Article

Lattice-Based Revocable Certificateless Signature

Ying-Hao Hung, Yuh-Min Tseng * and Sen-Shan Huang

Department of Mathematics, National Changhua University of Education, Jin-De Campus, Chang-Hua 500, Taiwan; hungyinghao@gmail.com (Y.-H.H.); shuang@cc.ncue.edu.tw (S.-S.H.)

* Correspondence: ymtseng@cc.ncue.edu.tw; Tel.: +886-4-723-2105 (ext. 3216)

Received: 27 September 2017; Accepted: 18 October 2017; Published: 20 October 2017

Abstract: Certificateless signatures (CLS) are noticeable because they may resolve the key escrow problem in ID-based signatures and break away the management problem regarding certificate in conventional signatures. However, the security of the mostly previous CLS schemes relies on the difficulty of solving discrete logarithm or large integer factorization problems. These two problems would be solved by quantum computers in the future so that the signature schemes based on them will also become insecure. For post-quantum cryptography, lattice-based cryptography is significant due to its efficiency and security. However, no study on addressing the revocation problem in the existing lattice-based CLS schemes is presented. In this paper, we focus on the revocation issue and present the first revocable CLS (RCLS) scheme over lattices. Based on the short integer solution (SIS) assumption over lattices, the proposed lattice-based RCLS scheme is shown to be existential unforgeability against adaptive chosen message attacks. By performance analysis and comparisons, the proposed lattice-based RCLS scheme is better than the previously proposed lattice-based CLS scheme, in terms of private key size, signature length and the revocation mechanism.

Keywords: cryptography; lattice; certificateless signature; short integer solution (SIS); assumption; post-quantum cryptography

1. Introduction

Identity (ID)-based public-key cryptography (ID-PKC) was introduced by Shamir [1] to break away the requirement of certificates in conventional public-key cryptography (PKC). In ID-PKC, the public key of a user is decided by his/her associated identity information, such as e-mail address, telephone number, social security number, and so on. With the public key of a user, a trusted third party (called private key generator (PKG) produces the user’s associated private key and sends it to the user via a secure channel. Thus, the legitimacy of public keys can be verified publicly. Boneh and Franklin [2] employed Shamir’s concept to construct a workable ID-based encryption (IBE) scheme using bilinear maps such as Ate, Tate, and Weil pairings. Since the PKG knows all the users’ private keys, the PKG may impersonate all the users to forge a signature on any message and encrypt any ciphertext. In such a case, all ID-based cryptographic schemes have the key escrow problem.

In 1993, certificateless public-key cryptography (CL-PKC) was introduced by Al-Riyami and Paterson [3] to simultaneously repeal the use of certificates in conventional PKC and resolve the key escrow problem in ID-PKC. They concretely presented a certificateless signature (CLS) and a certificateless public-key encryption (CL-PKE) scheme. In CL-PKC, the private key of a user includes two parts, namely, a secret value and a partial private key. The secret value is randomly selected by the user while the partial private key is generated with her/his identity by a key generation center (KGC). Hence, the KGC does not know a user’s private key so that the key escrow problem occurred in ID-PKC is avoided. In addition, the user independently generates and publishes the public key, so the need of certificates in conventional PKC is abolished. Afterwards, numerous works [4–9] have addressed the CL-PKC area.
That are several cases that request a user’s public key to be invalidated before its preplanned expiration time. Thus, a public-key setting should offer a revocation method to cancel compromised or illegal users from the system. Tseng and Tsai [10] presented the revocation method using public channel. In addition, two primitives (encryption and signature) in revocable certificateless public-key cryptography (RCL-PKC) were also proposed, such as revocable certificateless public-key encryption (RCL-PKE) schemes [11,12] and revocable certificateless signature (RCLS) scheme [13,14]. Furthermore, Hung et al. [15] presented a short RCLS scheme.

Indeed, the security of these conventional PKC, ID-PKC, CL-PKC and RCL-PKC mentioned above rely on the difficulty of solving the discrete logarithm or integer factorization problems. However, when quantum computers come into reality, both hard problems would become easy to compute [16] so that those cryptographic schemes based on them would become insecure. Whereas, several new mathematical methods for cryptography have been constructed to resist quantum attacks. For post-quantum cryptography, lattice-based cryptography is significant [17] because no efficient quantum algorithm can solve the related problems that include the short integer solution (SIS) and short independent vector problem (SIVP) problems over lattices. Moreover, lattice-based cryptography is more efficient than other post-quantum cryptographies.

1.1. Related Work

For lattice-based cryptography, Goldreicha et al. [18] proposed lattice-based signature and public-key encryption schemes under the conventional PKC settings. Unfortunately, its signature scheme was completely broken in [19]. Afterward, several famous signature schemes were presented, including Gentry et al.’s scheme [20] and Lyubashevsky’s schemes [21,22]. The former is provably secure. In their scheme, Gentry et al. employed the Gaussian sampling and the hash-and-sign techniques, respectively, to produce users’ private keys and signatures. However, the private key is lengthy while the hash-and-sign technique is inefficient. Lyubashevsky [21] employed the Fiat-Shamir transformation technique to propose an efficient lattice-based signature scheme while its security is based on the short integer solution problem (SIS) over lattices. The Fiat-Shamir transformation turns out to be more efficient than the hash-and-sign technique when generating a signature. Moreover, to improve the efficiency further, Lyubashevsky [22] proposed another lattice-based signature scheme, which employed the rejection sampling technique to produce the signature. Lyubashevsky’s second scheme is simple and needs just a few matrix-vector multiplications and rejection samplings.

To combine the advantages of ID-PKC and lattice-based cryptography, Ruckert [23] presented two ID-based signature (IBS) schemes over lattice assumptions. One was shown to be secure in the standard model and the other is secure in the random oracle model. The framework of Ruckert’s scheme followed Gentry et al.’s scheme [20]. Therefore, the private key and the signature remain lengthy. Then, several lattice-based IBS schemes [24–26] were presented to enhance the security and efficiency. Recently, Xiang [27] adopted the binary tree structure used in [28] to construct a revocable IBS (RIBS) scheme over lattices. To improve the efficiency, Hung et al. [29] furthermore presented a new lattice-based RIBS. Their scheme adopted the NTRU lattice in [26] to produce the private key of a user. Therefore, the private key size and signature size are shorter than those of Xiang’s scheme.

In the past, the study of the lattice-based certificateless signature (CLS) schemes received little attention. Tian and Huang [30] proposed the first lattice-based CLS scheme. Since they adopted the GPV lattice in [20] to generate the private key of a user, it is of the form \( \left( S_1, S_2 \right) \), where \( S_1 \) is an \( m_1 \times k \) matrix and \( S_2 \) is an \( m_2 \times k \) matrix, with \( m_1, m_2 > 5 \log_2 q \) and \( q \) being a prime. However, the private key above turns out to be lengthy, so is the associated signature. Therefore, their scheme is inefficient and impractical. Moreover, no study on addressing the revocation problem in the existing lattice-based CLS schemes is presented.
1.2. Contribution and Organization

In this paper, we focus on the revocation issue and present the first revocable CLS (RCLS) scheme over lattices while improving the performance of Tian and Huang’s CLS scheme [30] mentioned above. Our RCLS scheme provides a revocation method using public channel to cancel compromised or illegal users. The revocation method follows the revocation concept of our previous literature [10]. In our RCLS scheme, a user’s private key consists of three parts that include a secret value, a time update key and a partial private key. The secret value is randomly selected by the user while the partial private key is generated with her/his identity by a key generation center (KGC). The point is that the time update key is changed along with time period and the KGC periodically sends new time update keys to non-revoked users via a public channel. If the KGC would like to cancel compromised or illegal users, the KGC just stops generating the new time update keys of these users. In our RCLS scheme, the partial private key is generated by using the key extract algorithm of Ducas et al.’s ID-based encryption over lattices [26]. In the key extract phase, Ducas et al. adopted a particular sampling algorithm to improve Gentry et al.’s key extract algorithm [20] by producing short trapdoor (private key). Meanwhile, in our signing phase, we adopt the rejection sampling technique in [22] to produce a signature. Therefore, our lattice-based RCLS scheme has shorter private key size and signature length than others. Relied on the difficulty of solving the short integer solution (SIS) problem [31], we show that the proposed lattice-based RCLS scheme offers existential unforgeability against adaptive chosen-message attacks for three adversaries that include Type I adversary (outsider), Type II adversary (honest-but-curious KGC) and Type III adversary (revoked user). When compared with the previously proposed lattice-based CLS scheme, the proposed lattice-based RCLS scheme possesses better security and similar efficiency.

The rest of the paper is arranged as follows. In Section 2, preliminaries are presented. The framework and security model of RCLS schemes are given in Section 3. The proposed lattice-based RCLS scheme is presented in Section 4. In Section 5, the security analysis of our scheme is demonstrated. Comparisons are presented in Section 6. Conclusions are drawn in Section 7.

2. Preliminaries

2.1. Notations

Throughout this paper, we denote several parameters as follows:
- $N$: a power-of-two integer.
- $\mathbb{R}$: the set of real numbers.
- $\mathbb{Z}$: the set of integers.
- $\mathbb{Z}_q$ for $q > 0$: the interval be the set of integers with $[-q/2, q/2]$.
- $R_q = \mathbb{Z}_q[X]/(X^N + 1)$: a ring of polynomials modulo $X^N + 1$ with coefficients in $\mathbb{Z}_q$.

For a vector $x$ and a matrix $X$, $\|x\| = \sqrt{\sum x_i^2}$ and $\|X\|_\infty = \max_i \|x_i\|$, respectively, denote the Euclidean norm of $x$ and the longest norm of all columns of $X$. Let $f = \sum_{i=0}^{N-1} f_i x^i$ and $g = \sum_{i=0}^{N-1} g_i x^i$ be two polynomials in $R_q$.

For a set $S$, the notation $y \leftarrow S$ denotes that $y$ is uniformly selected at random from $S$. For a distribution $D$, $z \leftarrow D$ means that $z$ is selected according to the distribution $D$.

2.2. Anticirculant Matrices

Anticirculant matrices have a special structure and useful properties. An $N$-dimensional anticirculant matrix $C_N(f)$ is defined as follows.

**Definition 1.** $C_N(f)$ is a Toeplitz matrix represented by
\[
C_N(f) = \begin{bmatrix}
(f)

(x \cdot f)

\vdots

(x^{N-1} \cdot f)
\end{bmatrix} = \begin{bmatrix}
f_0 & f_1 & \cdots & f_{N-2} & f_{N-1} \\
-f_{N-1} & f_0 & \cdots & f_{N-3} & f_{N-2} \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
-f_1 & -f_2 & \cdots & -f_{N-1} & f_0
\end{bmatrix},
\]

where \( f = \sum_{i=0}^{N-1} f_i x_i \in R_q \).

For convenience, \( C_N(f) \) is abbreviated as \( C(f) \) in the sequel. Anticirculant matrices have the following nice property.

**Lemma 1.** If \( f, g \in R_q \), we have \( C(f) \ast C(g) = C(f \ast g) \) and \( C(f) + C(g) = C(f + g) \) \([26]\).

### 2.3. Lattice and NTRU Lattice

Here, we briefly define a lattice and an NTRU lattice. A lattice is a full-rank discrete subgroup of \( \mathbb{R}^n \). And an NTRU lattice comes from a particular class of convolution modular lattices. The detailed definitions are given below.

**Definition 2.** Let \( n \) vectors \( v_1, v_2, \ldots, v_n \) be linearly independent and \( B = \{v_1, \ldots, v_n\} \) be the basis of the \( n \)-dimensional lattice \( \Lambda \). The lattice \( \Lambda \) produced by the basis \( B \) is presented as

\[
\Lambda = \{L(v_1, \ldots, v_n) = \{\sum_{i=1}^n x_i v_i : x_i \in \mathbb{R}^n\} : L = \text{linear function}\}.
\]

**Definition 3.** Let \( h = g \ast f^{-1} \), where \( f, g \in R_q \). The NTRU lattice \( \Lambda_{h, q} \) associated with \( h \) and a positive integer \( q \) is a full-rank lattice of \( \mathbb{Z}^{2N} \) and is represented as

\[
\Lambda_{h, q} = \{(u, v) \in R_q^2 \mid u + v \ast h = 0\}.
\]

Indeed, \( \Lambda_{h, q} \) is produced by the rows of

\[
\Lambda_{h, q} = \begin{bmatrix}
-C(h) & I_N \\
qI_N & O_N
\end{bmatrix},
\]

where \( I_N \) is the \( N \times N \) unit matrix, \( O_N \) and \( C(h) \), respectively, denote the \( N \times N \) null matrix and an \( N \)-dimensional anticirculant matrix with \( h \). If \( h \) is uniformly distributed in \( R_q \), the basis \( \Lambda_{h, q} \) is not suitable to solve the usual lattice problems. Hence Hoffstein et al. \([32]\) remedied this situation by constructing another appropriate basis for \( \Lambda_{h, q} \), namely,

\[
B_{f, g} = \begin{bmatrix}
C(g) & -C(f) \\
C(G) & -C(F)
\end{bmatrix},
\]

where \( f, G \in R_q \) such that \( f \ast G - g \ast F = q \).

Indeed, we can efficiently find \( F \) and \( G \). By the following lemma, \( B_{f, g} \) is called the short basis for \( \Lambda_{h, q} \) due to the fact \( \|B_{f, g}\| \leq \|\Lambda_{h, q}\| \).
Lemma 2. Let \( f, g \in R_q \) and \( h = g \ast f^{-1} \), and let \( F, G \in R_q \) satisfy the equality \( f \ast G - g \ast F = q \) [26]. Then, \( B_{f,g} \) generates the same NTRU lattice \( \Lambda_{h,q} \) as \( \Lambda_{h,q} \) does and \( \|B_{f,g}\| \leq \|A_{h,q}\| \).

Lemma 3. Given a prime \( q \), a power-of-two integer \( N \) and \( \sigma = 1.17/\sqrt{2N} \), there exists a probabilistic polynomial-time (PPT) algorithm \( \text{TrapGen}(q, N) \) that can produce a pair of polynomials \( f \) and \( g \) and then computes \( h = g \ast f^{-1} \), and outputs a trapdoor matrix \( B_{f,g} \) as a short basis of \( \Lambda_{h,q} \). Here, \( h \) is published publicly and is statistically close to be uniform in \( R_q \) [26].

2.4. Gaussian Distribution

Here, we present the definitions of the continuous and discrete Gaussian distributions, which are useful tools in lattice-based cryptography.

Definition 4. The continuous Gaussian distribution over \( \mathbb{R}^N \) with the center \( c \in \mathbb{R}^N \) and the standard deviation \( s > 0 \), is defined as
\[
\rho_c^N(x) = \left( \frac{1}{s \sqrt{2\pi}} \right)^N e^{-\frac{|x-c|^2}{2s^2}}, \text{ where } x \in \mathbb{R}^N.
\]

We scale this distribution for any lattice \( \Lambda \in \mathbb{R}^N \) by \( \rho_{c,\Lambda}^N = \sum_{x \in \Lambda} \rho_c^N(x) \) so as to make the distribution fitting and acquire a probability function.

Definition 5. The discrete Gaussian distribution over \( \mathbb{R}^N \) with the center \( c \in \mathbb{R}^N \) and the standard deviation \( s > 0 \), is defined as
\[
D_{c,s}^N = \frac{\rho_c^N(x)}{\rho_c^N(A)}, \text{ where } x \in \mathbb{R}^N.
\]

In this paper, \( \rho_c^N \) and \( D_c^N \) are abbreviated from \( \rho_{c,\Lambda}^N \) and \( D_{c,\Lambda}^N \) respectively for convenience. In the following lemma, Lyubashevsky [22] gave two properties of a discrete distribution \( D_{c,\sigma}^N(x) \) in dimension \( N \) with standard deviation \( \sigma \) at center \( c \).

Lemma 4. Let \( c \in \mathbb{Z}^N \).

1. If \( \sigma = \alpha \parallel \log N \), then \( \Pr[x \in D_{\sigma}^N \setminus D_{\sigma}^N(x) / D_{c,\sigma}^N(x) = O(1)] = 1 - 2^{-\alpha \log N} \).
2. If \( \sigma = \alpha \parallel \log N \) and \( \alpha > 0 \), then \( \Pr[x \in D_{\sigma}^N \setminus D_{\sigma}^N(x) / D_{c,\sigma}^N(x) < e^{12/\alpha + 1/(2\sigma^2)}] > 1 - 2^{-100} \).

2.5. Sampling Technique

According to [31], if one takes a so-called noise vector from a Gaussian distribution and adds this vector to a lattice, then one can obtain a distribution that is statistically close to uniform one. Based on this, Gentry et al. [20] presented a sampling algorithm and a trapdoor generation algorithm by using the Gaussian sampling technique over general lattices. To reduce the private key size, Ducas et al. [26] improved Gentry et al.’s scheme to propose a particular sampling algorithm over NTRU lattices that can produce short trapdoor by using a short basis \( B_{f,g} \) of \( \Lambda_{h,q} \) which is generated by \( \text{TrapGen} \) in the previous subsection. In our scheme, we will use Ducas et al.’s technique to produce the private key of a user by the short basis \( B_{f,g} \) without leaking any information of \( B_{f,g} \). Ducas et al.’s trapdoor generation algorithm has the following properties.
Lemma 5. Given a prime \( q \), an \( N \)-dimensional lattice \( \Lambda \), a short basis \( B_{f,s} \), if \( s \geq \| B_{f,s} \| \alpha (\sqrt{\log N}) \) and \( 0 < \varepsilon < 1 \), where \( B_{f,s} \) denotes \( B_{f,s} \)'s Gram-Schmidt orthogonalization, we have

\[
\Pr[\| x - c \| > \varepsilon \sqrt{N}] \leq \frac{1 + \varepsilon}{1 - \varepsilon} 2^{-N} \quad \text{for any } c \in \mathbb{R}^N \quad \text{and} \quad x \sim D_{\varepsilon}^N.
\]

and, there is an algorithm \( \text{SampleGau} (B_{f,s}, s, c) \) which produces a distribution statistically close to \( D_{\varepsilon}^N \) [26].

2.6. Rejection Sampling Algorithm

Lyubashevsky [22] proposed the rejection sampling technique to sign a message in lattice-based cryptography. This technique is simple and needs just a few matrix-vector multiplications and rejection samplings. Indeed, Lyubashevsky’s signing algorithm [22] is different from the one proposed by Micciancio and Peikert [33] even though both algorithms employ similar public keys. The main difference is that Lyubashevsky produces a signature by using the rejection sampling instead of the hash-and-sign technique. Moreover, the sizes of both signature and private key in Lyubashevsky’s scheme are smaller than those in Micciancio and Peikert’s scheme under the same security level. Here, we explain the workings of Lyubashevsky’s rejection sampling technique. A signer first selects a private key \( S \) which is an \( m \times k \) matrix of random integers of absolute value at most \( d \). And then the signer chooses an \( n \times m \) matrix \( A \) of random integers in \( q\mathbb{Z} \) and computes the other matrix \( T = AS \). The signer’s associated public key consists of \( A \) and \( T \). In addition, a cryptographic hash function \( H : \{0, 1\}^* \rightarrow \{v : v \in [-1,0,1]^k, \|v\| \leq \lambda\} \) is selected, where \( \lambda \) is constant.

In the sign procedure, the signer takes her/his private key \( S \), public key \( A \) and a message \( \mu \) as input, and returns a signature \((z, c)\). Upon receiving a signature \((z, c)\), a verifier validates the signature using the public keys \( A \) and \( T \). The setup, sign and verify procedures of the rejection sampling technique are presented in the following Algorithm 1.

### Algorithm 1: Rejection Sampling Technique

**Setup** \((n, \lambda, m, k)\):

\[
H : \{0, 1\}^* \rightarrow \{v : v \in [-1,0,1]^k, \|v\| \leq \lambda\}, \lambda \text{ is constant.}
\]

Private Key: \( S \leftarrow [-d, \ldots, 0, \ldots, d]^{mxk} \).

Verification Key: \( A \leftarrow q^m \mathbb{Z}, T = AS \).

**Sign** \((A, S, \mu)\):

1. \( y \leftarrow D_{\sigma}^n \).
2. \( c = H(A, y, \mu) \).
3. \( z = Sc + y \).
4. Output the pair \((z, c)\) with the probability \( \min \left[ \frac{D_{\sigma}^{n}(z)}{M \cdot D_{Sc, \sigma}^{n}(z)}, 1 \right] \), where \( M = O(1) \).

**Verify** \((A, T, z, c, \mu)\):

Accept it if both conditions \( \|z\| \leq 2\sigma \sqrt{m} \) and \( c = H(Az - Tc, \mu) \) hold.

Here, we present the main concept of the signing algorithm. The rejection sampling technique is to enable the distribution of the signature \((z, c)\) independent of the secret key \( S \). Thus, we would like to obtain a target distribution \( z \) from \( D_{\sigma}^{n} \), but \( z \) in the signing algorithm comes from the distribution \( D_{Sc, \sigma}^{n} \). For an appropriately-chosen value \( M \) and a standard deviation \( \sigma \), e.g. \( M = 2.72 \) and \( \sigma = 15,157 \) [22], the signing algorithm will output a valid signature satisfying both
$|z| \leq 2\sigma \sqrt{m}$ and $c = H(Az - Tc, \mu)$ with probability approximately $1/M$. And the distribution of $z$ is statistically close to the distribution chosen from $D^m$.

2.7. Hardness Assumptions

In this section, we present the short integer solution (SIS) problem over lattices as the security assumption. The difficulty of solving the SIS problem is equivalent to the difficulty of the worst case of solving the short independent vector problem (SIVP) with an approximation polynomial factor. The SIS problem and its assumption are defined as follows.

**Definition 6.** Let $q$ and $\beta$, respectively, be a positive integer and a real number, and $f_1, f_2, \ldots, f_m$ be polynomials chosen uniformly and independently from $\mathbb{R}_q$. The SIS$_{q,n,\beta}$ problem over lattices is to find $m$ non-zero integers $r_1, r_2, \ldots, r_m$ that satisfy two conditions $\sum_{i=1}^m r_if_i \equiv 0 \mod q$ and $\|r_1, r_2, \ldots, r_m\| \leq \beta$.

**Definition 7 (SIS assumption).** Given a real number $\beta$, a positive integer $q$, and $m$ polynomials $f_1, f_2, \ldots, f_m$ chosen uniformly and independently from $\mathbb{R}_q$, there exists no probabilistic polynomial-time adversary $A$ with non-negligible probability for solving the SIS problem. The successful probability (advantage) of the adversary $A$ is presented as

$$\text{Adv}_A = \text{Pr}\left[ A(<b, q, f_1, f_2, \ldots, f_m>) = (r_1, r_2, \ldots, r_m): \|r_1, r_2, \ldots, r_m\| \leq \beta \right].$$

As stated in Lemma 3, the distribution of $h = g/f$ is statistically close to the uniform distribution of $\mathbb{R}_q$. Hence, the SIS problem on NTRU lattice is to find a pair $(z_1, z_2)$ such that $z_1 + h * z_2 = 0$ and $\|z_1, z_2\| \leq \beta$.

3. Syntax and Security Model of RCLS

The framework of RCLS scheme is identical to that of the RCLS schemes in [14,15]. In an RCLS scheme, there are three roles, namely, a key generation center (KGC), signers and verifiers. An RCLS scheme consists of eight algorithms that are defined as follows.

**Definition 8.** An RCLS scheme contains eight algorithms:

- **Setup** ($N$): The algorithm is probabilistic and performed by an KGC. The algorithm takes as input a security parameter $N$, it returns the public parameters $\text{Parms}$ and a system secret key $S_{\text{KGC}}$. $S_{\text{KGC}}$ is kept secret by the KGC and $\text{Parms}$ are made public.

- **Partial private key extract** (ID): This deterministic algorithm is performed by the KGC. Upon receiving the identity ID of a user, the KGC produces the user’s partial private key $D_{\text{id}}$ and the first partial public key $P_{\text{id}}$ that are returned to the user.

- **Time key update** (ID, $t$): This deterministic algorithm is performed by the KGC. Upon receiving the identity ID of a user and a time period $t$, the KGC produces the time update key $T_{\text{id},t}$ of the user and returns it to the user.

- **Set secret value** (ID): This probabilistic algorithm is performed by a user with ID. The user randomly selects a secret value $S_{\text{id}}$, with which the user computes the second partial public key $R_{\text{id}}$.

- **Set private key** (D$_{\text{id}}$, $T_{\text{id},t}$, $S_{\text{id}}$): This deterministic algorithm is performed by a user with ID. The private key $S_{\text{K}} = (D_{\text{id}}, T_{\text{id},t}, S_{\text{id}})$ is set by the user.
- **Set public key** \((P_{id}, R_{id})\): This deterministic algorithm is performed by a user with ID. The public key \(PK_{id} = (P_{id}, R_{id})\) is set by the user, where \(P_{id}\) and \(R_{id}\) are the first partial and the second partial public keys respectively.

- **Sign** \((ID, SK_{id}, \mu, t)\): This probabilistic algorithm is performed by a user with ID. It takes as input the private key \(SK_{id}\) of the user, a message \(\mu\) and a time period \(t\), and returns a signature \(\zeta\) on \(\mu\).

- **Verify** \((ID, PK_{id}, \mu, \zeta, t)\): This deterministic algorithm is performed by a verifier (or receiver). It takes as input the public key \(PK_{id}\) of a user with ID, a message \(\mu\), a time period \(t\), and a signature \(\zeta\) and it returns “accept” if the signature \(\zeta\) is validated. Otherwise, it returns “reject”.

By the security model of RCLS schemes in [14,15], adversaries have three types that are presented as follows.

- **Type I adversary** (outsider): The adversary knows the time update key and the secret value of any entity, which are respectively obtained by listening the public channel and replacing the associated public key.

- **Type II adversary** (honest-but-curious KGC): The adversary may produce the partial private key and time update key of any entity, but it does not know the associated secret value.

- **Type III adversary** (revoked user): The adversary owns the partial private key and knows the associated secret value, but it does not get the current time update key.

**Definition 9.** We say that an RCLS scheme has existential unforgeability against adaptive chosen message attacks (RCLS-UF-ACMA) if a PPT adversary \(A\) with a non-negligible advantage wins the following RCLS-UF-ACMA game, which is cooperatively performed by \(A\) and a challenger \(C\).

- **Setup.** The setup algorithm is performed by the challenger \(C\) to produce public parameters \(Parms\) and the system secret key \(S_{KGC}\). \(S_{KGC}\) is kept secret for \(C\). It is worth mentioning, that if the adversary \(A\) is Type II, \(S_{KGC}\) is sent to \(A\). Note that for Type I and III adversaries, the KGC plays as the role of the challenger \(C\). For Type II adversary, the honest-but-curious KGC is the adversary \(A\).

- **Queries:** A may issue a number of different queries to \(C\) adaptively as follows. It is worth mentioning, that Type II adversary has the system secret key \(S_{KGC}\) so that it may compute the partial private key and time update key of any entity.
  
  - **Partial private key extract queries** \((ID)\). Upon receiving the identity \(ID\) of a user, \(C\) performs the partial private key extract algorithm to produce and return the user’s partial private key \(D_{id}\) to \(A\).
  
  - **Time key update queries** \((ID, t)\). Upon receiving the identity \(ID\) of a user and a time period \(t\), the \(C\) performs the time key update algorithm to produce and return the time update key \(T_{id,t}\) to \(A\).
  
  - **Secret value queries** \((ID)\). Given a user’s \(ID\), \(C\) performs the set secret value algorithm to produce and return the secret value \(S_{id}\) to \(A\).
  
  - **Public key queries** \((ID)\). Upon receiving the identity \(ID\) of a user, \(C\) returns \(PK_{id}\) to \(A\).
  
  - **Public key replacement queries** \((ID, PK_{id}')\). Upon receiving the identity \(ID\) of a user and a new public key \(PK_{id}'\), \(C\) records this replacement.
  
  - **Sign queries** \((ID, PK_{id}, \mu, t)\). Upon receiving \(ID\) and \(PK_{id}\) of a user, a message \(\mu\) and a time period \(t\), \(C\) plays the role of the signer and performs the sign algorithm to produce a valid signature \(\zeta\) on \(\mu\) and returns \(\zeta\) to \(A\).
- **Forgery**: Assume that the adversary A produces \((ID', \mu', \tau')\). It is worth mentioning, that \(ID'\) is the target identity. It is said that A wins the RCLS-UF-ACMA game when the following situations hold:

- \((ID', \mu', \tau')\) was never issued in the **sign queries**.
- The verify algorithm on \((ID', \mu', \tau')\) outputs “accept”.
- If A is of Type I adversary, the **partial private key extract queries** on \(ID'\) was never issued.
- If A is of Type II adversary, \(ID'\) was never issued in the **secret value** and **public key replacement queries**.
- If A is of Type III adversary, the **time key update queries** on \((ID', \tau')\) was never issued.

### 4. Concrete RCLS Scheme over Lattices

As defined in Definition 8 in Section 4, an RCLS scheme consists of eight algorithms. Here, we propose an efficient lattice-based RCLS scheme. Eight algorithms are presented as follows:

- **Setup**: Let \(s > 0, \sigma > 0, \text{ and } \lambda \) be a positive integer and \(N\) be a security parameter, the KGC chooses a prime \(q\). Then, the KGC runs \(\text{TrapGen}(q, N)\) of Lemma 3 in Section 2.3 to obtain \((f, g, h, g, h, G, L)\). Furthermore, the KGC sets the system secret key \(SK_{GC}\) as \(B\) and selects two system public keys \(a_1, a_2 \in Z^*_q\) and three hash functions \(h, H: \{0, 1\} \rightarrow Z_q^N\) and \(H: Z_q^N \times Z_q^N \times \{0, 1\} \rightarrow \{0, 1\}^n\), \(v\), \(v\) denotes the amount of nonzero elements of the vector \(v\). The public parameters are \(\text{Parms} = \langle \sigma, \alpha, \lambda, q, a_1, a_2, H_0, H_1, H_2 \rangle\).

- **Partial private key extract**: Upon receiving the identity \(ID(0, 1)^t\) of a user, the KGC produces the partial private key \((s_1, s_2)\) such that \(s_1 + h \cdot s_2 = P_{ID} \) and \(\|s_1, s_2\| < \sqrt{2N}\) by running \(\text{SampleGau}(B, s, (P_{ID}, 0))\) of Lemma 5 in Section 2.5, where \(P_{ID} = H_0(ID) \in Z_q^N\) is the first partial public key. The KGC returns the partial private key \(D_{ID} = (s_1, s_2)\) to the user securely. Note that Lyubashevsky et al. [36] have shown that if one knows \((h, P_{ID})\), recovering \((s_1, s_2)\) is still hard.

- **Time key update**: Upon receiving the identity \(ID\) of a non-revoked user and a time period \(t\), the KGC produces the time update key \((s_3, s_4)\) such that \(s_3 + h \cdot s_4 = T_{ID} \) and \(\|s_3, s_4\| < \sqrt{2N}\) by running \(\text{SampleGau}(B, s, (T_{ID}, 0))\) of Lemma 5 in Section 2.5, where \(T_{ID} = H_1(ID, t) \in Z_q^N\). The KGC then sends the time update key \(T_{ID,t} = (s_3, s_4)\) to the user by using a public channel.

- **Set secret value**: The user with \(ID\) randomly chooses a secret value \(S_{ID} = (s_5, s_6)\) uniformly from \([-d, \ldots, 0, \ldots, d]\), where \(1 \leq d \leq 31\). Meanwhile, the second partial public key is \(R_{ID} = a_1 \cdot s_5 + a_2 \cdot s_6\).

- **Set private key**: The user with \(ID\) may set the private key \(SK_{ID} = (D_{ID}, T_{ID,t}, S_{ID})\).

- **Set public key**: The user with \(ID\) may set the public key \(PK_{ID} = (P_{ID}, R_{ID})\).

- **Sign**: A signer with the private key \(SK_{ID}\) takes as input a message \(\mu \in \{0, 1\}^*\), the signer randomly and independently selects \(y_1, y_2, y_3, y_4, y_5, y_6\) by the distribution \(D_\mu\), and computes the following values:

\[
\begin{align*}
    c &= H_2(y_1 + h \cdot y_2, y_3 + h \cdot y_4, a_1 \cdot y_5 + a_2 \cdot y_6, \mu); \\
    z_1 &= y_1 + s_1 \cdot c; \quad z_2 = y_2 + s_2 \cdot c; \quad z_3 = y_3 + s_3 \cdot c; \\
    z_4 &= y_4 + s_4 \cdot c; \quad z_5 = y_5 + s_5 \cdot c; \quad z_6 = y_6 + s_6 \cdot c,
\end{align*}
\]
where \( \|z_1, z_2, z_3, z_4, z_5, z_6\| \leq 2\sigma\sqrt{6N} \). If no such \((z_1, z_2, z_3, z_4, z_5, z_6, c)\) is produced, repeat this algorithm. The above procedure is the rejection sampling technique. Finally, there exists a constant \( M = O(1) \) such that the user can produce a signature \((z_1, z_2, z_3, z_4, z_5, z_6, c)\) with probability \( \min \left( \frac{D^N_{\sigma}(z)}{MD^N_{\sigma}(z)} \right) \) which is similar to the ring variants of Lyubashevsky’s scheme [22], where

\[
\mathbf{z} = [z_1^T \| z_2^T \| z_3^T \| z_4^T \| z_5^T \| z_6^T]^T
\]

and

\[
\mathbf{v} = [(s_1 \ast c)^T \| (s_2 \ast c)^T \| (s_3 \ast c)^T \| (s_4 \ast c)^T \| (s_5 \ast c)^T \| (s_6 \ast c)^T]^T.
\]

**Verify:** Given a signature \((z_1, z_2, z_3, z_4, z_5, z_6, c)\) for a user’s ID on a message \(\mu\), a verifier needs to validate the signature by the equality

\[
\mathbf{c} = H_2(z_1 + h \ast z_2 - P_{ID} \ast c, z_3 + h \ast z_4 - T_{ID} \ast c, a_1 \ast z_5 + a_2 \ast z_6 - R_{ID} \ast c, \mu).
\]

The verify algorithm returns “accept” if the checking equality holds. Otherwise, it returns “reject.”

The correctness of the checking equality follows by

\[
(z_1 + h \ast z_2 - P_{ID} \ast c, z_3 + h \ast z_4 - T_{ID} \ast c, a_1 \ast z_5 + a_2 \ast z_6 - R_{ID} \ast c)
\]

\[
= (y_1 + s_1 \ast c + h \ast (y_2 + s_2 \ast c) - (s_1 + h \ast s_2) \ast c, y_3 + s_3 \ast c + h \ast (y_4 + s_4 \ast c) - (s_3 + h \ast s_4) \ast c,
\]

\[
a_1 \ast (y_5 + s_5 \ast c) + a_2 \ast (y_6 + s_6 \ast c) - (a_1 \ast s_5 + a_2 \ast s_6) \ast c
\]

\[
= (y_1 + h \ast y_2, y_3 + h \ast y_4, a_1 \ast y_5 + a_2 \ast y_6).
\]

### 5. Security Analysis

In the following, we demonstrate that our lattice-based RCLS scheme is secure against both Type I adversary (outsider) and Type III adversary (revoked user) in Theorem 1 while the security against Type II adversary (honest-but-curious KGC) is proven in Theorem 2. The proof technique of both theorems employs the rejection sampling technique in [22] and the Forking lemma in [37].

**Theorem 1.** Let three hash functions \( H_0 \), \( H_1 \) and \( H_2 \) be random oracles and \( N \) be the security parameter. Assume that a PPT adversary \( A \) (Types I and III) can break our lattice-based RCLS scheme with non-negligible probability \( \epsilon \). Thus, an algorithm \( C \) is constructed to resolve the SIS problem with non-negligible probability \( (1 - 2^{-\omega(N)}) \epsilon \).

**Proof.** Let \( q \) be a prime, \( N \) be a positive integer and \( \lambda, s, \sigma > 0 \). Let the algorithm \( C \) be a challenger who receives a random instance \((q, 2N, 2\delta s\sqrt{2N} + 4\sigma\sqrt{2N})\) of the SIS problem. In the following, we will show how the challenger \( C \) can compute a non-zero vector solution \((u_1, u_2) \in \mathbb{R}^2_\lambda\) of the SIS problem by using \( A \). Here, \( A \) (Type I or Type III adversary) interacts with \( C \) as defined in the RCLS-UF-ACMA game of Definition 9.

**Setup.** The challenger \( C \) randomly chooses polynomials \( a_i, a_2, h \in \mathbb{R}_s\) and controls the random oracles \( H_0, H_1 \) and \( H_2 \). The public parameters \( \text{Parms} = <N, s, \lambda, q, h, a_1, a_2, H_0, H_1, H_2> \) are sent to \( A \). Meanwhile, \( C \) maintains several initially empty lists \( L_0, L_1, L_2 \) and \( L_s \).

**Queries.** \( A \) can adaptively issue several queries to \( C \) as follows:

- **\( H_0 \) queries:** Let \( L_0 \) consist of tuples of the form \(<ID_1, D_{ID_1}, P_{ID_1}>\). Upon receiving a query
with \(ID_i\) from A, C produces a response to this query as follows.

1. Search \(ID_i\) in \(L_0\). If it is found, the same answer in \(L_0\) is returned to A because the query has been ever issued.

2. Otherwise, select \(s_{i1}, s_{i2} \in \mathbb{D}_s^N\) at random such that \(\|s_{i1}, s_{i2}\| < s\sqrt{2N}\) and compute the polynomial \(P_{ID_i} = s_{i1} + h \cdot s_{i2}\). Then \(P_{ID_i}\) is sent to A and \(<ID_i, D_{ID_i} = (s_{i1}, s_{i2}), P_{ID_i}>\) is added in the list \(L_0\).

1. \(H_1\) queries: Let \(L_1\) consist of tuples of the form \(<ID_i, t, T_i, T_{ID_i}>\). Upon receiving a query with \((ID_i, t)\) from A, C produces a response to this query as follows.
   1. Search \((ID_i, t)\) in \(L_1\). If it is found, the same answer in \(L_1\) is returned to A because the query has been ever issued.
   2. Otherwise, select \(s_{i3}, s_{i4} \in \mathbb{D}_s^N\) at random such that \(\|s_{i3}, s_{i4}\| < s\sqrt{2N}\) and compute the polynomial \(T_i = s_{i3} + h \cdot s_{i4}\). Then \(T_i\) is sent to A and \(<ID_i, t, T_i, T_{ID_i}>\) is added in the list \(L_1\).

2. \(H_2\) queries: Let \(L_2\) consist of tuples of the form \(<w_j, x_j, v_j, \mu_j, c_j>\). Upon receiving a query with \((w_j, v_j, x_j, \mu_j)\) from A, C produces a response to this query as follows.
   1. Search \((w_j, v_j, x_j, \mu_j)\) in \(L_2\). If it is found, the same answer in \(L_2\) is returned to A because the query has been ever issued.
   2. Otherwise, randomly select \(c_j \in Z_q^N\). Then \(c_j\) is sent to A and \(<w_j, x_j, v_j, m_j, c_j>\) is added in the list \(L_2\).

3. Partial private key queries: A issues this query along with \(ID_i\), C produces a response to this query as follows.
   1. Search \(ID_i\) in \(L_0\). If it is found, the same answer in \(L_0\) is returned to A because the query has been ever issued.
   2. Otherwise, issue the \(H_0\) query to obtain the tuple \(<ID_i, D_{ID_i}, P_{ID_i}>\). Then, return \(D_{ID_i}\) to A.

4. Time key update queries: A issues this query along with \((ID_i, t)\), C produces a response to this query as follows.
   1. Search \((ID_i, t)\) in \(L_1\). If it is found, the same answer in \(L_1\) is returned to A because the query has been ever issued.
   2. Otherwise, issue the \(H_1\) query to obtain the tuple \(<ID_i, t, T_i, T_{ID_i}>\). Then, return \(T_i\) to A.

5. Secret value queries: Let \(L_5\) consist of tuples of the form \(<ID_i, S_{ID_i}, R_{ID_i}>\). Upon receiving a query with \(ID_i\) from A, C produces a response to this query as follows.
   1. Search \(ID_i\) in \(L_5\). If it is found, the same answer in \(L_5\) is returned to A because the query has been ever issued.
   2. Otherwise, randomly select \(s_{i5}, s_{i6} \in \{-d, ..., 0, ..., d\}\), where \(1 \leq d \leq 31\), and compute the polynomial \(R_{ID_i} = a_1 \cdot s_{i5} + a_2 \cdot s_{i6}\). Then \(S_{ID_i} = (s_{i5}, s_{i6})\) is sent to A and \(<ID_i, S_{ID_i}, R_{ID_i}>\) is added in the list \(L_5\).

6. Public key queries: A issues this query along with \(ID_i\), C produces a response to this query
as follows.

1. Search \( ID_i \) in \( L_0 \) and \( L_s \). If it is found, which means that the query has been ever issued, then C returns A with the same answer \( PK_{ID_i} = (P_{ID_i}, R_{ID_i}) \), where \( P_{ID_i} \) and \( R_{ID_i} \) are taken from \( L_0 \) and \( L_s \), respectively.
2. Otherwise, issue the \( Ho \) query and Secret value query to obtain \( P_{ID_i} \) and \( R_{ID_i} \). Then \( PK_{ID_i} = (P_{ID_i}, R_{ID_i}) \) is sent to A.

- **Public key replacement queries**: A issues this query along with a new public key \( PK'_{ID_i} = (P'_{ID_i}, R'_{ID_i}) \) of \( ID_i \) to replace the old public key \( PK_{ID_i} = (P_{ID_i}, R_{ID_i}) \). C replaces the \( iID_{PA} \) in \( L_0 \) with \( iID'_{PA} \) and the \( iID_{RA} \) in \( L_s \) with \( iID'_{RA} \).

- **Sign queries**: Upon receiving a request from A along with a message \( \mu \) and a time period \( t \), \( (ID_i, PK_{ID_i}) \), where \( PK_{ID_i} = (P_{ID_i}, R_{ID_i}) \), the challenger C makes the following steps to produce a valid signature.

1. Search \( ID_i \) in \( L_0 \), \( L_1 \) and \( L_s \), respectively, to obtain \( iID_{PA}, iID_{RA}, T_{ID_i} \) and \( iID_{PA}, iID_{RA}, T_{ID_i} \) with \( \| z_i, z_2, z_3, z_4, z_5, z_6 \| \leq 2\sigma \sqrt{N} \). Then, compute \( w_j = z_i + h * z_2 - P_{ID_i} * c_j \), \( v_j = z_3 + h * z_4 - T_{ID_i} * c_j \) and \( x_j = a_1 * z_5 + a_2 * z_6 - R_{ID_i} * c_j \).
2. Add \( w_j, v_j, x_j, \mu, c_j \) in the list \( L_2 \) and send the signature \( (z_i, z_2, z_3, z_4, z_5, z_6, c_j) \) on \( \mu \) to A.

Note that the signature \( \zeta = (z_i, z_2, z_3, z_4, z_5, z_6, c_j) \) is valid because it may satisfy the following equality:

\[
\zeta_{ID_i} = H_2(z_i + h * z_2 - P_{ID_i} * c_j, z_3 + h * z_4 - T_{ID_i} * c_j, a_1 * z_5 + a_2 * z_6 - R_{ID_i} * c_j, \mu) = H_2(w_j, v_j, x_j, m).
\]

Therefore, when the adversary A issues the Sign query, the challenger C can output a valid signature even though C does not possess the valid secret key or time update key.

- **Forgery**: After making all the queries needed, the adversary A forges a signature tuple \( (z'_1, z'_2, z'_3, z'_4, z'_5, z'_6, c') \) on message \( \mu' \) for \( ID'_i \) at time period \( t' \).

When A successfully forges a valid signature \( (z'_1, z'_2, z'_3, z'_4, z'_5, z'_6, c') \), the challenger C uses the Forking lemma [37] and replays A with different hash value of \( H_2 \) queries to produce another valid signature \( (z'_1, z'_2, z'_3, z'_4, z'_5, z'_6, c') \) such that \( c' \neq c \) by the same random tape. Because \( (z'_1, z'_2, z'_3, z'_4, z'_5, z'_6, c') \) and \( (z'_1, z'_2, z'_3, z'_4, z'_5, z'_6, c') \) are two valid signatures on the message \( \mu' \) for \( (ID'_i, PK_{ID'_i}, \mu') \), we can obtain the equality

\[
H_2(z'_1 + h * z'_2 - P_{ID'_i} * c', z'_3 + h * z'_4 - T_{ID'_i} * c', a_1 * z'_5 + a_2 * z'_6 - R_{ID'_i} * c', \mu') = H_2(z'_1 + h * z'_2 - P_{ID'_i} * c', z'_3 + h * z'_4 - T_{ID'_i} * c', a_1 * z'_5 + a_2 * z'_6 - R_{ID'_i} * c', \mu'),
\]

which reduces to

\[
z'_1 + h * z'_2 - P_{ID'_i} * c' = z'_1 + h * z'_2 - P_{ID'_i} * c'.
\]

Since \( P_{ID'_i} = s_1 + h * s_2 \), we arrive at
\[ z_1^* + h \ast z_2^* - (s_1 + h \ast s_2) \ast c^* = z_1^* + h \ast z_2^* - (s_1 + h \ast s_2) \ast c^* \]
\[ z_1^* - z_1^* - s_1 (c^* - c') + h \ast (z_2^* - z_2^* - s_2 (c^* - c')) = 0 \]
\[ (1, h) \ast (z_1^* - z_1^* - s_1 (c^* - c'), z_2^* - z_2^* - s_2 (c^* - c')) = 0. \]

Then, the challenger C sets \( (u_1, u_2) = (z_1^* - z_1^* - s_1 (c^* - c'), z_2^* - z_2^* - s_2 (c^* - c')). \)

If \( \| (z_1^* - z_1^*, z_2^* - z_2^*) \| \leq 4\sqrt{2N} \) and \( \| (s_1, s_2) \| \leq s\sqrt{2N} \) with overwhelming probability, we can obtain \( \| (u_1, u_2) \| \leq 2s\sqrt{2N} + 4s\sqrt{2N} \). As stated in Lemma 3, the distribution of \( h = g/f \) is statistically close to the uniform distribution of \( R_q \) [35]. The SIS problem on NTRU lattice is to find a pair \((u_1, u_2) \in R_q^2\) such that \( u_1 + h \ast u_2 = 0 \) and \( \| (u_1, u_2) \| \leq \beta \), where \( \beta \) is \( 2s\sqrt{2N} + 4s\sqrt{2N} \). Since the adversary A does not know the system secret key \( B \) generated by \( g, f \in R_q \) and has generated such a pair \((u_1, u_2)\), we say that the adversary A solves the SIS problem. According to the same probability analysis in [22], if the adversary A can break our lattice-based RCLS scheme with non-negligible probability \( \epsilon \). Then, we can construct an algorithm C to solve the SIS problem with non-negligible probability \( 1 - 2^{-\omega(\log N)} \). □

**Theorem 2.** Let three hash functions \( H_0, H_1 \) and \( H_2 \) be random oracles and \( N \) be the security parameter. Assume that an PPT adversary A (Types II) can break our lattice-based RCLS scheme with non-negligible probability \( \epsilon \). Thus, an algorithm C is constructed to resolve the SIS problem with non-negligible probability \( 1 - 2^{-\omega(\log N)} \epsilon \).

**Proof.** Let \( q \) be a prime, \( N \) be a positive integer and \( \lambda, s, \sigma > 0 \). Let the algorithm C be a challenger who receives a random instance \((q, 2N, 2\lambda s\sqrt{2N} + 4\sigma s\sqrt{2N})\) of the SIS problem. In the following, we will show how C can compute a non-zero vector solution \((u_1, u_2)\) of the SIS problem by using A. Here, A (Type II adversary) interacts with the challenger C as defined in the RCLS-UF-ACMA game of Definition 9.

- **Setup.** The challenger C performs the Setup algorithm of our lattice-based RCLS scheme to set \( S_{KGC} = B \) and \( \text{Parms} = \langle N, s, \lambda, \sigma, h, a_1, a_2, H_0, H_1, H_2, 3 > \rangle \), where three hash functions \( H_0, H_1 \) and \( H_2 \) are random oracles. The system secret key and \( \text{Parms} \) are then sent to A. Having the system secret key \( S_{KGC} \), C can compute the partial private key \( D_{ID_t, t} \), time update key \( T_{ID_t} \), and partial public key \( P_{ID_t} \) of any user with \( ID_t \) without issuing the other queries. Meanwhile, C maintains several initially empty lists \( L_0, L_1, L_2 \) and \( L_3 \).

- **Queries.** A can adaptively issue several queries to C as follows:
  - \( H_0 \) queries: Let \( L_0 \) consist of tuples of the form \( < ID_t, D_{ID_t, t}, P_{ID_t} > \). Upon receiving a query with \( ID_t \) from A, C produces a response to this query as follows.
    1. Search \( ID_t \) in \( L_0 \). If it is found, the same answer in \( L_0 \) is returned to A because the query has been ever issued.
    2. Otherwise, randomly select a \( P_{ID_t} \in Z_q^N \) and run the algorithm \( \text{SampleGau}(B, s, (P_{ID_t}, 0)) \) to obtain \( s_{ID_t}, s_{ID_t} \in D_q^N \) such that \( \| (s_{ID_t}, s_{ID_t}) \| < s\sqrt{2N} \). Then \( P_{ID_t} \) is sent to A and \( < ID_t, D_{ID_t} = (s_{ID_t}, s_{ID_t}), P_{ID_t} > \) is added in the list \( L_0 \).
  - \( H_1 \) queries: Let \( L_1 \) consist of tuples of the form \( < ID_t, t, T_{ID_t}, T_{ID_t} > \). Upon receiving a query
with \((ID, t)\) from A, C produces a response to this query as follows.

1. Search \((ID, t)\) in \(L_1\). If it is found, the same answer in \(L_1\) is returned to A because the query has been ever issued.

2. Otherwise, select \(s_{i5}, s_{i4} \in D^N\) at random such that \(||s_{i5}, s_{i4}|| < s\sqrt{2N}\) and compute the polynomial \(T_{ij} = s_{i5} + h * s_{i4}\). Then \(T_{ij}\) is sent to A and \(<ID, t, T_{ij}, T_{ID, t}>\) is added in the list \(L_1\).

- \(H_2\) queries: Let \(L_2\) consist of tuples of the form \(<w_j, x_j, v_j, \mu_j, c_j>\). Upon receiving a query with \((w_j, v_j, x_j, \mu_j)\) from A, C produces a response to this query as follows.

1. Search \((w_j, v_j, x_j, \mu_j)\) in \(L_2\). If it is found, the same answer in \(L_2\) is returned to A because the query has been ever issued.

2. Otherwise, randomly select \(N_{ji} \in \mathbb{Z}_q\) such that \(N_{ji} < \mathbb{Z}_q\) and compute the polynomial \(12 = t_{ij} + t_{ji}\). Then \(t_{ij}\) is sent to A and \(<w_j, v_j, x_j, \mu_j, c_j>\) is added in the list \(L_2\).

- Secret value queries: Let \(L_S\) consist of tuples of the form \(<iID, S_{ID, iD}, R_{ID, iD}>\). Upon receiving a query with \(iID\) from A, C produces a response to this query as follows.

1. Search \(iID\) in \(L_S\). If it is found, the same answer in \(L_S\) is returned to A because the query has been ever issued.

2. Otherwise, randomly select \(d_j \in \{-d, 0, ..., d\}\) where \(1 \leq d \leq 31\), and compute the polynomial \(R_{ID, iD} = a_1 * s_{i5} + a_2 * s_{i6}\). Then \(R_{ID, iD}\) is sent to A and \(<ID, t, S_{ID, iD}, R_{ID, iD}>\) is added in the list \(L_S\).

- Public key queries: A issues this query along with \(ID, t\), C produces a response to this query as follows.

1. Search \(ID, t\) in \(L_0\) and \(L_S\). If it is found, which means that the query has been ever issued, then C returns A with the same answer \((P_{ID, iD}, R_{ID, iD})\), where \(P_{ID, iD}\) and \(R_{ID, iD}\) are taken from \(L_0\) and \(L_S\), respectively.

2. Otherwise, issue the \(H_0\) queries and Secret value queries to obtain \(P_{ID, iD}\) and \(R_{ID, iD}\). Then \((P_{ID, iD}, R_{ID, iD})\) is sent to A.

- Public key replacement queries: A issues this query along with a new public key \((P^*, ID, t)\) of \(ID\) to replace the old public key \((P_{ID, iD}, R_{ID, iD})\), C replaces the \(P_{ID, iD}\) in \(L_0\) with \(P^*_{ID, iD}\) and the \(R_{ID, iD}\) in \(L_S\) with \(R^*_{ID, iD}\).

- Sign queries: Upon receiving a query from A along with \((\mu_j, ID, iD, PK_{ID, j})\) at time period \(t\), where \(PK_{ID, j} = (P_{ID, iD}, R_{ID, iD})\), the challenger C makes the following steps to produce a valid signature.

1. Search \(ID, t\) in \(L_0\) and \(L_S\), respectively, to obtain \(<ID, t, T_{ij}, T_{ID, j}>\) and \(<ID, t, S_{ID, iD}, R_{ID, iD}>\).

2. Randomly choose \(c_j \in \{0, 1\}^\lambda\) with \(\|c_j\| \leq \lambda\) and \(Z_1, Z_2, Z_3, Z_4, Z_5, Z_6 \in D^N\) with \(\|Z_1, Z_2, Z_3, Z_4, Z_5, Z_6\| \leq 2\sqrt{N}\). Then, compute \(w_j = Z_1 + h * Z_2 - P_{ID, iD} * c_j\), \(v_j = Z_3 + h * Z_4 - T_{ij} * c_j\) and \(x_j = a_1 * Z_5 + a_2 * Z_6 - R_{ID, iD} * c_j\).

3. Add \(<w_j, v_j, x_j, \mu_j, c_j>\) in the list \(L_1\) and send the signature \((Z_1, Z_2, Z_3, Z_4, Z_5, Z_6, c_j)\) on
\[ \mu_i \rightarrow A. \]

Finally, as in the proof of Theorem 1, the signature \((z_1, z_2, z_3, z_4, z_5, z_6, c)\) is valid and can pass the verification.

**Forgery:** After making all the queries needed, the adversary A forges a valid signature tuple \((z_1, z_2, z_3, z_4, z_5, z_6, c)\) on message \(\mu^*\) for \(ID^*\) at time period \(t^*\).

When A successfully forges a valid signature \((z_1, z_2, z_3, z_4, z_5, z_6, c)\), the challenger C uses the Forking lemma [37] and replays A with different hash value of \(H_2\) queries to produce another valid signature \((z_1, z_2, z_3, z_4, z_5, z_6, c)\) such that \(c^* \neq c'\) by the same random type. Because \((z_1, z_2, z_3, z_4, z_5, z_6, c)\) and \((z_1, z_2, z_3, z_4, z_5, z_6, c')\) are two valid signatures for \((\mu^*, ID^*, PK_{ID^*})\), we can obtain the equation

\[
H_2(z_1 + h \ast z_2 - P_{ID^*} \ast c', z_3 + h \ast z_4 - T_{ID^*} \ast c', a_1 \ast z_5 + a_2 \ast z_6 - R_{ID^*} \ast c', \mu')
\]

which reduces to

\[
a_1 \ast z_5 + a_2 \ast z_6 - R_{ID^*} \ast c' = a_1 \ast z_5 + a_2 \ast z_6 - R_{ID^*} \ast c'.
\]

Since \(R_{ID^*} = a_1 \ast s_5 + a_2 \ast s_6\), we arrive at

\[
(\mu_1, \mu_2) = (z_3 - z_5 - s_3(c - c'), z_3 - z_5 - s_3(c - c')).
\]

If \(|z_3 - z_5| \leq 4\sigma\sqrt{2N}\) and \(|s_3, s_6| \leq 2d\lambda\sqrt{2N}\) with overwhelming probability, we can obtain \(|u_1, u_2| \leq 2d\lambda\sqrt{2N} + 4\sigma\sqrt{2N}\). As stated in Lemma 3, the distribution of \(h = g/f\) is statistically close to the uniform distribution of \(R_0\) [35]. The SIS problem on NTRU lattice is to find a pair \((u_1, u_2) \in R_0^2\) such that \(u_1 + h \ast u_2 = 0\) and \(|u_1, u_2| \leq \beta\), where \(\beta = 2d\lambda\sqrt{2N} + 4\sigma\sqrt{2N}\). Since the adversary A does not know the system secret key \(B\) generated by \(g, f \in R_0\) and has generated such a pair \((u_1, u_2)\), we say that the adversary A solves the SIS problem. According to the same probability analysis in [22], if the adversary A can break our lattice-based RCLS scheme with non-negligible probability \(\epsilon\). Then, we can construct an algorithm C to solve the SIS problem with non-negligible probability \((1 - 2^{-d+\log N})\epsilon\).

**6. Comparisons**

To show the advantages of the proposed RCLS scheme, we make the comparisons between the previous schemes and ours. For convenience, we define the following notations to count the computational costs.

- \(T_s\): The required time of performing a sampling operation \(D_{\sigma}\).
- \(T_m\): The required time of performing a multiplication operation.
- \(T_a\): The required time of performing an addition/subtraction operation.
Table 1 demonstrates comparisons between Tian and Huang’s CLS scheme ([30]) and the proposed RCLS scheme in terms of lattice type, public key setting, averting key escrow problem, private key size, signature length, the computation costs of signing and verifying processes. Tian and Huang’s CLS scheme is constructed under the certificateless public key settings to solve the key escrow problem, but it does not address the revocation problem. Our RCLS scheme resolves both the revocation and key escrow problems. For the usage of Lattices, Tian and Huang’s CLS scheme uses the GPV lattice in [20] to generate the private key of a user. It is worth mentioning, that the related parameters in Table 1 have the following relationships: \( m > 2N \log q \), \( m > 64 + N \log q \), \( \hat{s} = \sqrt{m} \alpha(\sqrt{\log N}) \), \( \hat{s} = \sqrt{m} \alpha(\sqrt{\log N}) \), \( s = N^{5/2} \sqrt{2q} \alpha(\sqrt{\log N}) \), \( \hat{\sigma} = 12 \hat{s} \lambda m_1, \sigma = 12 \lambda s N \). In Table 2, we choose concrete parameters: \( N = 512, q = 2^{26}, k = 512, d = 31, \lambda = 14, m_1 = 38,400, m_2 = 25,600 \) and make the comparisons of instances in bit-length. According to Tables 1 and 2, for both the private key size, signature length, the computation costs of signing and verifying processes, our RCLS scheme is better than Tian and Huang’s CLS scheme. Our scheme adopts public channels to send the periodic time update keys.

Indeed, the signing processes of all three schemes mentioned above employ the same the rejection sampling technique in Lyubashevsky’s scheme [22] to produce signatures. Here, lets discuss the rejection probability in the signing process. If the rejection probability is too large, the performance of generating signatures may be inefficient. In our scheme, the signer can produce a useful signature \((z_1, z_2, z_3, z_4, z_5, z_6, c)\) with probability \(\min(2, 10^{-10}) / M\). That is, the signer with probability \((1 - 2^{-10}) / M\) may output a useful signature by Lemma 4. According to the specific parameters \(N = 512, q = 2^{26}, k = 512\) and \(d = 31\) in [22], the \(M\) value is about 7.4. Hence, the performance of signing process still remains efficiency.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Tian and Huang’s CLS Scheme</th>
<th>Our RCLS Scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lattice type</td>
<td>GPV lattice</td>
<td>NTRU lattice</td>
</tr>
<tr>
<td>Public-key setting</td>
<td>CLS</td>
<td>RCLS</td>
</tr>
<tr>
<td>Revocable functionality</td>
<td>No</td>
<td>Public channel</td>
</tr>
<tr>
<td>Averting key escrow problem</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Private key size</td>
<td>(2m_1 k \log(\hat{s}_1 \sqrt{m_1}) + 2m_2 k \log(\hat{s}_2 \sqrt{m_2}))</td>
<td>(6N \log(s \sqrt{N}))</td>
</tr>
<tr>
<td>Signature length</td>
<td>((m_1 + m_2) \log(12 \hat{\sigma}) + \lambda (\log k + 1))</td>
<td>(6N \log(12 \sigma) + \lambda (\log N + 1))</td>
</tr>
<tr>
<td>Computational cost of signing</td>
<td>((m_1 + m_2)(T_1 + 2NT_m + T_s))</td>
<td>(6NT + 9N(T_m + T_s))</td>
</tr>
<tr>
<td>Computational cost of verifying</td>
<td>(2N(m_1 + m_2)T_m + 2NT_s)</td>
<td>(7NT_m + 6NT_s)</td>
</tr>
</tbody>
</table>

Table 2. Comparisons of concrete instances in bit-length.

<table>
<thead>
<tr>
<th>Bit-length</th>
<th>Tian and Huang’s CLS Scheme</th>
<th>Our RCLS Scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>Private key size</td>
<td>595,222,811</td>
<td>127,749</td>
</tr>
<tr>
<td>Signature length</td>
<td>2,026,680</td>
<td>175,312</td>
</tr>
</tbody>
</table>

7. Conclusions

In this paper, we proposed the first provably secure RCLS scheme with a public channel over lattices, which possesses existential unforgeability against adaptive chosen-message attacks. Under the SIS assumption and in the random oracle model, we formally established the security of our lattice-based RCLS scheme for three types of adversaries, namely, outside adversary, honest-but-curious KGC and revoked user. By performance analysis and comparisons, we have demonstrated that the proposed lattice-based RCLS scheme is better than the previously proposed lattice-based CLS scheme, in terms of private key size, signature length, the security property and the revocation mechanism.
Acknowledgments: The authors would like to appreciate anonymous referees for their valuable comments and constructive suggestions. This research was partially supported by Ministry of Science and Technology, Taiwan, under contract no. MOST106-2221-E-018-007-MY2.

Author Contributions: For the research paper, Ying-Hao Hung and Yuh-Min Tseng proposed and designed the RCLS scheme with a public channel over lattices. Sen-Shan Huang presented the background about lattices. Three authors cooperatively proved the security of the proposed scheme. Ying-Hao Hung and Yuh-Min Tseng made performance comparisons.

Conflicts of Interest: The authors declare no conflict of interest.

References


© 2017 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).