Sensors and Materials, Vol. 33, No. 5 (2021) 1565–1575
MYU Tokyo

S & M 2555

Proxy Signature for Sensor Networks against Cyber Attack

Jinbin Zheng,[1] Huang Zhang,[2] Baodian Wei,[3*] Yusong Du,[3] and Hsien-Wei Tseng[1**]

[1]School of Mathematics and Information Engineering, Longyan University, Fujian 364012, China
[2]School of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha 410114, China
[3]School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou 510006, China

(Received July 20, 2020; accepted March 19, 2021)

Keywords: bonsai trees, lattice, proxy signature, quantum-resistant, standard model, sensor networks

The Internet of Things (IoT) needs complex sensor networks. The increasing number of sensors and their networks underlines the importance of network security. Traditional digital signature schemes validate signatures from each node of networks, which need huge resources for storage and computation. To reduce the resources needed for security, a new scheme with a proxy signature is proposed. If an IoT application adopts the scheme and assigns its signing rights to the underlying nodes, the storage of their public keys is no longer necessary as the application trusts the signing rights. The development of computing technology introduces many risks to cryptographic schemes with a traditional digital signature, and a system with improved security at the quantum-resistant level is required. We propose a scheme with lattice-based cryptography using the bonsai-tree technique for a security system. The new scheme effectively protects the system from a chosen static message attack in the standard model. In the new scheme, an IoT application securely recognizes its honest nodes in the underlying sensor networks at a low storage cost, even in quantum computing.

1. Introduction

A sensor network is essential in the perceptual recognition layer of the IoT. The security of sensor networks depends on IoT applications distinguishing whether data is submitted from honest or malicious nodes. A conventional digital signature requires IoT applications to validate signatures and data in a node. This requires all IoT applications to record all the public keys of the honest nodes in the underlying sensor networks, leading to huge storage and searching costs. A proxy signature scheme is a suitable alternative for solving the problems arising from using conventional schemes. A proxy signature scheme is designed for the social requirement of assigning signatures. When the original signer temporarily permits a colleague, i.e., a proxy signer, the right to use their signature, the proxy signer can sign on behalf of the original signer in a proxy signature scheme. The theory of this concept was proposed by Mambo *et al.*(1) They categorized proxy signature schemes into full delegation, partial delegation, and delegation by a warrant scheme. The scheme then developed rapidly owing to increasing demand. Proxy

*Corresponding author: e-mail: weibd@mail.sysu.edu.cn
**Corresponding author: e-mail: hsienwei.tseng@gmail.com
https://doi.org/10.18494/SAM.2021.3044

ISSN 0914-4935 © MYU K.K.
https://myukk.org/

signature schemes require security systems that usually employ the factoring problem (FP) and the discrete logarithmic problem (DLP).(2–5) However, the polynomial-time quantum algorithms of quantum computing make existing security systems less secure. Thus, a proxy signature scheme that defends against attack by quantum computers is worth developing. As a new proxy signature scheme, a lattice-based cryptosystem is attracting increasing interest owing to its quantum resistance.(6) Kawachi *et al.* proposed multibit encryption for lattice problems.(7) Regev studied a new worst case of a case problem learning with error and built secured public encryption under a chosen-plaintext attack.(8) Goldreich *et al.* proposed the first lattice-based signature,(9) although it was not effective for ensuring security. In response, Gentry *et al.* designed a Gaussian sampling algorithm for hard uniform random lattices and proposed a lattice-based signature scheme based on the random oracle model.(10) Cash *et al.* designed a lattice-based signature scheme to protect against static chosen-message attack (EU-SCMA) based on the standard model.(11) There have been other studies on lattice-based cryptographic schemes,(12,13) some of which are worth reviewing and developing further. Jiang *et al.* proposed a proxy scheme of a new lattice-based signature by using the technique of bonsai trees.(14) However, their scheme was designed and analyzed for a random oracle model, which is not secure enough in general. In response, we propose in this paper a new scheme that eliminates the flaws of previous schemes. This scheme is for a secured proxy signature in the standard model.

2. Theoretical Background

2.1 Notations

We use $\mathbb{Z}$, $\mathbb{Z}^+$, $\mathbb{N}$, and $\mathbb{R}$ for the sets of integers, positive integers, natural numbers, and all real numbers, respectively. For $k \in \mathbb{Z}^+$, [k] denotes the set $\{x \mid 1 \le x \le k\}$. Here, column vectors are written in bold lower case (e.g., $\mathbf{x}$) and matrices in bold upper case (e.g., $\mathbf{X}$). A matrix $\mathbf{X}$ is identified with the set $\{\mathbf{x}_i\}$ of its column vectors. $\mathbf{X}_1 \| \mathbf{X}_2$ presents the ordered concatenation of sets $\mathbf{X}_1$ and $\mathbf{X}_2$. For a vector $\mathbf{x} \in \mathbb{Z}^n$, $\|\mathbf{x}\|_p$ represents its l_p norm and p is omitted if $p = 2$. Similarly, the l_p norm of a matrix $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$. is defined by $\max_{i\in[n]}\|\mathbf{x}_i\|$. The standard big O notation and $negl(\cdot)$ (for a negligible function) are used for any polynomial function $poly(\cdot)$. For a sufficiently large $n \in \mathbb{N}$, $negl(n) < 1/poly(n)$. PPT means probabilistic polynomial time.

2.2 Lattice

Let $\mathbb{R}^m$ be a Euclidean space of m dimensions. An integer lattice in $\mathbb{R}^m$ is included in the following set:

$$\zeta(\mathbf{b}_1, \cdots, \mathbf{b}_n) = \left\{ \sum_{i=1}^{n} x_i \mathbf{b}_i \mid x_i \in \mathbb{Z} \right\}, \tag{1}$$

where the integral combinations of n linearly independent vectors are $\mathbf{b}_1, \ldots, \mathbf{b}_n$ in $\mathbb{Z}^m$ ($m \ge n$), and integers n and m are the rank and dimension of the lattice, respectively. The lattice is fully ranked if $m = n$. The sequence of vectors $\mathbf{b}_1, ..., \mathbf{b}_n$ is lattice-based, and these vectors are the base column vectors of matrix $\mathbf{B} = [\mathbf{b}_1, ...,\mathbf{b}_n]$. The set in Eq. (1) is rewritten as the compact set

$$\zeta(\mathbf{B}) = \{\mathbf{Bx} \mid x \in \mathbb{Z}^n\}, \tag{2}$$

where $\mathbf{Bx}$ is the matrix-vector product. There are an infinite number of forms of the same lattice.

This paper includes a discussion on a certain family of integer lattices studied by Ajtai.(6) When $n \geq 1$ and modulus $q \geq 2$, where n and q are integers, the dimension n becomes the main cryptographic security parameter. All other parameters are implicitly expressed as a function of n. Then, an m-dimensional lattice is defined as

$$\Lambda^{\perp}(\mathbf{A}) = \left\{\mathbf{x} \in \mathbb{Z}^m \mid \mathbf{Ax} = 0 \bmod q\right\} \subseteq \mathbb{Z}^m. \tag{3}$$

Any $\mathbf{y}$ in the subgroup $\mathbb{Z}_q^n$ is generated by the column vectors of $\mathbf{A}$. Then, the lattice $\Lambda^{\perp}(\mathbf{A})$ is defined as

$$\Lambda_{\mathbf{y}}^{\perp}(\mathbf{A}) = \left\{x \in \mathbb{Z}^m \mid \mathbf{Ax} = \mathbf{y} \bmod q\right\} = \Lambda^{\perp}(\mathbf{A}) + \overline{\mathbf{x}}, \tag{4}$$

where $\overline{\mathbf{x}} \in \mathbb{Z}^m$ is an arbitrary solution of $\mathbf{Ax} = \mathbf{y} \bmod q$.(4)

For a constant $C > 1$ and $m \geq Cn\log q$, the columns of a random $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ generate all of $\mathbb{Z}_q^n$, except when $2^{-\Omega(n)} = negl(n)$. Therefore, a uniform $\mathbf{A}$ is assumed to generate $\mathbb{Z}_q^n$.(8)

In the security proofs of our proxy signature, we need the hardness assumption for the short integer solution (SIS) problem. A $\mathrm{SIS}_{q,\beta}$ problem in an l_2 norm is solved with a uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ for any $m = poly(n)$. Then, a nonzero integer vector $\mathbf{v} \in \mathbb{Z}^m$ is found in $\|\mathbf{v}\| \leq \beta$ and $\mathbf{Av} = 0 \bmod q$, that is, $\mathbf{v} \in \Lambda^{\perp}(\mathbf{A})$. The advantageousness of an algorithm S for solving $\mathrm{SIS}_{q,\beta}$ is defined as its probability of finding a correct answer and is denoted by $Adv_{SIS_{q,\beta}}(S)$. Obtaining a solution of the SIS is as difficult as approximating the shortest vector problem, one of the lattice problems.

2.3 Gaussian measures of lattices

The Gaussian measures of lattices satisfy the solution proposed by Gentry *et al.*(14) The Gaussian function at $\mathbf{c} \in \mathbb{R}^m$ centered on $\mathbb{R}$ with $s > 0$ is defined as

$$\forall \mathbf{x} \in \mathbb{R}^m,\ \rho_{s,\mathbf{c}}(\mathbf{x}) = \exp\left(-\pi \frac{\|\mathbf{x} - \mathbf{c}\|^2}{s^2}\right). \tag{5}$$

Here, s and $\mathbf{c}$ are 1 and $\mathbf{0}$, respectively.

For $\mathbf{c} \in \mathbb{R}^m$, $s > 0$, and Λ of an n-dimensional lattice, the discrete Gaussian distribution over Λ

is defined as

$$\forall \mathbf{x} \in \Lambda,\ D_{\Lambda,s,\mathbf{c}}(\mathbf{x}) = \frac{\rho_{s,\mathbf{c}}(\mathbf{x})}{\rho_{s,\mathbf{c}}(\Lambda)}. \tag{6}$$

The denominator in Eq. (6) is a normalization factor. The probability $D_{\Lambda,s,\mathbf{c}}(\mathrm{x})$ is related to $\rho_{s,\mathbf{c}}(\mathbf{x})$. The discrete Gaussian distribution over the coset of lattices is defined similarly except for the support set $\Lambda + y$, which is $\Lambda_{\mathbf{y}}^{\perp}(\mathbf{A})$ in this study.

Several definitions from previous studies on discrete Gaussians over lattices are summarized as follows.

Lemma 1: When S is a basis of $\Lambda^{\perp}(\mathbf{A})$ for $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ whose columns generate $\mathbb{Z}_q^n$, $\mathbf{y} \in \mathbb{Z}_q^n$ is arbitrary, and $s \geq \|\tilde{\mathbf{S}}\| \cdot \omega(\sqrt{\log n})$, the following hold.

(1) Lemma 1.1:(15) $\Pr_{\mathbf{x} \leftarrow D_{\Lambda_{\mathbf{y}}^{\perp}(\mathbf{A}),s}}[\|\mathbf{x}\| > s \cdot \sqrt{m}] \leq negl(n)$.

(2) Lemma 1.2:(16) $\Pr_{\mathbf{x} \leftarrow D_{\Lambda^{\perp}(\mathbf{A}),s}}[\mathbf{x} = 0] \leq negl(n)$.

(3) Corollary 1.1:(8) A set of $O(m^2)$ with independent samples from $D_{\Lambda_{\mathbf{y}}^{\perp}(\mathbf{A}),s}$ contains a set of m linearly independent vectors, except with the probability of negl(n).

(4) Proposition 1.1:(14) When $\mathbf{x} \leftarrow D_{\mathbb{Z}^m,s}$, the marginal distribution $\mathbf{y} = \mathbf{A}\mathbf{x} \in \mathbb{Z}_q^n$ becomes uniform up to the statistical distance of negl(n). The conditional distribution of $\mathbf{x}$ with $\mathbf{y}$ is defined as $D_{\Lambda_{\mathbf{y}}^{\perp}(\mathbf{A}),s}$, from which a PPT algorithm SamplePre(S,$\mathbf{A}$,$\mathbf{y}$,s) is generated.

(5) Theorem 1.1:(14) A PPT algorithm SampleD(S,$\mathbf{A}$,$\mathbf{c}$,s) generates a sample algorithm from $D_{\Lambda_{\mathbf{y}}^{\perp}(\mathbf{A}),s,c}$.

Thus, SampleD(S,$\mathbf{A}$,$\mathbf{c}$,s), a Gaussian sampling algorithm, is an important tool in lattice-based cryptographic systems. Peikert improved its performance so that we can adopt the new algorithm in our scheme.(17)

2.4 Algorithm for lattice-based cryptographic system

The trapdoor in lattice-based cryptographic systems has a “short basis”. There are useful algorithms for lattice-based cryptographic systems.

Proposition 1:(18) For a constant $C > 1$, the PPT algorithm GenBasis(1^n,1^m,q) outputs $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and $\mathbf{S} \in \mathbb{Z}^{m\times m}$ for $poly(n) \geq m \geq Cn\log q$ as follows.

- The distribution of $\mathbf{A}$ within a statistical distance of uniform negl(n)
- $\mathbf{S}$: a basis of $\Lambda^{\perp}(\mathbf{A})$.
- $\|\tilde{\mathbf{S}}\| \leq \tilde{L} = O(\sqrt{n \log q})$.

Lemma 2:(10) The deterministic polynomial-time algorithm ExtBasis with the following properties is explained as follows: The columns of an arbitrary $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ generate the entire group $\mathbb{Z}_q^n$, an arbitrary $\mathbf{S} \in \mathbb{Z}^{m\times m}$ of $\Lambda^{\perp}(\mathbf{A})$, and $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n\times \bar{m}}$. Here, ExtBasis$(\mathbf{S}, \mathbf{A_B} = \mathbf{A} \,\|\, \bar{\mathbf{A}})$ outputs S of $\Lambda^{\perp}(\mathbf{A}') \subseteq \mathbb{Z}^{m+\bar{m}}$ such as $\|\tilde{\mathbf{S}}'\| = \|\tilde{\mathbf{S}}\|$. The same holds for any permutated columns of $\mathbf{A}'$.

The algorithm of ExtBasis can extend the lattice and its basis to an arbitrarily high dimension with no quality loss of the basis.

Lemma 3:[10] The PPT algorithm RandBasis($\mathbf{S}$,s) is based on $\mathbf{S}$ of a lattice Λ of m dimensions and a parameter $s \geq \|\tilde{\mathbf{S}}\| \cdot \omega(\sqrt{\log n})$. It yields a basis $\mathbf{S}'$ of Λ such as $\|\mathbf{S}'\| \leq s \cdot \sqrt{m}$. For two bases $\mathbf{S}_0$ and $\mathbf{S}_1$ of the same lattice with the condition $s \geq \max\{\|\widetilde{\mathbf{S}_0}\| \| \|\widetilde{\mathbf{S}_1}\|\} \cdot \omega(\sqrt{\log n})$, RandBasis($\mathbf{S}_0$,$s$) and ($\mathbf{S}_1$,$s$) are within the statistical distance *negl*(n).

2.5 Shortest non-prefix set

The security proof of our proxy signature requires a special set P that is also briefly introduced in Ref. (10). To improve the readability, we define it as the shortest non-prefix (SNP) set and give its properties in this subsection.

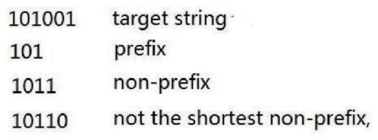
Definition 1: For a list that consists of distinct strings $\mathbf{u}^1$, ..., $\mathbf{u}^Q \in \{0,1\}^k$, the SNP P is a non-empty set of strings $\mathbf{p} \in \{0,1\}^{\leq k}$ in which $\mathbf{p}$ is the shortest string. No $\mathbf{u}^i$ has $\mathbf{p}$ as a prefix in the string.

Figure 1 is an example of an SNP. In the string "101001", the first three bits "101" are the prefix. As "1011" does not match the first four bits of "101001", it is judged as a non-prefix of the string. If the last bit of "1011" is removed, "101" becomes a prefix of the string, which implies that "1011" is the SNP.

A string space $\{0,1\}^k$ can be linked with a k+1-depth full binary tree in which each edge is labeled by 0 or 1. Figure 2 exhibits an example for $k = 3$. Each path from the root to a leaf node specifies a string $\{0,1\}^k$. For example, if $\mathbf{u}^1$ = "101", the corresponding node of $\mathbf{u}^1$ is μ (Fig. 3). The SNP set of $\{\mathbf{u}^1\}$ is $P = \{a, b, c\}$.

A list of distinct strings in an SNP set can be generated efficiently by running the algorithm proposed in Ref. 5. A simple example of the SNP set for $\mathbf{u}$ = "101" and $\boldsymbol{\pi}$ = "010" is shown in Fig. 4. The following theorem gives an upper bound for the size of the SNP set depending on the algorithm.

Theorem 1: For a string list $\mu^{(1)}, \mu^{(2)}, ..., \mu^{(Q)} \in \{0,1\}^k$, the size of the SNP set P is at most $(k-1) \cdot Q + 1$.



101001	target string
101	prefix
1011	non-prefix
10110	not the shortest non-prefix,

Fig. 1. Example of SNP.

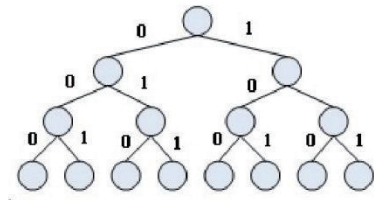


Fig. 2. Full binary tree corresponding to $\{0,1\}^3$.

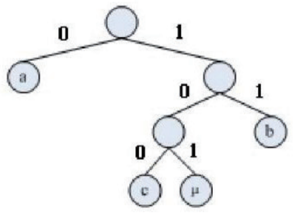


Fig. 3. SNP set of {$\mathbf{u}^1$ = "101"}.

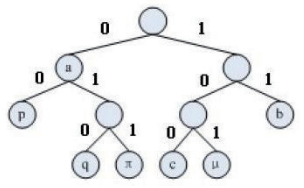


Fig. 4. SNP set of {$\mathbf{u}^1$ = "101", $\boldsymbol{\pi}$ = "101"}.

3. Proxy Signature Scheme

3.1 Syntax

A proxy signature consists of seven efficient algorithms, PS = (*InstGen*(·), *KeyGen*(·), *Sign*(·), *Verify*(·), *ProxcyKeyGen*(·), *ProxySign*(·), *ProxyVerify*(·)), which stand for instance generation, key generation, signing, verifying, proxy key generation, proxy signing, and proxy signature verifying, respectively.

(1) $params \leftarrow InstGen(1^n)$: Using a security parameter n, this algorithm generates the global parameter *params* to set up the proxy signature system; *params* will be a default input for the other algorithms.

(2) $(pk,sk) \leftarrow KeyGen(\cdot)$: This algorithm generates a key pair (pk,sk) for a client that is invoked by this algorithm.

(3) $\sigma_1 \leftarrow Sign(pk,sk,\mathbf{m})$: If a client decides to sign a message **m** directly, the algorithm will be executed with the client's key pair (pk,sk) as input. After that, a signature σ_1 is returned.

(4) $b \leftarrow Verify(pk,\mathbf{m},\sigma_1)$: Upon input **m**, the public key of the original signer pk, and the corresponding signature σ_1, this algorithm returns $b = 1$ if the signature is valid and $b = 0$ otherwise.

(5) $sk_{12} \leftarrow ProxcyKeyGen(pk_1,sk_1,pk_2)$: This algorithm is executed by the original signer. It requests the original signer's key pair (pk_1,sk_1) and the public key pk_2 of the proxy as the input, and it outputs a proxy secret key sk_{12}.

(6) $\sigma_2 \leftarrow ProxyVerify(pk_1,pk_2,sk_2,sk_{12},\mathbf{m})$: If the proxy signer wishes to sign **m** on behalf of the original signer, the signer invokes the algorithm with the original signer's public key pk_1, his/her own key pair (pk_2, sk_2), and the proxy secret key sk_{12}. The algorithm returns a corresponding proxy signature σ_2.

(7) $b \leftarrow ProxyVerify(pk_1,pk_2,\mathbf{m},\sigma_2)$: The algorithm is executed by a verifier. Upon the input of the original signer's and proxy signer's public keys pk_1, pk_2, respectively, the message m, and its corresponding proxy signature σ_2, this algorithm returns $b = 1$ if the signature is valid and $b = 0$ otherwise.

The correctness of the proxy signature scheme states that any signature generated by *Sign*(·) or *ProxySign*(·) should pass the check of the corresponding verification algorithm *Verify*(·) or *ProxyVerify*(·), respectively

On the other side, the security of a signature is defined by an experiment (also called a game), in which there is existential unforgeability under a static chosen-message attack (EU-SCMA). The experiment was carried out as follows for the adversary F and the challenger C.

(1) F outputs query messages $\mu_1,\ldots, \mu_0$ for Q and sends the list to C.

(2) C performs the key generation procedure to obtain $(pk,sk) \leftarrow Gen(\cdot)$ and generates the signature $\sigma_i \leftarrow Sign(\mu_i,sk)$ for $i \in [Q]$. Then, F is given pk, σ_i $(i \in [Q])$.

(3) F outputs an attempted forgery (μ^*,σ^*).

The advantage $Adv_{PS}^{EU-SCMA}(F)$ of F is defined as the probability of accepting $Ver(pk,\mu^*,\sigma^*)$ and $\mu^* \neq \mu_i$ when $i \in Q$. The probability is calculated for all uniform randomnesses in the experiment. A proxy signature scheme is secure if $Adv_{PS}^{EU-SCMA}(F) = negl(n)$.

Note that the EU-SCMA is weaker than existential unforgeability under adaptive chosen-message attacks that adaptively choose the message μ_i with *pk* given to *F*.

3.2 Our construction

Without loss of generality, the original signer is called Alice and the proxy signer Bob. The proxy signature involves the following seven PPT algorithms: *PS* = (*InstGen*(·), *KeyGen*(·), *Sign*(·), *Verify*(·), *ProxcyKeyGen*(·), *ProxySign*(·), *ProxyVerify*(·)).

(1) *InstGen*(1^n): Upon input n, this procedure computes $q = poly(n)$, the dimension of lattices $m = O(n\log q)$, and $\widetilde{L} = O(\sqrt{n\log q})$. Let $h(\cdot)$: $\{0,1\}^{n\times m} \to \{0,1\}^m$ and $H(\cdot)$: $\{0,1\}^* \to \{0,1\}^k$ be two cryptographic hash functions. $s = \widetilde{L}\cdot\omega(\sqrt{\log q})$ is the Gaussian parameter. Matrices $\mathbf{B}_i^{(b)} \in \mathbb{Z}_q^{n\times m}$ are sampled uniformly at random for $i \in [k]$, $b \in \{0,1\}$. For $j \in [m]$, $b \in \{0,1\}$ and $\mathbf{c}_j^{(b)} \in \mathbb{Z}_q^n$ are uniformly chosen vectors.

(2) *KeyGen*(1^n,1^m,q) : This procedure runs *GenBasis*(1^n,1^m,q) to obtain the key pair (**A**,**S**), where $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ is the public key, $\mathbf{S} \in \mathbb{Z}^{m\times m}$ is the private key, and $\|\widetilde{\mathbf{S}}\| \le \widetilde{L}$. Both the original signer Alice and the proxy signer Bob use this procedure to obtain their public–private key pairs. Alice and Bob hold (**A**,**S**) and ($\mathbf{A_B}$,$\mathbf{S_B}$), respectively.

(3) *Sign*(**A**,**S**,**m**): Upon inputting the key pair (**A**,**S**) of Alice and a message $\mathbf{m} \in \{0,1\}^*$, the algorithm first computes $\mathbf{u} = H(\mathbf{m})$. Then, it chooses the matrix $\mathbf{B}_i^{(b_i)}$ for each bit of **u** and concatenates them as $\mathbf{B} = \mathbf{B}_1^{(u_1)} \| \cdots \| \mathbf{B}_k^{(u_k)}$.

(4) *ProxcyKeyGen*(**A**,**S**,$\mathbf{A_B}$): Using the public and private keys of Alice, this algorithm first computes $\theta = h(\mathbf{A_B})$. Then, it chooses a suitable vector $\mathbf{c}_j^{(b)}$ from the system parameters based on each bit of θ and concatenates the vectors as $\mathbf{C} = \mathbf{c}_1^{(\theta_1)} \| \cdots \| \mathbf{c}_m^{(\theta_m)}$. After that, it runs the algorithm *ExtBasis*(**S**,**A**‖**C**) to obtain the temporary short basis $\overline{\mathbf{S}}$ and executes the algorithm RandBasis($\overline{\mathbf{S}}$,s) to uniformly randomize $\overline{\mathbf{S}}$ as $\mathbf{S}_\delta$. Finally, Alice sends the proxy signing key $\mathbf{S}_\delta$ to Bob. Bob can verify the validity of the key with the equation $(\mathbf{A}\|\mathbf{C}) \cdot \mathbf{S_\delta} = 0 \bmod q$.

(5) *ProxcySign*(**A**,$\mathbf{A_B}$,$\mathbf{S_B}$,$\mathbf{S_\delta}$,**m**) For $\mathbf{m} \in \{0,1\}^*$, $\mathbf{u} = H(\mathbf{m})$ is computed. Then, matrix $\mathbf{B}_i^{(b_i)}$ is chosen for each bit of **u** and the matrices are concatenated as $\mathbf{B} = \mathbf{B}_1^{(u_1)} \| \cdots \| \mathbf{B}_k^{(u_k)}$. Similarly, $\mathbf{C} = \mathbf{c}_1^{(\theta_1)} \| \cdots \| \mathbf{c}_m^{(\theta_m)}$, where $\theta = h(\mathbf{A_B})$. After that, the algorithm *ExtBasis*($\mathbf{S_\delta}$, **A**‖**C**‖**B**) obtains a short basis $\mathbf{S}_1$ of the lattice $\Lambda^\perp(\mathbf{A}\|\mathbf{C}\|\mathbf{B})$. Then, the algorithm *ExtBasis*($\mathbf{S_B}$,$\mathbf{A_B}$‖**B**)computes $\mathbf{S}_2$ of the lattice $\Lambda^\perp(\mathbf{A_B}\|\mathbf{B})$. Finally, it computes

$$\mathbf{e}_\delta \leftarrow SampleD(\mathbf{S_1},(\mathbf{A}\|\mathbf{C}\|\mathbf{B}),0,s), \tag{7}$$

$$\mathbf{e} \leftarrow \mathrm{SampleD}(\mathbf{S_2},\mathbf{A_B}\|\mathbf{B}, 0, s), \tag{8}$$

to output ($_\mathrm{m}$,$\mathbf{e_\delta}$,**e**) as the proxy signature.

(6) Verify(**A**,$\mathbf{A_B}$,**m**,$\mathbf{e_\delta}$,**e**): If $\|\mathbf{e_\delta}\| \le s\sqrt{m(k+2)}$ and $\|\mathbf{e}\| \le s\sqrt{m(k+1)}$, the process moves to the next step, and returns 0 otherwise. It computes $\mathbf{u} = H(\mathbf{m})$, $\theta = h(\mathbf{A_B})$, and matrices $\mathbf{B}_1^{(u_1)} \| \cdots \| \mathbf{B}_k^{(u_k)}$, $\mathbf{C} = \mathbf{c}_1^{(\theta_1)} \| \cdots \| \mathbf{c}_m^{(\theta_m)}$. Then if the two equalities $(\mathbf{A}\|\mathbf{C}\|\mathbf{B}) \cdot \mathbf{e}_\delta = 0 \bmod q$, $(\mathbf{A_B}\|\mathbf{B}) \cdot \mathbf{e} = 0 \bmod q$ hold, the process returns 1 to accept the proxy signature, otherwise it returns 0.

One can easily check that a signature generated according to *ProxySign*(·) will be accepted by *Verify*(·) in the sense that the latter returns 1. Consequently, the above scheme is correct.

4. Security Proofs

Previous proxy signature schemes were built in the uniform random oracle model, which is an idealized cryptographic setting that permits all parties to access a uniform random function. The protocols in this model are secured with a cryptographic hash function that is efficient and computable and replaces the uniform random function. However, this process is not practical. Thus, the security of the new scheme in the standard model needs to be proved under EU-SCMA.

4.1 Adversary types

In practical scenarios there are three types of adversaries against a proxy signature scheme.

(1) The first type of adversary only obtains the public keys of the original and proxy signers.

(2) The second type of adversary holds the public keys of the original and proxy signers and knows the proxy signer's private key.

(3) The third type of adversary possesses the public keys of the original and proxy signers and the private key of the original signer.

As the first type of adversary is less dangerous than the second and third types, it can be proved that the scheme is secure under the attack of the last two adversaries.

4.2 Security against second type of adversary

The second type of adversary knows the private key of the proxy signer. Thus, we have the following theorem.

Theorem 2: When the second type of adversary F is a PPT EU-SCMA of the proxy signature scheme PS, F makes Q queries with advantage $Adv_{PS}^{EU-SCMA}(F)$. Then, a PPT algorithm S uses F as a subroutine whose advantage for solving $\mathrm{SIS}_{q,\beta}$ is

$$Adv_{SIS_{q,\beta}}(S^F) \geq Adv_{PS}^{EU\text{-}SCMA}(F) / ((k-1) \cdot Q + 1) - negl(n), \tag{9}$$

where $\beta = \sigma\sqrt{(k+2)m}$.

Proof. By assuming that F is an adversary that tries an EU-SCMA on PS, a reduction S is constructed against the attack $\mathrm{SIS}_{q,\beta}$. S accepts the input $m' = (2k + 3)m$ (uniformly random) and independent samples from $\mathbb{Z}_q^n$ of a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m'}$, parsing $\mathbf{A}$ as

$$\mathbf{D} = \mathbf{A} \,||\, \mathbf{C}_1^{(0)} \,||\, \mathbf{C}_1^{(1)} \,||\cdots||\, \mathbf{C}_m^{(0)} \,||\, \mathbf{C}_m^{(1)} \,||\, \mathbf{U}_1^{(0)} \,||\, \mathbf{U}_1^{(1)} \,||\cdots||\, \mathbf{U}_k^{(0)} \,||\, \mathbf{U}_k^{(1)}, \tag{10}$$

where $\mathbf{C}_i^{(b)} \in \mathbb{Z}_q^n$ and $\mathbf{A}, \mathbf{U}_j^{(b)} \in \mathbb{Z}_q^{n\times m}$ for $i \in [m], j \in [k], b \in \{0,1\}$.

The reduction S simulates the static chosen-message attack F as follows.

(1) Query

F sends Q messages $\mathbf{u}^1, ..., \mathbf{u}^Q \in \{0,1\}^k$ to S (without loss of generality with the assumption of $\mathbf{u}^i = H(\mathbf{m}^i)$ for simplicity).

(2) System Generation

a. **S** generates the SNP set P for $\mathbf{u}^1, ..., \mathbf{u}^Q$. According to the conclusion in Theorem 1, the size of the set is upper-bounded by $\|P\| \le (k-1)\cdot Q + 1$.

b. **S** runs the algorithm *GenBasis*$(1^n,1^m,q)$ to obtain $\mathbf{A_B}$, $\mathbf{S_B}$ and treats them as the key pair of the proxy signer. Then, **S** extracts **A** from **D** and lets **A** act as the public key of the original signer.

c. **S** extracts $\mathbf{C}_i^{(b)}$ when $i \in [k]$, $b \in \{0,1\}$. It sets the matrices for the system parameters.

d. **S** chooses a uniformly random element $\mathbf{p} \in P$ with $t = |\mathbf{p}|$. For $i \in [t]$, **S** sets $\mathbf{B}_i^{\mathbf{p}_i} = \mathbf{U}_i^{(0)}$ and runs the algorithm *GenBasis*$(1^n,1^m,q)$ to obtain the matrix $\mathbf{B}_i^{1-\mathbf{p}_i}$ and the corresponding short basis of $\Lambda^{\perp}(\mathbf{B}_i^{1-\mathbf{p}_i})$. For $j \in \{t+1, ..., k\}$, $b \in \{0,1\}$, we have $\mathbf{B}_j^{(b)} = \mathbf{U}_j^{(b)}$.

S sends the following to the adversary F: a system parameter $\{\mathbf{C}_i^{(b)}\}_{i=1}^{m}$, $\{\mathbf{B}_j^{(b)}\}_{j=1}^{k}$, the key pair of the proxy signer $(\mathbf{A_B}, \mathbf{S_B})$, and the public key of the original signer.

(3) Query Answer

a. For each message **u** designed in the query step (the symbol i of $\mathbf{u}^i$ is omitted for simplicity), S generates the matrix $\mathbf{B} = \mathbf{B}_1^{(\mathbf{u}_1)} \| \cdots \| \mathbf{B}_k^{(\mathbf{u}_k)}$ for each bit of **u**. In the same way, it builds the matrix $\mathbf{C} = \mathbf{C}_1^{(\theta_1)} \| \cdots \| \mathbf{C}_m^{(\theta_m)}$ for each bit of $\theta = h(\mathbf{A_B})$. Since **p** is not the prefix of **u** queried in the previous step, there exists at least one bit (the ith bit) of **u** that is different from **p**. As a result, $\mathbf{B}_i^{1-\mathbf{p}_i}$ is chosen in the signing process and the short basis of $\Lambda^{\perp}(\mathbf{B}_i^{1-\mathbf{p}_i})$ becomes **S**. Using the "undirected growth" property of bonsai trees, the short basis of $\Lambda^{\perp}(\mathbf{A}\|\mathbf{C}\|\mathbf{B})$ can be computed. Thus, **S** obtains a short $\mathbf{e}_a$ with $(\mathbf{A}\|\mathbf{C}\|\mathbf{B}) \cdot \mathbf{e}_\delta = 0 \bmod q$.

b. For similar reasons, **S** obtains a short **e** with $(\mathbf{A_B}\|\mathbf{B}) \cdot \mathbf{e} = 0 \bmod q$ because the short basis of $\Lambda^{\perp}(\mathbf{A_B})$ is already known.

c. **S** sends all of $\{u, e_\delta, e\}$ to F.

(4) Forgery

a. After F verifies the validity of signatures, it produces and sends a forgery $(\mathbf{u}^*,\mathbf{e}_\delta^*,\mathbf{e}^*)$ to **S**. F can always forge $\mathbf{e}^*$ as it holds the private key of the proxy signer. Furthermore, $\| \mathbf{e}_\delta^* \| \le \sigma\sqrt{(k+2)m}$, $\| \mathbf{e}^* \| \le \sigma\sqrt{(k+1)m}$.

b. If **p** is not a prefix $\mathbf{u}^*$, the process stops. Otherwise, the following is obtained:

$$(\mathbf{A} \| \mathbf{C}_1^{(\theta_1)} \| \cdots \| \mathbf{C}_m^{(\theta_m)} \| \mathbf{B}_1^{(\mathbf{u}_1^*)} \| \cdots \| \mathbf{B}_k^{(\mathbf{u}_k^*)}) \cdot \mathbf{e}_\delta^* = 0 \bmod q \tag{11}$$

or

$$(\mathbf{A} \| \mathbf{C}_1^{(\theta_1)} \| \cdots \| \mathbf{C}_m^{(\theta_m)} \| \mathbf{U}_1^{(0)} \| \mathbf{U}_2^{(0)} \| \cdots \| \mathbf{U}_t^{(0)} \| \cdots \| \mathbf{U}_k^{(\mathbf{u}_k^*)}) \cdot \mathbf{e}_\delta^* = 0 \bmod q. \tag{12}$$

Thus, by inserting zeros into $\mathbf{e}^{\delta^*}$, **S** generates nonzero $\mathbf{v} \in \mathbb{Z}^{m'}$, so $\mathbf{Dv} = 0 \bmod q$. As $\|\mathbf{v}\| = \|\mathbf{e}^{\delta^*}\| \le \sigma\sqrt{(k+2)m}$, this is a solution for $\mathrm{SIS}_{q,\beta}$.

According to Definition 2, there is only one element as the prefix for u^*. Since $\|P\| \le (k-1)\cdot Q + 1$ and p is a uniformly random element in P, the probability that p is the prefix of u^* is at least $1/((k-1)\cdot Q + 1)$. If F wins the EU-SCMA game with a probability of $Adv_{PS}^{EU-SCMA}(F) - negl(n)$, **S** solves $\mathrm{SIS}_{q,\beta}$ with a probability of

$$\begin{aligned} Adv_{SIS_{q,\beta}}(S^F) &\geq \left[Adv_{PS}^{EU\text{-}SCMA}(F) - negl(n) \right] \cdot \left[1/((k-1)\cdot Q+1) \right] \\ &= Adv_{PS}^{EU\text{-}SCMA}(F) / \left(\left(k-1 \right)\cdot Q+1 \right) - negl(n). \end{aligned} \tag{13}$$

Consequently, if a PPT adversary *F* attacks PS with non-negligible advantage, the SIS problem can be solved with a non-negligible probability within a polynomial time. Thus, according to the hardness assumption of the SIS problem, no such adversary F exists.

4.2 Security against third type of adversary

The third type of adversary possesses the private key of the original signer. For such a scenario, we have the following theorem.

Theorem 3: When *F* is a PPT EU-SCMA adversary of our proxy signature scheme PS that creates *Q* queries with advantage $Adv_{PS}^{EU-SCMA}(F)$, a PPT algorithm using *F* is possible as a subroutine, whose advantage for solving is

$$Adv_{SIS_{q,\beta}}(S^F) \geq Adv_{PS}^{EU\text{-}SCMA}(F) / ((k-1)\cdot Q+1) - negl(n), \tag{14}$$

where $\beta = \sigma\sqrt{(k+1)m}$.

Proving Theorem 3 is similar to proving Theorem 2. The difference is that the original signer does not know the secret key of the proxy signer (which is $\mathbf{S_B}$ in the description of the scheme), so that it is intractable for an adversary to generate the second part of a signature under the SIS assumption.

5. Conclusion

With the further development and increasing use of the IoT, the security of the underlying sensor networks is of significant concern. To grant IoT applications the ability to securely recognize their underlying nodes in sensor networks even of quantum computing, we proposed a lattice-based proxy signature scheme. The proxy signature was proved to be existentially unforgeable against static chosen-message attack based on the standard model. In this scheme, an IoT application uses its key pair to assign its signing rights to the underlying nodes and later checks the signatures generated by the nodes. In all steps, the IoT application does not need to restore any public key of the underlying nodes, which reduces the storage cost required in a system of traditional digital signatures. The new proxy signature is appropriate for sensor networks and IoT applications. The results of this study will help enhance the performance of the current scheme and find other solutions to authentication in IoT systems.

Acknowledgments

This work was supported by Guangdong Major Project of Basic and Applied Research (No. 2019B030302008), Guangdong Basic and Applied Basic Research Foundation (No. 2020A1515010687), National Nature Science Foundation of China (No. 61972431), and the Fundamental Research Funds for the Central Universities (No. 19lgpy217).

References

1 M. Mambo, K. Usuda, and E. Okamoto: Proc. 3rd ACM Conf. Computer and Communications Security (CCS, 1996) 48–57. https://doi.org/10.1145/238168.238185
2 X. Huang, Y. Mu, W. Susilo, F. Zhang, and X. Chen: Proc. Int. Conf. Embedded and Ubiquitous Computing (EUC, 2005) 480–489. https://doi.org/10.1007/11596042_50
3 X. Huang, W. Susilo, Y. Mu, and W. Wu: Proc. Int. Conf. Mobile Ad-hoc and Sensor Networks (MSN, 2006). 473–484. https://doi.org/10.1007/11943952_40
4 Z. Shao: Inf. Process. Lett. **85** (2003) 137. https://doi.org/10.1016/S0020-0190(02)00367-8
5 K. Zhang: Proc. Int. Workshop Information Security (ISW, 1997) 282–290. https://doi.org/10.1007/BFb0030429
6 M. Ajtai: Proc. 28th Annu. ACM Symp. Theory of Computing (STOC, 1996) 99–108. https://doi.org/10.1145/237814.237838
7 A. Kawachi, K. Tanaka, and K. Xagawa: Proc. Int. Workshop Public Key Cryptography (PKC, 2007) 315–329. https://doi.org/10.1007/978-3-540-71677-8_21
8 O. Regev: Proc. 37th Annu. ACM Symp. Theory of Computing (STOC, 2005) 84–93. https://doi.org/10.1145/1060590.1060603
9 O. Goldreich, S. Goldwasser, and S. Halevi: Proc. Annu. Int. Cryptology Conf. (CRYPTO, 1997) 112–131. https://doi.org/10.1007/BFb0052231
10 C. Gentry, C. Peikert, and V. Vaikuntanathan: Proc. 40th Annu. ACM Symp. Theory of Computing (STOC, 2008) 197–206. https://doi.org/10.1145/1374376.1374407
11 D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert: Proc. Annu. Int. Conf. Theory and Application of Cryptographic Techniques (Eurocrypt, 2010) 523–552. https://doi.org/10.1007/978-3-642-13190-5_27
12 X. Boyen: Proc. Int. Workshop Public Key Cryptography (PKC, 2010) 499–517. https://doi.org/10.1007/978-3-642-13013-7_29
13 M. Rückert: Proc. Int. Workshop Post-quantum Cryptography (PQCrypto, 2010) 182–200. https://doi.org/10.1007/978-3-642-12929-2_14
14 Y. Jiang, F. Kong, and X. Ju: Proc. Int. Conf. Computational Intelligence and Security (CIS, 2010) 382–385. https://doi.org/10.1109/CIS.2010.88
15 D. Micciancio and O. Regev: Proc. 45th Annu. IEEE Symp. Foundations of Computer Science (FOCS, 2004) 372–381. https://doi.org/10.1109/FOCS.2004.72
16 C. Peikert and A. Rosen: Proc. Theory of Cryptography Conf. (TCC, 2006) 145–166. https://doi.org/10.1007/11681878_8
17 C. Peikert: Proc. Annu. Int. Cryptology Conf. (CRYPTO, 2010) 80–97. https://doi.org/10.1007/978-3-642-14623-7_5
18 J. Alwen and C. Peikert: Theory Comput. Syst. **48** (2011) 535. https://doi.org/10.1007/s00224-010-9278-3