 4.4] For $s \ge \|\widetilde{T_A}\| \cdot \omega(\sqrt{\log n})$, we have*
$$\Pr[\mathbf{x} \leftarrow \mathcal{D}_{\Lambda^{\mathbf{u}}_q(A),s} \ : \ \|\mathbf{x}\| > s\sqrt{m}] \le \mathsf{negl}(n).$$
2. *[14, Theorem 4.1] There is a PPT algorithm $\mathsf{SampleD}(B, s, \mathbf{v})$ that, given a basis B of an n-dimensional lattice $\Lambda := \mathcal{L}(B)$, a parameters $s \ge \|\widetilde{B}\| \cdot \omega(\sqrt{\log n})$ and a center $\mathbf{v} \in \mathbb{R}^n$, outputs a sample from a distribution statistically close to $\mathcal{D}_{\Lambda,s,\mathbf{v}}$.*

3. *[14, Subsection 5.3.2] There is a PPT algorithm* $\mathsf{SampleISIS}(A, T_A, s, \mathbf{u})$ *that, on input a matrix* A*, its associated trapdoor* T_A*, a Gaussian parameter* $s \geq \|\widetilde{T_A}\| \cdot \omega(\sqrt{\log n})$ *and a given vector* $\mathbf{u}$*, outputs a vector* $\mathbf{e}$ *from* $\mathcal{D}_{\Lambda_q^{\mathbf{u}}(A),s}$*. It performs as follows: first it chooses an arbitrary* $\mathbf{t} \in \mathbb{Z}^m$ *satisfying that* $A\mathbf{t} = \mathbf{u} \pmod q$ *(*$\mathbf{t}$ *exists for all but an at most* q^{-n} *fraction of* A*). It then samples* $\mathbf{w} \leftarrow \mathcal{D}_{\Lambda_q^{\perp}(A),s}$ *using* $\mathsf{SampleD}(T_A, s, -\mathbf{t})$ *and finally outputs* $\mathbf{e} = \mathbf{t} + \mathbf{w}$.
4. *[19, Section 2] There is a PPT algorithm* $\mathsf{SampleKey}(A, T_A, s, K)$ *that takes as input a matrix* $A \in \mathbb{Z}_q^{n \times m}$*, its associated trapdoor* $T_A \in \mathbb{Z}_q^{m \times m}$*, a real number* $s \geq \|\widetilde{T_A}\| \cdot \omega(\sqrt{\log n})$ *and matrix* $K \in \mathbb{Z}_q^{n \times k}$ *to output a random (column) matrix* $S \in \mathbb{Z}^{m \times k}$ *such that the* j*-th column* $S[j] \in \mathsf{Dom} := \{\mathbf{e} \in \mathbb{Z}^m : \|\mathbf{e}\| \leq s\sqrt{m}\}$ *for all* $j \in [k]$ *and that* $A \cdot S = K \pmod q$ *with overwhelming probability. The distribution of* S *is* $\mathcal{D}_{\mathbb{Z}^{m \times k},s}$ *statistically close to the uniform distribution over* Dom^k*. It performs by calling* k *times the algorithm* $\mathsf{SampleISIS}(A, T_A, s, \mathbf{u})$ *in which* $\mathbf{u} = K[j]$ *for* $j \in \{1, \cdots, k\}$.

In order to securely delegate a basis for an extended lattice, one can call the ExtBasis algorithm described below.

Lemma 3 ([2, Theorem 5]). *Let* $A := [A_1 \| A_2 \| A_3]$ *be a concatenation of three matrices* A_1, A_2, A_3*. Suppose that* T_{A_2} *is a basis of* $\Lambda_q^{\perp}(A_2)$*. Then, there is a deterministic polynomial time algorithm* $\mathsf{ExtBasis}(A, T_{A_2})$ *that outputs a basis* T_A *for* $\Lambda_q^{\perp}(A)$ *such that* $\|\widetilde{T_A}\| = \|\widetilde{T}_{A_2}\|$.

Hardness Assumption. Forward-security of our construction is proven assuming hardness of the SIS problem.

Definition 3 (l_2-$\mathsf{SIS}_{q,n,m,\beta}$ problem, [21, Definition 3.1]). *Given a random matrix* $A \leftarrow_{\$} \mathbb{Z}_q^{n \times m}$*, find a vector* $\mathbf{z} \in \mathbb{Z}^m \setminus \{\mathbf{0}\}$ *such that* $A\mathbf{z} = \mathbf{0} \pmod q$ *and* $\|\mathbf{z}\| \leq \beta$.

The hardness of l_2-SIS is stated by the following theorem.

Theorem 3 ([14, Proposition 5.7]). *For any poly-bounded* m*,* $\beta = poly(n)$ *and for any prime* $q \geq \beta \cdot \omega(\sqrt{n \log n})$*, the average case problem* $l_2 - \mathsf{SIS}_{q,n,m,\beta}$ *is as hard as approximating the SIVP problem (among others) in the worst case for a factor* $\gamma = \beta \cdot \tilde{O}(\sqrt{n})$.

Define the $\mathsf{SIS}_{q,n,m,d}$ distribution by the pair $(A, A\mathbf{s})$, where $A \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$ and $\mathbf{s} \xleftarrow{\$} \{-d, \cdots, 0, \cdots, d\}$ are chosen at random. The distribution is characterised by the following lemma.

Lemma 4 (Discussed in [21]). *For* $d \gg q^{m/n}$*, the* $\mathsf{SIS}_{q,n,m,d}$ *distribution is statistically close to uniform over* $\mathbb{Z}_q^{n \times m} \times \mathbb{Z}_q^n$*. Given* $(A, \mathbf{u})$ *from the* $\mathsf{SIS}_{q,n,m,d}$ *distribution, there are many possible solutions* $\mathbf{s}$ *satisfying* $A\mathbf{s} = \mathbf{u}$.

Rejection Sampling. This is an aborting technique that is frequently used in lattice-based cryptography. The technique plays an important role in guaranteeing the blindness as well as it is used in simulation of the forward-security proof for our signature.

Lemma 5 (Rejection Sampling, [21, Theorem 4.6]). *Let $V = \{\mathbf{v} \in \mathbb{Z}^m : \|\mathbf{v}\| \leq \delta\}$ be a subset of $\mathbb{Z}^m$ and $s = \omega(\delta \log \sqrt{m})$ be a real number. Define a probability distribution $h : V \to \mathbb{R}$. Then there exists a universal $M = O(1)$ satisfying that two algorithms $\mathcal{A}$ and $\mathcal{B}$ defined as:*

1. *($\mathcal{A}$): $\mathbf{v} \leftarrow h$, $\mathbf{z} \leftarrow \mathcal{D}^m_{\mathbf{v},s}$, output $(\mathbf{z}, \mathbf{v})$ with probability $\min(\frac{\mathcal{D}^m_s(\mathbf{z},)}{M\mathcal{D}^m_{\mathbf{v},s}(\mathbf{z})}, 1)$, and*
2. *($\mathcal{B}$): $\mathbf{v} \leftarrow h$, $\mathbf{z} \leftarrow \mathcal{D}^m_s$, output $(\mathbf{z}, \mathbf{v})$ with probability $1/M$,*

have a negligible statistical distance $\Delta(\mathcal{A}, \mathcal{B}) := 2^{-\omega(\log m)}/M$. Moreover, the probability that $\mathcal{A}$ outputs something is at least $(1-2^{-\omega(\log m)})/M$. In particular, if $s = \alpha\delta$ for any $\alpha > 0$, then $M = e^{12/\alpha+1/(2\alpha^2)}$, $\Delta(\mathcal{A}, \mathcal{B}) = 2^{-100}/M$ and the probability that $\mathcal{A}$ outputs something is at least $(1 - 2^{-100})/M$.

Commitment Functions. A commitment function com maps a pair of two strings $(\mu,\mathbf{d}) \in \{0,1\}^* \times \{0,1\}^n$ (called *committed string*) to a *commitment string* $C := \mathsf{com}(\mu, \mathbf{d}) \in \{0,1\}^n$. We need com that is both *statistically hiding* and *computationally binding*. For more details, see [24], [19].

3 Framework of Forward-secure Blind Signatures

In this section, we recap the syntax and the security model for forward-secure blind signatures (FSBS). We follow [13], which is in turn adapted from [6].

3.1 Syntax of Forward-secure Blind Signature Schemes

A forward-secure blind signature (or FSBS for short) consists of the four algorithms Setup, KeyUp, Sign, and Verify. They are described as follows:

- $(pp, pk, sk_0) \xleftarrow{\$} \mathsf{Setup}(1^n)$. The algorithm is a PPT one that takes as input a security parameter n and generates common parameters pp, a public key pk and an initial secret key sk_ϵ.
- $sk_{t+1} \xleftarrow{\$} \mathsf{KeyUp}(sk_t, t)$: The key update algorithm is a PPT one, which derives a secret key sk_{t+1} for the time period $t+1$ from a secret key sk_t for a time period t. After execution, the algorithm deletes the secret key sk_t.
- $(\mathcal{V}, \Sigma) \xleftarrow{\$} \mathsf{Sign}(pp, pk, sk_t, t, \mu)$: The signing algorithm involves an interaction between a user, say $\mathcal{U}(pp, pk, t, \mu)$ and a signer, say $\mathcal{S}(pp, pk, sk_t, t)$. At a time period t, the user blinds the message μ using the secret key sk_t and sends it to the signer. The signer replies with a signature of the blinded message. After successful interactions, the user obtains a signature Σ of the original message μ at the time t. The signer gets its own view $\mathcal{V}$. If the interaction fails, the user and signer output $\Sigma := \perp$ and $\mathcal{V} := \perp$, respectively.
- $1/0 := \mathsf{Verify}(pp, pk, t, \mu, \Sigma)$: The verification algorithm is a deterministic one that outputs either 1 if Σ is non-$\perp$ and valid or 0, otherwise. As the input, it accepts a parameter pp, a public key pk, a time period t, a message μ and a signature Σ.

The correctness of FSBS is defined as follows. For any $(pp, pk, sk_0) \leftarrow \mathsf{Setup}(1^n)$ and $(\Sigma, \mathcal{V}) \leftarrow \mathsf{Sign}(pp, pk, sk, t, \mu)$, the verification algorithm fails with a negligible probability or

$$\Pr[\mathsf{Verify}(pp, pk, t, \mu, \Sigma) = 1] = 1 - \mathsf{negl}(n).$$

3.2 Security of Forward-secure Blind Signatures

Two properties required for forward-secure blind signatures are *blindness* and *forward security*. Blindness ensures that it is impossible for the signer to learn any information about messages being signed.

Definition 4 (Blindness). FSBS *is blind if for any efficient algorithm* $\mathcal{S}^*$*, the advantage of* $\mathcal{S}^*$ *in the blindness game* $\mathsf{Blind}^{\mathcal{S}^*}_{\mathsf{FSBS}}$ *is negligible. That is*

$$\mathsf{Adv}^{\mathsf{Blind}}_{\mathsf{FSBS}}(\mathcal{S}^*) := \Pr[\mathsf{Blind}^{\mathcal{S}^*}_{\mathsf{FSBS}} \Rightarrow 1] - 1/2 \leq \mathsf{negl}(n).$$

FSBS *is called perfectly blind if* $\Pr[\mathsf{Blind}^{\mathcal{S}^*}_{\mathsf{FSBS}} \Rightarrow 1]$ *is exactly* $1/2$.

The blindness game $\mathsf{Blind}^{\mathcal{S}^*}_{\mathsf{FSBS}}$ consists of three phases defined below.

1. **Initialization.** The adversary $\mathcal{S}^*$ chooses a security parameter n, then obtains common parameters pp, a public key pk and an initial secret key sk_0 using $\mathsf{Setup}(1^n)$.
2. **Challenge.** $\mathcal{S}^*$ selects and gives the challenger $\mathcal{C}$ two messages μ_0 and μ_1. The challenger $\mathcal{C}$ flips a coin $b \in \{0, 1\}$ and initiates two signing interactions with $\mathcal{S}^*$ on input μ_b and μ_{1-b} (not necessarily in two different time periods). The adversary $\mathcal{S}^*$ acts as the signer in these two interactions and finally attains two corresponding view/signature pairs $(\mathcal{V}_b, \Sigma_b)$ and $(\mathcal{V}_{1-b}, \Sigma_{1-b})$.
3. **Output.** The adversary $\mathcal{S}^*$ outputs $b' \in \{0, 1\}$. It wins if $b' = b$.

Following [13], we define forward-security as the *forward-secure unforgeability*. In the $\mathsf{FSUF}^{\mathcal{U}^*}_{\mathsf{FSBS}}$ game, the forger $\mathcal{U}^*$ is a malicious user (adversary).

Definition 5 (Forward-secure Unforgeability). FSBS *is forward-secure unforgeable* (FSUF) *if for any efficient algorithm* $\mathcal{U}^*$*, the advantage of* $\mathcal{U}^*$ *in the forward-secure unforgeability game* $\mathsf{FSUF}^{\mathcal{U}^*}_{\mathsf{FSBS}}$ *is negligible. That is,*

$$\mathsf{Adv}^{\mathsf{FSUF}}_{\mathsf{FSBS}}(\mathcal{U}^*) := \Pr[\mathsf{FSUF}^{\mathcal{U}^*}_{\mathsf{FSBS}} \Rightarrow 1] \leq \mathsf{negl}(n).$$

In our work, the forward-secure unforgeability game $\mathsf{FSUF}^{\mathcal{U}^*}_{\mathsf{FSBS}}$ in defined in the random oracle model. (We use hashing as an instantiation of random oracle.) We assume that, whenever the adversary wants to make a signing query, it always makes a random oracle query in advance.

1. **Setup.** The forger $\mathcal{U}^*$ gives a security parameter n to the challenger $\mathcal{C}$. The challenger $\mathcal{C}$ generates system parameters pp and outputs the key pair (pk, sk_0) by calling $\mathsf{Setup}(1^n)$. Then $\mathcal{C}$ sends pp and pk to the forger $\mathcal{U}^*$. The key sk_0 is kept secret.

2. **Queries.** At a time period t, the forger $\mathcal{U}^*$ can make a polynomially many random oracle queries as well as a polynomially many signing queries in an adaptive manner. In order to move to the next time period, the forger makes a key update query to get the secret key sk_{t+1} for the time period t+1. Note that, once the forger makes a key update query, i.e., it obtains the secret key sk_{t+1}, it cannot issue random oracle and signing queries for past time intervals. Finally, the forger is allowed to make a single break-in query at a time period $\bar{t} \leq T-1$, when it wants to stop the query phase. The time interval $\bar{t}$ is called the *break-in time*. Once the forger makes the break-in query, it is not able to make further random oracle (or hash) and signing queries. Details of the challenger actions in response to the forger queries are given below.
   - For key update query $KQ(t)$: if $t < T-1$, then the challenger updates the secret key sk_t to sk_{t+1} and updates t to $t+1$. If $t = T-1$ then sk_T is given as an empty string.
   - For each hash queries $HQ(t, \mu)$: the challenger has to reply with a random value.
   - For each signing query $SQ(t, \mu)$: the challenger must send a valid signature back to $\mathcal{U}^*$.
   - For the break-in query $BQ(\bar{t})$ (note that the query is allowed once only): the challenger must send the secret key $sk_{\bar{t}}$ to the adversary and move the game to the output phase.
3. **Output.** $\mathcal{U}^*$ outputs at least one forgery (μ^*, t^*, Σ^*) at time period t^*. He wins the game if $t^* < \bar{t}$, $SQ(t^*, \mu^*)$ has been never queried, and (μ^*, t^*, Σ^*) is valid.

4 Our Construction

4.1 Binary Tree Hierarchy for Time Periods

Our design applies a binary-tree data structure. In the context of encryption, binary trees have been introduced by [8]. For the lattice setting, they have been adapted by Cash et al. in [9]. The tree structure is useful for constructing forward-secure public key encryption schemes [8], HIBE [9] and recently for forward-secure group signature [20]. We need time periods $t \in \{0, \cdots, 2^\ell - 1\}$ to be assigned to leaves of a binary tree of the depth ℓ. The tree leaves are arranged in increasing order from left to right – see Figure 1. For a time period t, there is a unique path $t = (t_1, \cdots, t_\ell)$ from the root ϵ to the leaf, where for each level $i \in [\ell]$, $t_i = 0$ if this is the left branch or $t_i = 1$ if this is the right branch. Consequently, the i-th level node $w^{(i)}$ in the binary tree can be described by a unique binary bit string $w^{(i)} = (w_1, \cdots, w_i)$ that follows the path from the root to the node. This means that for the node $w^{(i)} = (w_1, \cdots, w_i)$, we can create a corresponding matrix $W_t = [A_0 \| A_1^{(w_1)} \| \cdots \| A_i^{(w_i)}]$ (resp., $F_t = [A_0 \| A_1^{(t_1)} \| \cdots \| A_\ell^{(t_\ell)}]$), where A_0 and its ascociated trapdoor T_{A_0} are generated by $\mathsf{TrapGen}$ and $A_i^{(b)}$ are random matrices for all $i \in [\ell], b \in \{0, 1\}$.

Updating secret keys from time period t to $t+1$ is done by the trapdoor delegation mechanism using $\mathsf{ExtBasis}$. Each node $w^{(i)} = (w_1, \cdots, w_i)$ is associated with a secret key $T_{w^{(i)}}$, which can be computed from the initial secret key $sk_0 = T_{A_0}$ by evaluating

$$T_{w^{(i)}} \leftarrow \mathsf{ExtBasis}(A_{w^{(i)}}, T_{A_0}), \text{ where } A_{w^{(i)}} = \left[A_0 \| A_1^{(w_1)} \| A_2^{(w_2)} \| \cdots \| A_i^{(w_i)}\right].$$

$T_{w^{(i)}}$ is easily computed if a secret key $T_{w^{(k)}}$ for an ancestor $w^{(k)}$ of $w^{(i)}$ is known. Assume that the binary representation of $w^{(i)}$ is $w^{(i)} = (w_1, \cdots, w_k, w_{k+1}, \cdots, w_i)$, where $k < i$. Then

$$T_{w^{(i)}} \leftarrow \mathsf{ExtBasis}(A_{w^{(i)}}, T_{w^{(k)}}), \text{ where } A_{w^{(i)}} = \left[A_0 \| A_1^{(w_1)} \| \cdots \| A_2^{(w_k)} \| \cdots \| A_i^{(w_i)}\right].$$

Similarly, a secret key for a time period (i.e., a leaf) can be computed if we have any its ancestor's secret key.

4.2 Description of the Proposed Signature

Our lattice-based forward-secure blind signature (FSBS) consists of a setup algorithm Setup, a key update algorithm KeyUp, an interactive signing algorithm Sign and a verification algorithm Verify. They all are described below. Note that, we also use a commitment function com.

$\underline{\mathsf{Setup}}(1^n, 1^\ell)$: For a security parameter n and a binary tree depth ℓ, the algorithm runs through the following steps.

- Choose $q = poly(n)$ prime, $m = O(n \log q)$, k, κ, ℓ, $\tau = 2^\ell$, σ, σ_1, σ_2, σ_3 (see Section 5.4 for details).
- Let $\mathcal{M} = \{0,1\}^*$ be the message space of the scheme.
- Choose randomly a matrix $K \xleftarrow{\$} \mathbb{Z}_q^{n \times k}$. Similarly, select matrices $A_1^{(0)}, A_1^{(1)}$, $A_2^{(0)}, A_2^{(1)}, \cdots, A_\ell^{(0)}, A_\ell^{(1)}$ from $\mathbb{Z}_q^{n \times m}$ at random.
- Run $\mathsf{TrapGen}(q, n)$ to obtain a pair (A_0, T_{A_0}), where $A_0 \in \mathbb{Z}_q^{n \times m}$ and $T_{A_0} \in \mathbb{Z}^{m \times m}$ are a matrix and its associated trapdoor.
- Let $H : \{0,1\}^* \to \mathcal{R}_H$ be a collision-resistant and one-way hash function, where $\mathcal{R}_H := \{\mathsf{e}' \in \{-1, 0, 1\}^k : \|\mathsf{e}'\| \leq \kappa\}$.
- Let $\mathsf{com} : \{0,1\}^* \times \{0,1\}^n \to \{0,1\}^n$ be a computationally binding and statistically hiding commitment function.
- Output $pp \leftarrow \{n, q, m, \ell, \tau, k, \kappa, \sigma, \sigma_1, \sigma_2, \sigma_3, \mathcal{M}, H, \mathsf{com}\}$, $pk \leftarrow \{A_0, A_1^{(0)}, A_1^{(1)}, \cdots, A_\ell^{(0)}, A_\ell^{(1)}, K\}$, and $sk_\epsilon \leftarrow T_{A_0}$ as common parameters, public key and the initial secret key, respectively.

$\underline{\mathsf{KeyUp}}(pp, pk, sk_t, t)$: We need a key evolution mechanism (KVM) that "forgets" all secret keys of internal nodes that can produce past keys. Additionally, we expect that KVM stores the smallest number of keys necessary for signature to work properly. The key evolution mechanism KVM works as follows.

- For any leaf t, define the minimal cover $\mathsf{Node}(t)$ to be the smallest subset of nodes that contains an ancestor of all leaves in $\{t, \cdots, T-1\}$ but does not contain any ancestor of any leaf in $\{0, \cdots, t-1\}$. For example, in Figure 1, $\mathsf{Node}(0) = \{\epsilon\}$, $\mathsf{Node}(1) = \{001, 01, 1\}$, $\mathsf{Node}(2) = \{01, 1\}$, $\mathsf{Node}(3) = \{011, 1\}$ (i.e., two black circles in the tree), $\mathsf{Node}(4) = \{1\}$, $\mathsf{Node}(5) = \{101, 11\}$, $\mathsf{Node}(6) = \{11\}$, $\mathsf{Node}(7) = \{111\}$.
- The secret key sk_t at time period t contains secret keys corresponding to all nodes (including leaves) in $\mathsf{Node}(t)$. For example, for the tree from Figure 1, we have $sk_0 = sk_\epsilon = \{T_{A_0}\}$, $sk_1 = \{T_{001}, T_{01}, T_1\}$, where T_{001}, T_{01}, and T_1 are associated trapdoors for $F_{001} = [A_0 \| A_1^{(0)} \| A_2^{(0)} \| A_3^{(1)}]$, $F_{01} = [A_0 \| A_1^{(0)} \| A_2^{(1)}]$ and $F_1 = [A_0 \| A_1^{(1)}]$, respectively.
- To update sk_t to sk_{t+1}, the signer determines the minimal cover $\mathsf{Node}(t+1)$, then derives keys for all nodes in $\mathsf{Node}(t+1) \setminus \mathsf{Node}(t)$ using the keys in sk_t as described in Section 4.1. Finally the signer deletes all keys in $\mathsf{Node}(t) \setminus \mathsf{Node}(t+1)$. For example, $sk_2 = \{T_{01}, T_1\}$ (mentioned above), since $\mathsf{Node}(2) \setminus \mathsf{Node}(1) = \{01, 1\}$ and $\mathsf{Node}(1) \setminus \mathsf{Node}(2) = \{001\}$.

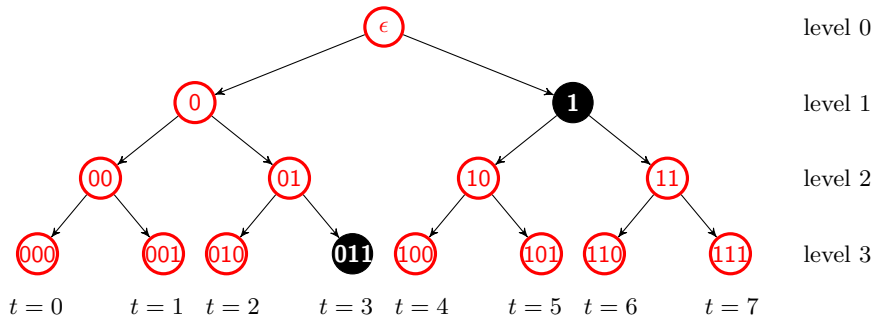

Fig. 1. Binary tree of depth $\ell = 3$, i.e., for $\tau = 8$ time periods. The root is denoted by ϵ. For convenience, we name nodes by their binary representations

$\underline{\mathsf{Sign}}(pp, pk, sk_t, t, \mu)$: The signer interacts with the user in order to produce a signature for a message $\mu \in \mathcal{M}$ at time period t. The interaction consists of five phases. Phases 1,3 and 5 are done by the signer. Phases 2 and 4 – by the user.

- *Phase 1*: The signer constructs the matrix $F_t = \left[A_0 \| A_1^{(t_1)} \| \cdots \| A_\ell^{(t_\ell)}\right] \in \mathbb{Z}_q^{n \times (\ell+1)m}$ for the time $t = (t_1, \cdots, t_\ell)$. Next it computes an ephemeral secret key S_t using $\mathsf{SampleKey}$ described in Theorem 2, where $F_t \cdot S_t = K$. Note that S_t can be computed at Phase 3 as well. The signer samples $\mathbf{r} \in \mathbb{Z}^{(\ell+1)m}$ according to the distribution $\mathcal{D}_{\sigma_2}^{(\ell+1)m}$. It finally computes and sends $\mathbf{x} = F_t \mathbf{r} \in \mathbb{Z}_q^n$ to the user.
- *Phase 2*: Upon receiving $\mathbf{x}$, the user samples blind factors $\mathbf{a} \leftarrow \mathcal{D}_{\sigma_3}^{(\ell+1)m}$ and $\mathbf{b} \leftarrow \mathcal{D}_{\sigma_1}^{k}$, $\mathbf{d}' \xleftarrow{\$} \{0,1\}^n$. It computes $\mathbf{u} = \mathbf{x} + F_t \mathbf{a} + K\mathbf{b}$ and hashes

it with $\mathbf{c} := \mathsf{com}(\mu, \mathbf{d}') \in \{0,1\}^n$ using the hash function H to obtain a *real challenge* $\mathbf{e}'$. The rejection sampling technique is called to get the *blinded challenge* $\mathbf{e}$, which is sent back to the user.

- *Phase 3*: The ephemeral secret key S_t and $\mathbf{r}$ are used to compute $\mathbf{z} = \mathbf{r} + S_t\mathbf{e}$. In order to guarantee that no information of S_t is leaked, the rejection sampling is applied, which implies that the distribution of $\mathbf{z}$ and $\mathbf{r}$ are the same. Finally, the *blinded signature* $\mathbf{z}$ is delivered to the user.
- *Phase 4*: The user computes the unblinded signature $\mathbf{z}' = \mathbf{z} + \mathbf{a}$. Again, the rejection sampling is called to make sure that $\mathbf{z}'$ and $\mathbf{z}$ are independent of each other and $\mathbf{z}'$ is bounded in some desired domain. The user returns $(t, \mu, \Sigma = (\mathbf{d}', \mathbf{e}', \mathbf{z}'))$ as the *final signature* if $\|\mathbf{z}'\| \leq \sigma_3\sqrt{(1+\ell)m}$ holds. Otherwise, he outputs "$\perp$". The user is required to confirm validity of the final signature by sending result to the signer: $\mathsf{result} := \mathsf{accept}$ means the final signature is good, while $\mathsf{result} := (\mathbf{a}, \mathbf{b}, \mathbf{e}', \mathbf{c})$) requires the user to restart the signing protocol.
- *Phase 5*: Having obtained result, the signer checks whether or not $\mathsf{result} \neq \mathsf{accept}$. If not, it returns the *view* $\mathcal{V} = (t, \mathbf{r}, \mathbf{e}, \mathbf{z})$. Otherwise it makes some check-up operations before restarting the signing algorithm. The check-up allows the signer to detect an adversary who controls the user and tries to forge a signature.

Note that the rejection sampling in Phase 2 is not able restart the signing algorithm as it is used locally. In contrast, the rejection sampling in Phase 3 and Phase 4 can make the signing algorithm restart. The reader is referred to Section 5 for more details. Figure 2 illustrates the signing algorithm.

$\underline{\mathsf{Verify}(t, pk, \mu, \Sigma)}$: The algorithm accepts a signature Σ on the message μ for the time period $t = (t_1, \cdots, t_\ell)$ and public key pk as its input and performs the following steps:

(i) parse $\Sigma = (\mathbf{d}', \mathbf{e}', \mathbf{z}')$;

(ii) form $F_t := \left[A_0 \| A_1^{(t_1)} \| \cdots \| A_\ell^{(t_\ell)}\right] \in \mathbb{Z}^{n \times (1+\ell)m}$;

(iii) compute $\widehat{\mathbf{e}} := H(F_t\mathbf{z}' - K\mathbf{e}' \bmod q, \mathsf{com}(\mu, \mathbf{d}'))$;

(iv) if $\|\mathbf{z}'\| \leq \sigma_3\sqrt{(1+\ell)m}$ and $\widehat{\mathbf{e}} = \mathbf{e}'$, then output 1, otherwise return 0.

5 Correctness, Security and Parameters for FSBS

5.1 Correctness

Theorem 4 (Correctness). *The correctness of* FSBS *scheme holds after at most* e^2 *restarts with probability not smaller than* $1 - 2^{-100}$.

Proof. Given $(t, \mu, \Sigma = (\mathbf{d}', \mathbf{e}', \mathbf{z}'))$ produced by $\mathsf{Sign}(pp, pk, sk_t, \mu)$ – see Figure 2. It is east to show that $H(F_t\mathbf{z}' - K\mathbf{e}' \pmod q, \mathsf{com}(\mu, \mathbf{d}')) = \mathbf{e}'$. Note that $\|\mathbf{z}'| \leq \sigma_3\sqrt{(1+\ell)m}$ with overwhelming probability by Statement 1 of Theorem 2. Remark 1 implies that if $s = 12\|\mathbf{c}\|$, then $\frac{\mathcal{D}_s^m(\mathbf{x})}{M \cdot \mathcal{D}_{s,\mathbf{c}}^m(\mathbf{x})} \leq \frac{e^{1+1/288}}{M}$ with probability at least $1 - 2^{-100}$. The rejection sampling requires that $\mathcal{D}_s^m(\mathbf{x})/(M \cdot \mathcal{D}_{s,\mathbf{c}}^m(\mathbf{x})) \leq 1$,

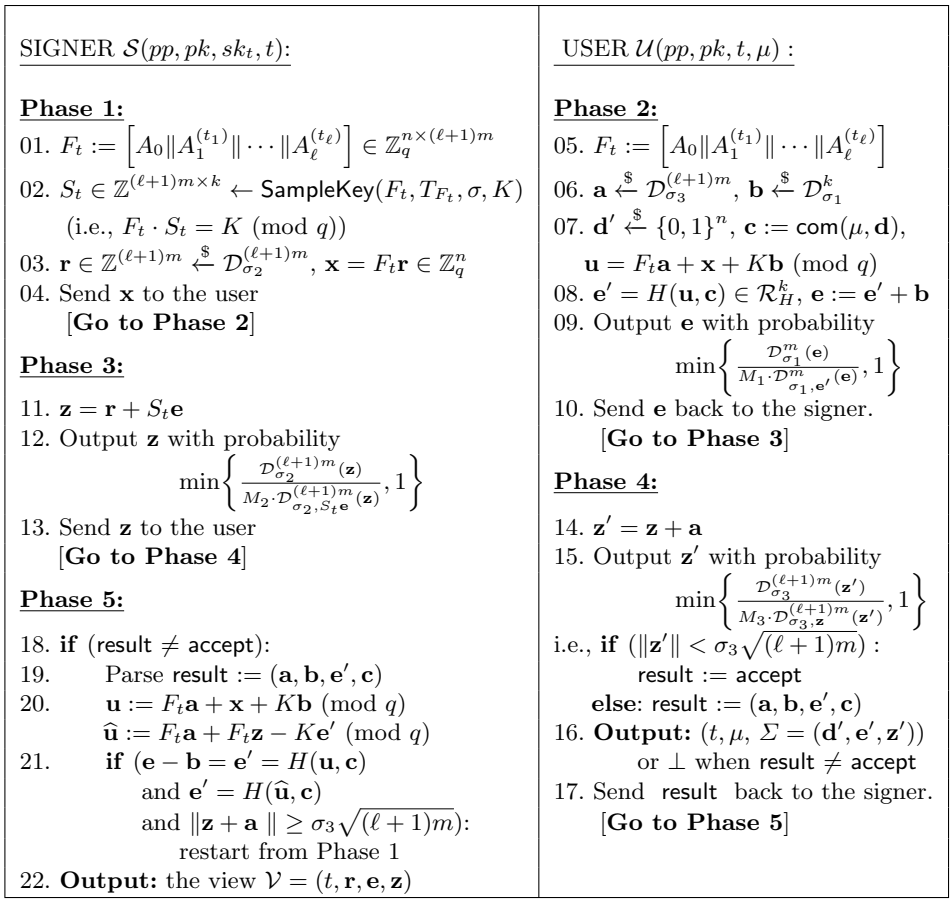

Fig. 2. The signing algorithm $\mathsf{Sign}(pp, pk, sk_t, t, \mu)$

meaning that $M \geq e^{1+1/288}$. It is easy to see that $M \approx e^{1+1/288}$ is the best choice. Applying this observation to the rejection samplings in Phases 3 and 4, we see that a valid signature can be successfully produced after at most $M_2 \cdot M_3 \approx e^2$ repetitions. □

5.2 Blindness

Theorem 5 (Blindness). *Let* com *be a statistically hiding commitment and H be an one-way and collision-resistant hash function. Then, the proposed forward-secure blind signature* FSBS *is blind.*

Proof. In the blindness game $\mathsf{Blind}_{\mathsf{FSBS}}^{\mathcal{S}^*}$, the adversarial signer $\mathcal{S}^*$ gives the challenger $\mathcal{C}$ two messages μ_0 and μ_1. The challenger $\mathcal{C}$ chooses uniformly at random

a bit $b \in \{0,1\}$ and interacts with $\mathcal{S}^*$ in order to sign both messages μ_b and μ_{1-b}. $\mathcal{C}$ acts as two users $\mathcal{U}_b := \mathcal{U}(pp, pk, t, \mu_b)$ and $\mathcal{U}_{1-b} := \mathcal{U}(pp, pk, t, \mu_{1-b})$. Finally, $\mathcal{S}^*$ gets two pairs $(\mathcal{V}_b, \Sigma_b)$ and $(\mathcal{V}_{1-b}, \Sigma_{1-b})$ that correspond to the users $\mathcal{U}_b$ and $\mathcal{U}_{1-b}$, respectively. We argue that the knowledge of $(\mathcal{V}_b, \Sigma_b)$ and $(\mathcal{V}_{1-b}, \Sigma_{1-b})$ is independent of the signed messages. In other words, $\mathcal{S}^*$ cannot distinguish, which user it is communicating with. In other words, it cannot guess b with non-negligible probability.

Indeed, for $\mathcal{V}_b = (t, \mathbf{r}_b, \mathbf{e}_b, \mathbf{z}_b)$ and $\mathcal{V}_{1-b} = (t, \mathbf{r}_{1-b}, \mathbf{e}_{1-b}, \mathbf{z}_{1-b})$, we need to consider the pair $(\mathbf{e}_b, \mathbf{e}_{1-b})$ only, since $\mathbf{z}_b$ and $\mathbf{z}_{1-b}$ are produced by $\mathcal{S}^*$ itself. In Phase 2, the rejection sampling makes sure that the distribution of both $\mathbf{e}_b$ and $\mathbf{e}_{1-b}$ are the same, which is $\mathcal{D}^k_{\sigma_1}$. This means that $\mathbf{e}_b$ and $\mathbf{e}_{1-b}$ are independent of the signed messages. Consider $\Sigma_b = (\mathbf{d}'_b, \mathbf{e}'_b, \mathbf{z}'_b)$ and $\Sigma_{1-b} = (\mathbf{d}'_{1-b}, \mathbf{e}'_{1-b}, \mathbf{z}'_{1-b})$. As Phase 4 uses the rejection sampling, both $\mathbf{z}'_b$ and $\mathbf{z}'_{1-b}$ have the same distribution, which is $\mathcal{D}^{(1+\ell)m}_{\sigma_3}$. It means that $\mathcal{S}^*$ does not learn anything about the signed messages from the knowledge of $(\mathbf{d}'_b, \mathbf{d}'_{1-b})$ and $(\mathbf{e}'_b, \mathbf{e}'_{1-b})$. This is true because the former pair are randomly chosen and the latter pair are hash values of the one-way and collision-resistant function H.

Finally, it is easy to see that restarts, which may happen in Phase 5, do not increase advantage of $\mathcal{S}^*$ in the blindness game. In fact, a restart occurs if the user has sent $\mathsf{result} := (\mathbf{a}, \mathbf{b}, \mathbf{e}', \mathbf{c})$ to $\mathcal{S}^*$. The values $\mathbf{d}'$, $\mathbf{a}$ and $\mathbf{b}$ are freshly sampled by the user. Additionally, as com is a statistically hiding commitment, knowing $\mathbf{c}$, $\mathcal{S}^*$ cannot tell apart μ_b from μ_{1-b}. □

5.3 Forward-secure Unforgeability

We recall the following lemma, which we use to support our witness indistinguishability argument.

Lemma 6 (Adapted from [21, Lemma 5.2]). *Given a matrix* $\mathbf{F} \in \mathbb{Z}_q^{n \times (\ell+1)m}$, *where* $(\ell+1)m > 64 + n \log q / \log(2d+1)$ *and* $\mathbf{s} \xleftarrow{\$} \{-d, \cdots, 0, \cdots, d\}^{(\ell+1)m}$. *Then there exists another* $\mathbf{s}' \xleftarrow{\$} \{-d, \cdots, 0, \cdots, d\}^{(\ell+1)m}$ *such that* $\mathbf{Fs} = \mathbf{Fs}' (mod\ q)$ *with probability at least* $1 - 2^{-100}$.

Note that Lemma 4 also gives the same conclusion as Lemma 6 but with the not so clear condition $d \gg q^{(\ell+1)m/n}$.

Theorem 6 (Forward-secure Unforgeability). *Suppose that the commitment function* com *used in* FSBS *is computationally binding and that there exists a forger* $\mathcal{A}$, *who can break the forward-secure unforgeablity of* FSBS. *Then, one can construct a polynomial-time algorithm* $\mathcal{B}$ *that solves an* $l_2\text{-}\mathsf{SIS}_{q,n,(1+2\ell)m,\beta}$ *problem with* $\beta = \max\{(2\sigma_3 + 2\sigma\sqrt{\kappa})\sqrt{(1+\ell)m}, (2\sigma_3 + \sigma_2)\sqrt{(1+\ell)m}\}$.

Proof. The reduction is as follows:

Phase 0 (Instance). Assume that $\mathcal{B}$ wants to solve an instance of the $\mathsf{SIS}_{q,n,(1+2\ell)m,\beta}$ problem

$$F \cdot \mathbf{v} = 0 \mod q, \|\mathbf{v}\| \leq \beta, F \in \mathbb{Z}_q^{n \times (1+2\ell)m}, \tag{1}$$

in which F is parsed as $F = \left[A_0 \| U_1^{(0)} \| U_1^{(1)} \| \cdots \| U_\ell^{(0)} \| U_\ell^{(1)}\right]$ with $A_0^{(k)}, U_i^{(b)} \in \mathbb{Z}_q^{n \times m}$ for $\beta = \max\{(2\sigma_3 + 2\sigma\sqrt{\kappa})\sqrt{(1+\ell)m}, (2\sigma_3 + \sigma_2)\sqrt{(1+\ell)m}\}$ and $b \in \{0,1\}$.

Phase 1 (Guessing the target). $\mathcal{B}$ guesses the target time period t^* that $\mathcal{A}$ wants to attack by choosing randomly $t^* = (t_1^*, \cdots, t_\ell^*) \xleftarrow{\$} \{0, \cdots, \tau - 1\}$. The success probability of guessing t^* is $1/\tau$.

Phase 2 (Initialize). $\mathcal{B}$ sets common parameters pp as in the Setup algorithm. However, $\mathcal{B}$ sets the public key pk according to the following steps.

- For $i \in [\ell]$, $\mathcal{B}$ sets $A_i^{(t_i^*)} = U_i^{(t_i^*)}$. For each bit $b \in \{0,1\}$ such that $b \neq t_i^*$, $\mathcal{B}$ invokes $\mathsf{TrapGen}$ to generate $A_i^{(b)}$ together with a short basis $T_{A_i^{(b)}}$ of $A_q^\perp(A_i^{(b)})$.
- $\mathcal{B}$ samples $S^* \leftarrow \mathcal{D}_\sigma^{(1+\ell)m \times k}$ and sets $K := F_{t^*} \cdot S^*$, where $F_{t^*} = \left[A_0 \| A_1^{(t_1^*)} \| \cdots \| A_\ell^{(t_\ell^*)}\right] \in \mathbb{Z}_q^{n \times (1+\ell)m}$. Let $d := \sigma\sqrt{(1+\ell)m}$. Then σ should be chosen sufficiently large to satisfy Lemma 2 (i.e., $\sigma \geq \omega(\sqrt{\log((1+\ell)m)})$), Lemma 4 (i.e., $d \gg q^{(1+\ell)m/n}$) and Lemma 6 (i.e., $(1+\ell)m > 64 + n \log q / \log(2d+1)$). Statement 1 of Theorem 2 gurantees that $\|S^*\| \leq d$ with overwhelming probability. According Lemma 2, K is statistically close to uniform.
- Finally, $\mathcal{B}$ sends pp, and $pk \leftarrow \{A_0, A_1^{(0)}, A_1^{(1)}, \cdots, A_\ell^{(0)}, A_\ell^{(1)}, K\}$ to $\mathcal{A}$ as the common parameters and the public key, while keeping $T_{A_i^{(b)}}$'s and S^* secret.

$\mathcal{B}$ creates and maintains a list $\mathcal{L}_H$ consisting of random oracle queries $(\mathbf{u}, \mathbf{c}) \xleftarrow{\$} \mathbb{Z}_q^n \times \{0,1\}^n$ and their corresponding hash value $\mathbf{e}' \in \mathcal{R}_H$. In other words, $\mathcal{L}_H = \{(\mathbf{u}, \mathbf{c}, \mathbf{e}') \in \mathbb{Z}_q^n \times \{0,1\}^n \times R_H : \mathbf{e}' = H(\mathbf{u}, \mathbf{c})\}$. In addition, $\mathcal{B}$ also prepares the set of replies for q_H hash queries $\mathcal{R} := \{\mathbf{r}_1, \cdots, \mathbf{r}_{q_H}\}$, where each $\mathbf{r}_i \xleftarrow{\$} \mathcal{R}_H$. It then chooses a random tape ρ and runs $\mathcal{A}$ on (pp, pk, ρ) in a black-box manner.

Phase 3 (Queries). $\mathcal{B}$ plays the role of signer and interacts with $\mathcal{A}$. $\mathcal{B}$ responds to $\mathcal{A}$ queries as follows:

- *Key update queries* $KQ(t), t = (t_1, \cdots, t_\ell)$*:* If $t \leq t^*$, $\mathcal{B}$ aborts the query. Otherwise, let $k \leq \ell$ be the minimum index such that $t_k \neq t_k^*$. Then, the adversary $\mathcal{B}$ first uses the trapdoor $T_{A_k^{(t_k)}}$ to compute the key T_{t_k} for the node t_k

$$T_{t_k} \leftarrow \mathsf{ExtBasis}(E \| A_k^{(t_k)}, T_{A_k^{(t_k)}}), \text{ where } E = \left[A_0 \| A_1^{(t_1)} \| \cdots \| A_{k-1}^{(t_{k-1})}\right],$$

  from which $\mathcal{B}$ computes all keys in sk_t as in the real key update algorithm.
- *Hash queries* $HQ(\mathbf{u}, \mathbf{c})$: Having received a hash query $(\mathbf{u}, \mathbf{c})$, $\mathcal{B}$ checks if the list $\mathcal{L}_H$ contains the query. If $\mathcal{B}$ finds out that $(\mathbf{u}, \mathbf{c})$ is in $\mathcal{L}_H$ already, then $\mathcal{B}$ sends the corresponding hash value $\mathbf{e}'$ to the forger $\mathcal{A}$. Otherwise, $\mathcal{B}$ chooses the first unused $\mathbf{r}_i, i \in [q_H]$ from $\mathcal{R}$, takes $\mathbf{e}' := \mathbf{r}_i$ and stores the query-hash value pair $((\mathbf{u}, \mathbf{c}), \mathbf{e}')$ in $\mathcal{L}_H$. Finally, $\mathcal{B}$ sends $\mathbf{e}'$ to the forger $\mathcal{A}$ as the answer.

- *Signing queries* $SQ(t, \mu)$: $\mathcal{B}$ constructs $F_t := \left[A_0^{(k_0)} \| A_1^{(t_1)} \| \cdots \| A_\ell^{(t_\ell)}\right]$ and checks if $t \neq t^*$ or not. If $t \neq t^*$, $\mathcal{B}$ computes $T_{F_t} \leftarrow \mathsf{ExtBasis}(F_t, T_{A_k^{(t_k)}})$, and $S_t \leftarrow \mathsf{SampleKey}(F_t, T_{F_t}, \sigma, K)$, where $k \leq \ell$ is the minimum index such that $t_k \neq t_k^*$. Note that $F_t \cdot S_t = K$. Otherwise, if $t = t^*$, $\mathcal{B}$ simply assigns $S_{t^*} \leftarrow S^*$ since $F_{t^*} \cdot S^* = K$.
- *Break-in queries* $BQ(t)$: Once the adversary $\mathcal{A}$ makes a query $BQ(t)$, if $t \leq t^*$, then $\mathcal{B}$ aborts. Otherwise, i.e., $t > t^*$, $\mathcal{B}$ decides that the break-in time is $\bar{t} \leftarrow t$. $\mathcal{B}$ answers to $\mathcal{A}$ by sending the secret key $\mathsf{sk}_{\bar{t}}$ in the same way as replying to the key update queries since $\bar{t} = t > t^*$.

Phase 4 (Forge). Eventually, $\mathcal{A}$ outputs a forgery $(t_1', \mu_1^*, \Sigma_1^*)$. $\mathcal{B}$ checks if $t_1' = t^*$ or not. If not, then $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ accepts the forgery. For the forgery $(t^*, \mu_1^*, \Sigma_1^*)$, we have: (i) $\Sigma_1^* = (\mathbf{d}_1', \mathbf{e}_1', \mathbf{z}_1')$; (ii) $\mathbf{e}_1' := H(F_{t^*}\mathbf{z}'_1 - K\mathbf{e}_1' \bmod q, \mathsf{com}(\mu_1^*, \mathbf{d}_1'))$, where $F_{t^*} := \left[A_0 \| A_1^{(t_1^*)} \| \cdots \| A_\ell^{(t_\ell^*)}\right] \in \mathbb{Z}^{n \times (1+\ell)m}$; and (iii) $\|\mathbf{z}'_1\| \leq \sigma_3 \sqrt{(1+\ell)m}$.

Analysis. We argue that the simulation of $\mathcal{B}$ is statistically perfect. In other words, the forger $\mathcal{A}$ is not able to distinguish the simulator $\mathcal{B}$ from the real challenger in the FSEU game. Indeed, the simulation proceeds as the real game except the following exceptions.

(i) Some matrices $A_i^{(b)}$ are not really random but is generated by $\mathsf{TrapGen}$. However, Theorem 1 ensures that the distribution of $A_i^{(b)}$ generated by $\mathsf{TrapGen}$ is close to uniform.

(ii) The matrix K is not randomly chosen. It is obtained by sampling S^* from $\mathcal{D}_\sigma^{(1+\ell)m \times m}$ and then assigning $K := F \cdot S^*$. Lemma 2 asserts that selection of K is close to uniform. Note that the sufficiently large choice of σ does not affect (iii).

(iii) The matrix S_{t^*} is equal to S^*, which is not computed using $\mathsf{SampleKey}$. The forger $\mathcal{A}$ does not know S_t so consequently does not know S^*. As $\mathbf{z}$ is generated (in Step 12) using the rejection sampling, we always guarantee that $\mathbf{z} \leftarrow \mathcal{D}_{\sigma_2}^{(\ell+1)m}$ and $\mathbf{z}$ is independent of S_t and S^*. Thus the view of $\mathcal{A}$ is independent of S^*.

Now, we show how to obtain the solution to the l_2-SIS problem given by Equation (1). Let $i \in [q_H]$ be the target forking index, for which $\mathbf{e}_1' = \mathbf{r}_i$. $\mathcal{B}$ follows the rewinding strategy by keeping $\{\mathbf{r}_1, \cdots, \mathbf{r}_{i-1}\}$ and sampling new fresh answers $\{\mathbf{r}_i', \cdots, \mathbf{r}_{q_H}'\} \xleftarrow{\$} \mathcal{R}_H$. Now, $\mathcal{B}$ uses $\mathcal{R}' := \{\mathbf{r}_1, \cdots, \mathbf{r}_{i-1}, \mathbf{r}_i', \cdots, \mathbf{r}_{q_H}'\}$ to answer to $\mathcal{A}$'s hash queries.

The forking lemma [23, Lemma 4] asserts that $\mathcal{A}$ outputs a new signature $(t_2', \mu_2^*, \Sigma_2^*)$, where $\Sigma_2^* = (\mathbf{d}_2', \mathbf{e}_2', \mathbf{z}_2')$ such that $\mathbf{e}_2' = \mathbf{r}_i'$ using the same hash query as in the first run (i.e., the i-th hash query). Recall that γ is the probability of a restart of FSBS. As before, if $t_2' \neq t^*$, then $\mathcal{B}$ aborts. If $\mathbf{e}_2' = \mathbf{e}_1'$, $\mathcal{B}$ aborts and replays $\mathcal{A}(pp, pk, \rho')$ at most $q_H^{q_S}$ times using different random tapes ρ' and different hash queries. If $\mathbf{e}_2' \neq \mathbf{e}_1'$, then $\mathcal{B}$ returns

$$((F_{t^*}\mathbf{z}'_1 - K\mathbf{e}_1', \mathsf{com}(\mu_1^*, \mathbf{d}_1')), (F_{t^*}\mathbf{z}'_2 - K\mathbf{e}_2', \mathsf{com}(\mu_2^*, \mathbf{d}_2')). \tag{2}$$

Since the pair in Equation (2) are both coming from the same hash query and com is computationally binding, we have $\mu_2^* = \mu_1^*$, $\mathbf{d}'_1 = \mathbf{d}'_2$ and

$$F_{t^*}\mathbf{z'}_1 - K\mathbf{e}'_1 = F_{t^*}\mathbf{z'}_2 - K\mathbf{e}'_2 \pmod q,$$

or equivalently,

$$F_{t^*}(\mathbf{z'}_1 - \mathbf{z'}_2 - S^*(\mathbf{e}'_1 - \mathbf{e}'_2)) = \mathbf{0} \pmod q.$$

Set $\widehat{\mathbf{v}} := \mathbf{z'}_1 - \mathbf{z'}_2 - S^*(\mathbf{e}'_1 - \mathbf{e}'_2)$. By Lemmas 4 and 6, there is at least one secret key S' such that $F_{t^*}S^* = F_{t^*}S' \pmod q$, where S^* and S' have all the same columns except the i-th column. The index i shows the position, where $\mathbf{e}'_1[i] \neq \mathbf{e}'_2[i]$. If $\mathbf{z'}_1 - \mathbf{z'}_2 - S^*(\mathbf{e}'_1 - \mathbf{e}'_2) = \mathbf{0}$, then we can choose $\widehat{v} := \mathbf{z'}_1 - \mathbf{z'}_2 - S'(\mathbf{e}'_1 - \mathbf{e}'_2) \neq \mathbf{0}$. Stress that the view of $\mathcal{A}$ is independent of both S^* and S'. We have shown that $\widehat{\mathbf{v}} \neq \mathbf{0}$ and $F_{t^*} \cdot \widehat{\mathbf{v}} = \mathbf{0} \pmod q$. It is easy to see that $\|\widehat{\mathbf{v}}\| \leq 2(\sigma_3 + \sigma\sqrt{\kappa})\sqrt{(1+\ell)m}$, as $\|S^*\| \leq \sigma\sqrt{(1+\ell)m}$, $\|\mathbf{z}'_i\| \leq \sigma_3\sqrt{(\ell+1)m}$, and $\|\mathbf{e}'_i\| \leq \sqrt{\kappa}$ for $i \in \{1, 2\}$.

In particular, we show that if $\mathcal{A}$ can produce a forgery by restarting the signing interaction (with $\mathcal{B}$), then $\mathcal{B}$ is able to find a solution to the l_2-SIS problem given by Equation (1). Indeed, to restart the signing interaction, $\mathcal{A}$ delivers $\mathsf{result}{:=}\ (\mathbf{a}, \mathbf{b}, \mathbf{e}', \mathbf{c})$ to $\mathcal{B}$. Now $\mathcal{B}$ with its view $\mathcal{V} = (t, \mathbf{r}, \mathbf{e}, \mathbf{z})$, will check whether all

$$\mathbf{e} - \mathbf{b} = \mathbf{e}' = H(\mathbf{x} + F_{t^*}\mathbf{a} + K\mathbf{b} \pmod q, \mathbf{c}), \tag{3}$$

$$\mathbf{e}' = H(F_{t^*}\mathbf{a} + F_{t^*}\mathbf{z} - K\mathbf{e}' \pmod q, \mathbf{c}), \tag{4}$$

$$\|\mathbf{z} + \mathbf{a}\| > \sigma_3\sqrt{(1+\ell)m}. \tag{5}$$

hold or not. If all are satisfied, $\mathcal{B}$ restarts the interaction with $\mathcal{A}$. Let assume that afterwards $\mathcal{A}$ successfully produces a valid signature $\widehat{\Sigma} = (\widehat{\mathbf{d}}', \widehat{\mathbf{e}}', \widehat{\mathbf{z}}')$. Let $\widehat{\mathbf{b}} \in \mathcal{D}_{\sigma_1}^m$ be such that $\mathbf{e} = \widehat{\mathbf{e}}' + \widehat{\mathbf{b}}$. Then, the following relations have to hold

$$\mathbf{e} - \widehat{\mathbf{b}} = \widehat{\mathbf{e}}' = H(\mathbf{x} + F_{t^*}\mathbf{a} + K\widehat{\mathbf{b}} \pmod q, \mathbf{c}), \tag{6}$$

$$\widehat{\mathbf{e}}' = H(F_{t^*}\widehat{\mathbf{z}}' - K\widehat{\mathbf{e}}' \pmod q, \mathsf{com}(\mu^*, \widehat{\mathbf{d}}')), \tag{7}$$

$$\|\widehat{\mathbf{z}}'\| \leq \sigma_3\sqrt{(1+\ell)m}. \tag{8}$$

Now, if $\widehat{\mathbf{e}}' \neq \mathbf{e}'$, then $\mathcal{B}$ aborts. Otherwise, Equations (4) and (7) give $F_{t^*}\mathbf{a} + F_{t^*}\mathbf{z} \pmod q = F_{t^*}\widehat{\mathbf{z}}' \pmod q$. Let $\widehat{\mathbf{v}} := \mathbf{a} + \mathbf{z} - \widehat{\mathbf{z}}'$, then $\widehat{\mathbf{v}} \neq 0$. This is true as otherwise $\mathbf{a} + \mathbf{z} = \widehat{\mathbf{z}}'$, which implies that $\|\mathbf{z} + \mathbf{a}\| \leq \eta\sigma_3\sqrt{m}$ (by Equation (8)). This contradicts Equation (5). Again, we have $F_{t^*} \cdot \widehat{\mathbf{v}} = \mathbf{0} \pmod q$, $\widehat{\mathbf{v}} \neq \mathbf{0}$ and $\|\widehat{\mathbf{v}}\| \leq \|\mathbf{a}\| + \|\mathbf{z}\| + \|\widehat{\mathbf{z}}'\| \leq (2\sigma_3 + \sigma_2)\sqrt{(1+\ell)m}$.

Note that $F_{t^*} = \left[A_0 \| A_1^{(t_1^*)} \| \cdots \| A_\ell^{(t_\ell^*)}\right] = \left[A_0 \| U_1^{(t_1^*)} \| \cdots \| U_\ell^{(t_\ell^*)}\right]$. We can get F from F_{t^*} by inserting into the gap between two sub-matrices in F_{t^*} the remaining matrices $\{U_i^{(1-t_i^*)}\}_i$ at relevant positions. We insert zeros into the corresponding position of $\widehat{\mathbf{v}}$ to get the desired solution $\mathbf{v}$ to the problem given by Equation (1). Obviously, $F \cdot \mathbf{v} = \mathbf{0} \pmod q$, and $\|\mathbf{v}\| = \|\widehat{\mathbf{v}}\|$.

Parameters	Value	Usage
n	–	Security parameter
ℓ	–	Binary tree depth
τ	2^ℓ	#time points
β	$\beta = \max\{(2\sigma_3 + 2\sigma\sqrt{\kappa})\sqrt{(1+\ell)m}, (2\sigma_3 + \sigma_2)\sqrt{(1+\ell)m}\}$	For l_2-$\mathsf{SIS}_{q,n,(1+2\ell)m,\beta}$ to be hard, Theorem 3
q	$q \geq \beta \cdot \omega(\sqrt{n \log n})$, prime	
m	$\max\{\frac{1}{1+\ell} \cdot (64 + \frac{n \log q}{\log(2d+1)}), \lceil 6n \log q \rceil\}$, $d = \sigma \cdot \sqrt{(1+\ell)m}$	Lemma 6, TrapGen
σ	$\geq O(\sqrt{n \log q}) \cdot \omega(\sqrt{\log n})$	SampleKey, Theorem 2
M_1, M_2, M_3	$M_1 = M_2 = M_3 = e^{1+1/288}$	Rejection sampling
σ_1	$12\sqrt{\kappa}$	
σ_2	$12\sigma\eta\sigma_1\sqrt{(1+\ell)mk}$	
σ_3	$12\eta\sigma_2\sqrt{m}$	
k, κ	$2^\kappa \cdot \binom{k}{\kappa} \geq 2^\gamma$	Min-entropy of the hash function H at least γ

Table 1. Choosing parameters for the proposed FSBS scheme

To summarise, we have shown that $\mathcal{B}$ can solve the l_2-$\mathsf{SIS}_{q,n,(1+2\ell)m,\beta}$ problem, with

$$\beta = \max\{(2\sigma_3 + 2\sigma\sqrt{\kappa})\sqrt{(1+\ell)m}, (2\sigma_3 + \sigma_2)\sqrt{(1+\ell)m}\}.$$

□

Remark 2. In the proof for the forward-secure unforgeability, one may think of the method of programming hash values, instead of using the real signing interaction (with a modification in generating the matrix S_t to compute $\mathbf{z} = \mathbf{r} + S_t\mathbf{e}$) in order to reply signing queries issued by $\mathcal{A}$. We argue that the programming method fails to simulate the perfect environment for the adversary $\mathcal{A}$. Assume that $\mathcal{B}$ does not want to compute S_t in the way we have done in our proof. Then, after replying to a hash query, say $(\mathbf{x} + F_t\mathbf{a} + K\mathbf{b} \pmod q, \mathsf{com}(\mu, \mathbf{d}'))$, by giving a hash value, say $\mathbf{e}'$, $\mathcal{B}$ simply chooses $\mathbf{z} \leftarrow \mathcal{D}_{\sigma_2}^{(\ell+1)m}$ and then sends $\mathbf{z}$ to $\mathcal{A}$. In turn, $\mathcal{A}$ gives $\mathbf{e} := \mathbf{e}' + \mathbf{b}$ to $\mathcal{B}$. After that $\mathcal{B}$ sets $H(F_t\mathbf{a} + F_t\mathbf{z} - K\mathbf{e}', \mathsf{com}(\mu, \mathbf{d}')) := \mathbf{e}'$. However, since the collision resistance of H, the relation $F_t\mathbf{a} + F_t\mathbf{z} - K\mathbf{e}' = \mathbf{x} + F_t\mathbf{a} + K\mathbf{b} \pmod q$ has to hold. Thus, $\mathcal{A}$ needs to check whether or not $F_t\mathbf{z} = \mathbf{x} + K\mathbf{e} \pmod q$ to distinguish the simulated signing interaction from the real one. One may think that $\mathcal{B}$ can choose $\mathbf{z} \leftarrow \mathcal{D}_{\sigma_2}^{(\ell+1)m}$ such that $F_t\mathbf{z} = \mathbf{x} + K\mathbf{e} \pmod q$ before sending $\mathbf{z}$ to $\mathcal{A}$. However, without the knowledge of a trapdoor for F_t, the problem of choosing such a $\mathbf{z}$ is not easy.

5.4 Choosing Parameters

First, we set n as security parameter, ℓ as the highest depth of the binary tree representing time points, $\tau = 2^\ell$ as the number of time points. For TrapGen,

we need $m \geq \lceil 6n \log q \rceil$. For SampleKey (Theorem 2) to work, we need $\sigma \geq O(\sqrt{n \log q}) \cdot \omega(\sqrt{\log n})$. Also, let $d := \sigma\sqrt{(1+\ell)m}$ and we set $(\ell+1)m \geq 64 + n \log q / \log(2d+1)$ via Lemma 6. To make sure the min-entropy of H is at least γ, we choose k and κ such that $2^\kappa \cdot \binom{k}{\kappa} \geq 2^\gamma$. Section 5.1 suggests setting $M_i := e^{1+1/288}$ for all $i \in [3]$. We then set $\sigma_1 = 12\|\mathbf{e}'\| = 12\sqrt{\kappa}$, $\sigma_2 = 12\|S^*\mathbf{e}\| = 12\sigma\sigma_1\sqrt{(1+\ell)mk}$ and $\sigma_3 = 12\|\mathbf{z}\| = 12\eta\sigma_2\sqrt{(1+\ell)m}$ (via *Remark* 1). For l_2-$\mathsf{SIS}_{q,n,(1+\ell)m,\beta}$ to be hard by Theorem 3, we set m poly-bounded, $\beta = poly(n)$ and $q \geq \beta \cdot \omega(\sqrt{n \log n})$, where $\beta = \max\{(2\sigma_3 + 2\sigma\sqrt{\kappa})\sqrt{(1+\ell)m}, (2\sigma_3 + \sigma_2)\sqrt{(1+\ell)m}\}$. The parameter setting is summarized in Table 1.

6 Conclusions and Future Works

In this paper, we propose, for the first time, a forward-secure blind signature based on the hardness of the SIS problem in lattices. Using the rejection sampling technique together with the trapdoor delegation and the binary tree structure for representing of time periods, the proposed signature is blind and forward secure. Forward security is proven in the random oracle setting. Lattice-based forward-secure blind signatures in the standard model should be an interesting topic for future research.

Acknowledgment. We all would like to thank anonymous reviewers for their helpful comments. This work is partially supported by the Australian Research Council Discovery Project DP200100144 and Linkage Project LP190100984. Huy Quoc Le has been sponsored by a Data61 PhD Scholarship. Ha Thanh Nguyen Tran acknowledges the support of the Natural Sciences and Engineering Research Council of Canada (NSERC) (funding RGPIN-2019-04209 and DGECR-2019-00428). Josef Pieprzyk has been supported by the Australian ARC grant DP180102199 and Polish NCN grant 2018/31/B/ST6/03003.

References

1. Abdalla, M., Reyzin, L.: A new forward-secure digital signature scheme. In: Okamoto, T. (ed.) Advances in Cryptology — ASIACRYPT 2000. pp. 116–129. Springer Berlin Heidelberg, Berlin, Heidelberg (2000)
2. Agrawal, S., Boneh, D., Boyen, X.: Efficient lattice (H)IBE in the standard model. In: Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Monaco / French Riviera, May 30 - June 3, 2010. Proceedings. pp. 553–572 (2010). https://doi.org/10.1007/978-3-642-13190-5_28, https://doi.org/10.1007/978-3-642-13190-5_28
3. Ajtai, M.: Generating Hard Instances of the Short Basis Problem. In: Automata, Languages and Programming, 26th International Colloquium, ICALP'99, Prague, Czech Republic, July 11-15, 1999, Proceedings. pp. 1–9 (1999). https://doi.org/10.1007/3-540-48523-6_1, https://doi.org/10.1007/3-540-48523-6_1

4. Alwen, J., Peikert, C.: Generating shorter bases for hard random lattices. In: 26th International Symposium on Theoretical Aspects of Computer Science, STACS 2009, February 26-28, 2009, Freiburg, Germany, Proceedings. pp. 75–86 (2009). https://doi.org/10.4230/LIPIcs.STACS.2009.1832, https://doi.org/10.4230/LIPIcs.STACS.2009.1832
5. Anderson, R.: Two remarks on public key cryptology. Technical Report, University of Cambridge, Computer Laboratory (2002), https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-549.pdf
6. Bellare, M., Miner, S.K.: A forward-secure digital signature scheme. In: Wiener, M. (ed.) Advances in Cryptology — CRYPTO' 99. pp. 431–448. Springer Berlin Heidelberg, Berlin, Heidelberg (1999)
7. Boyd, C., Gellert, K.: A Modern View on Forward Security. Cryptology ePrint Archive, Report 2019/1362 (2019), https://eprint.iacr.org/2019/1362
8. Canetti, R., Halevi, S., Katz, J.: A forward-secure public-key encryption scheme. In: Biham, E. (ed.) Advances in Cryptology — EUROCRYPT 2003. pp. 255–271. Springer Berlin Heidelberg, Berlin, Heidelberg (2003)
9. Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai trees, or how to delegate a lattice basis. In: Gilbert, H. (ed.) Advances in Cryptology – EUROCRYPT 2010. pp. 523–552. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
10. Chaum, D.: Blind Signatures for Untraceable Payments. In: Chaum, D., Rivest, R.L., Sherman, A.T. (eds.) Advances in Cryptology. pp. 199–203. Springer US, Boston, MA (1983)
11. Chow, S.S.M., Hui, L.C.K., Yiu, S.M., Chow, K.P.: Forward-secure multisignature and blind signature schemes. Applied Mathematics and Computation **168**, 895–908 (09 2005). https://doi.org/10.1016/j.amc.2004.09.015
12. Diffie, W., Oorschot, P., Wiener, M.: Authentication and authenticated key exchanges. Des Codes Crypt **2**, 107–125 (03 1992). https://doi.org/10.1007/BF00124891
13. Duc, D.N., Cheon, J.H., Kim, K.: A Forward-Secure Blind Signature Scheme Based on the Strong RSA Assumption. In: Qing, S., Gollmann, D., Zhou, J. (eds.) Information and Communications Security. pp. 11–21. Springer Berlin Heidelberg, Berlin, Heidelberg (2003)
14. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. Cryptology ePrint Archive, Report 2007/432 (2008), https://eprint.iacr.org/2007/432
15. Günther, C.G.: An Identity-Based Key-Exchange Protocol. In: Quisquater, J.J., Vandewalle, J. (eds.) Advances in Cryptology — EUROCRYPT '89. pp. 29–37. Springer Berlin Heidelberg, Berlin, Heidelberg (1990)
16. Itkis, G., Reyzin, L.: Forward-secure signatures with optimal signing and verifying. In: Kilian, J. (ed.) Advances in Cryptology — CRYPTO 2001. pp. 332–354. Springer Berlin Heidelberg, Berlin, Heidelberg (2001)
17. Jia, Y., Fanyu, K., Cheng, X., Rong, H., Yangkui, C., Xuliang, L., Guowen, L.: Forward-Secure Multisignature, Threshold Signature and Blind Signature Schemes. Journal of Networks **5** (06 2010). https://doi.org/10.4304/jnw.5.6.634-641
18. Kucharczyk, M.: Blind signatures in electronic voting systems. In: Kwiecień, A., Gaj, P., Stera, P. (eds.) Computer Networks. pp. 349–358. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
19. Le, H.Q., Duong, D.H., Susilo, W.: A blind ring signature based on the short integer solution problem. In: You, I. (ed.) Information Security Applications. pp. 92–111. Springer International Publishing, Cham (2020)

20. Ling, S., Nguyen, K., Wang, H., Xu, Y.: Forward-secure group signatures from lattices. In: Ding, J., Steinwandt, R. (eds.) Post-Quantum Cryptography. pp. 44–64. Springer International Publishing, Cham (2019)
21. Lyubashevsky, V.: Lattice signatures without trapdoors. Cryptology ePrint Archive, Report 2011/537, Full version of paper appearing at Eurocrypt 2012, last revised 18 Oct 2017 (2012), https://eprint.iacr.org/2011/537
22. Micciancio, D., Regev, O.: Worst-case to average-case reductions based on gaussian measures. In: 45th Symposium on Foundations of Computer Science (FOCS 2004), 17-19 October 2004, Rome, Italy, Proceedings. pp. 372–381 (2004). https://doi.org/10.1109/FOCS.2004.72, https://doi.org/10.1109/FOCS.2004.72
23. Pointcheval, D., Stern, J.: Provably secure blind signature schemes. In: Kim, K., Matsumoto, T. (eds.) Advances in Cryptology — ASIACRYPT '96. pp. 252–265. Springer Berlin Heidelberg, Berlin, Heidelberg (1996)
24. Rückert, M.: Lattice-based Blind Signatures. In: Abe, M. (ed.) Advances in Cryptology - ASIACRYPT 2010. pp. 413–430. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
25. Zhang, P., Jiang, H., Zheng, Z., Hu, P., Xu, Q.: A New Post-Quantum Blind Signature From Lattice Assumptions. IEEE Access **6**, 27251–27258 (2018). https://doi.org/10.1109/ACCESS.2018.2833103