Logarithmic-size Lattice-based Linkable Ring Signature for Cloud Data Management

Xue Chen[1], Shiyuan Xu[1(✉)], Fangda Guo[2(✉)], Yuer Yang[1], Chunxiao Li[3(✉)], and Siu-Ming Yiu[1(✉)]

[1] Department of Computer Science, School of Computing and Data Science, The University of Hong Kong, Pokfulam, Hong Kong
`syxu2@cs.hku.hk, xchen.serena666@connect.hku.hk, yueryang@connect.hku.hk, smyiu@cs.hku.hk`

[2] CAS Key Laboratory of AI Safety, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
`guofangda@ict.ac.cn`

[3] School of Artificial Intelligence, Beijing Normal University, Beijing, China
`lichunxiao@bnu.edu.cn`

Abstract. The increasing prevalence of cloud services has promoted storing data on cloud servers to facilitate data sharing, while also posing severe data security challenge. A straightforward approach is to utilize ring signatures in cloud data management to provide cloud data owner anonymity and cloud data unforgeability. Nonetheless, most ring signatures are susceptible to quantum attacks. To address this issue, researchers explore lattice-based ring signatures to enjoy quantum safety. But to our best knowledge, existing lattice-based ring signatures exhibit restricted applicability due to impractically large signature sizes and limited functionality. In this paper, we present a logarithmic-size lattice-based linkable ring signature for cloud data management. To achieve this, we introduce a generalized lattice-based Bulletproofs compression technique and apply it to scheme [AFRICACRYPT'13] to present a logarithmic-size lattice-based ring signature protocol. Building upon this protocol and our designed link label, we propose a logarithmic-size lattice-based linkable ring signature that provides user privacy, ensures cloud data unforgeability, and enables checking if multiple cloud data are from same user. Our scheme is proven to satisfy the anonymity, unforgeability, and linkability. Comprehensive evaluations demonstrate that our scheme outperforms existing state-of-the-art schemes.

Keywords: Cloud Data Management · Security and Privacy · Ring Signature · Lattice Cryptography · Bulletproofs.

1 Introduction

With the widespread development of cloud services [14, 23], the modes of data management [10, 18] have changed, making storing data on cloud servers for sharing convenience more popular [7,12,22]. This cloud-based data management approach enhances data accessibility and collaboration among users, and facilitates data processing and sharing. However, as data is transmitted and stored on cloud servers for sharing, the concerns about cloud data owner privacy [5], cloud data confidentiality and integrity [6] are becoming more prominent, which poses significant data security challenges that should not be ignored [24].

Ring signatures, initially proposed by Rivest et. al [13] in 2001, have garnered significant attention with strong anonymity and unforgeability, which are achieved through their non-administrator nature. Each individual ring user possesses a pair of keys (a secret key and a public key), enabling ring members to choose a subset of public keys and generate a signature for messages representing the associated ring without any external assistance, thereby effectively concealing their true identity within the ring. The ideal anonymity and unforgeability of ring signatures have attracted many scholars [5, 17] to apply them in cloud data management to ensure cloud data owner privacy and cloud data integrity.

Most existing ring signatures are based on classical assumption [15,19]. However, Shor [16] in 1997 proved that under quantum attacks, problems such as discrete logarithms can be solved in the polynomial time, which leads to the reality that most, if not all, of the current mainstream classical cryptographic schemes cannot be quantum-resistant. For instances, if one can use Shor's [16] and Grover's [9] algorithms to extract user keys from their public keys to generate variant transactions without authorization or forged signatures, the privacy of valid customers will be disclosed. Thus, the security resisting quantum attacks is crucial and far-reaching for heightened requirements of cloud data management.

To address this, scholars research on post-quantum ciphers [20, 21], among which lattice ciphers [2] are the most popular due to their advantages, such as provable security and explicit algebraic structure. Aguilar Melchor, C. [1] propose a lattice-based ring signature that offer the quantum security; however, that is impractical since the signature size is at least $O(N)$. Among most of the existing studies, the signature sizes are often linear, also with limited functionality, making them impractical for cloud data management.

Consequently, with the aforementioned issues, our work motivates by: Can we propose a logarithmic-size lattice-based linkable ring signature for cloud data management?

1.1 Our Contributions

- **Generalized Lattice-based Bulletproofs Compression.** Inspired by the previous work [3], we generalize the inner-produce argument in Bulletproofs to the form that is appropriate for Σ-protocol and then optimize the generalized Bulletproofs compression under the lattice setting. Our generalized lattice-based Bulletproofs compression technique is suitable for the Σ-based protocols/schemes. With the help of the generalized lattice-based Bulletproofs compression technique, we can reduce the size from N to $2 \log N$, thereby considerably improving the efficiency.
- **Logarithmic-Size Lattice-based Linkable Ring Signature** We first convert the lattice-based ring signature in [1] into the form of Σ-protocol.

Then, we apply our generalized lattice-based Bulletproofs compression technique to the above Σ-protocol to present our interactive logarithmic-size lattice-based ring signature protocol. Building on this protocol and the help of to the Fiat-Shamir heuristic, we incorporate our designed link label to propose a logarithmic-size lattice-based linkable ring signature which is suitable with cloud data management.

- **Solve the Security Challenges in Cloud Data Management** By leveraging our proposed logarithmic-size lattice-based linkable ring signature, cloud data owners can sign cloud data without compromising their true identity, ensuring user anonymity and the unforgeability of cloud data. Our scheme allows cloud users to verify the authenticity and linkability of cloud data independently, without relying on a central authority, enabling secure data management in the cloud while preserving privacy and security.
- **Security Analysis and Performance Evaluation** The security analysis prove that our proposed logarithmic-size lattice-based linkable ring signature satisfy the anonymity, unforgeability and linkability. The performance evaluation show that our proposed logarithmic-size lattice-based linkable ring signature outperform existing state-of-the-art schemes.

2 Preliminaries

2.1 Lattice

Definition 1 (Lattice) *Suppose that $\mathbf{A} = \mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_m \in \mathbb{R}^n$ are m linearly independent vectors, then the set of linear combinations is called lattice L, denoted by $L = L(\mathbf{A}) = \{x_1 \cdot \mathbf{a}_1 + x_2 \cdot \mathbf{a}_2 + \cdots + x_m \cdot \mathbf{a}_m | x_i \in \mathbb{Z}\}$, where the matrix $\mathbf{A} \in \mathbb{R}^{n\times m}$ is a set of the basis of L, m is the rank of L, and n is the dimension of L. In particular, when $m = n$, L is called a full rank.*

Definition 2 (Short Integer Solution (SIS) Hardness [11]) *Given the matrix $\mathbf{A} \leftarrow R_q^{n\times m}$, and find the vector $\mathbf{z} \in R_q^m$ such that $\mathbf{A}\mathbf{z} = 0 \mod q$ and $0 < \|\mathbf{z}\| \leq \gamma$.*

Definition 3 (Learning with Errors (LWE) Hardness [11]) *Given the $\mathcal{X}_\mathcal{B}$ as the distribution over R_q, the secret key as $\mathbf{s} \leftarrow {\mathcal{X}_\mathcal{B}}^n$. Define $LWE_{(q,\mathbf{s})}$ as the distribution obtained by sampling $\mathbf{a} \leftarrow R_q^n$, $e \leftarrow \mathcal{X}_\mathcal{B}$, output $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s}\rangle + e)$. We aim to distinguish between m given samples from either $LWE_{(q,\mathbf{s})}$ or $\mathcal{U}(R_q^n, R_q)$.*

Lemma 1 (Theorem 1 in [1]) *If there is a polynomial-time algorithm that can solve the hash collision problem with some non-negligible probability, then there is a polynomial-time algorithm that can solve the Shortest Vector Problem SVP_γ for each lattice corresponding to the ideal in $\mathcal{R}$, where $\gamma = 16dmn\log^2 n$.*

2.2 Rejection Sampling

In the Σ-protocol, the rejection sampling algorithm is utilized to ensure that the prover's response value is not dependent on the secret information. Specially, the witness $\mathbf{a}$ must be hidden with the challenge x and mask vector $\mathbf{y}$, compute $\mathbf{z} = x \cdot \mathbf{a} + \mathbf{y}$. The rejection sampling algorithm is able to constraint the distribution of $(x, \mathbf{z})$ to be independent of $\mathbf{a}$. When a response value $\mathbf{z}$ exceeds the bound, the rejection sampling algorithm will reject it.

2.3 Σ-Protocol.

The Σ protocol is a three-move type interactive protocol with three phases (commitment, challenge, and response), which is used by the prover $\mathcal{P}$ to convince the verifier $\mathcal{V}$ that some statement is true without revealing its real value. Specially, the Σ-protocol is one of the zero-knowledge proofs between two entities, the prover $\mathcal{P}$ and the verifier $\mathcal{V}$, and also existing a Probabilistic Polynomial Time (PPT) setup algorithm $\mathcal{G}$. The triple $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is defined for the polynomial-time decidable relation $\mathcal{R}$, and for $(v, w) \subseteq \mathcal{R}$, w is referred to as a witness for v.

Definition 4 (Completeness) *The triple algorithm $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is called the Σ-protocol, with the perfectly completeness, if for all PPT adversaries $\mathcal{A}$*

$$Pr\left[\mathcal{V}(pp, v, \mu, x, z) = 1 \,\middle|\, \begin{array}{c} pp \leftarrow \mathcal{G}(1^\lambda); (v, w) \leftarrow \mathcal{A}(pp); \\ \mu \leftarrow \mathcal{P}(pp, v, w); x \leftarrow \{0,1\}^\lambda; z \leftarrow \mathcal{P}(x); \end{array}\right] = 1.$$

Definition 5 (Soundness) *The triple $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is n-special soundness if there exists an efficient PPT extractor $\mathcal{E}$ that has the ability to compute the witness w by given n accepting transcripts with the same initial message. Thus, for all PPT adversaries $\mathcal{A}$*

$$Pr\left[(pp, v, w) \in R \,\middle|\, \begin{array}{l} pp \leftarrow \mathcal{G}(1^\lambda); (v, \mu, x_i, z_i) \leftarrow \mathcal{A}(pp); \\ \mathcal{V}(pp, v, \mu, x_i, z_i) = 1, \forall\ i \in [1, n]; \\ w \leftarrow \mathcal{E}(pp, v, \mu, x_1, z_1, \cdots, x_n, z_n); \end{array}\right] \approx 1 - \mathsf{negl}.$$

where the negl *is negligible, we say that the protocol is soundness.*

Definition 6 (Honest Verifier Zero-Knowledge (HVZK)) *The triple $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is special honest verifier zero-knowledge if there exists an efficient PPT simulator $\mathcal{S}$ such that for all interactive PPT adversaries $\mathcal{A}$*

$$\left| Pr\left[\begin{array}{c} \mathcal{A}(\mu, z) \\ = 1 \end{array} \,\middle|\, \begin{array}{c} pp \leftarrow \mathcal{G}(1^\lambda); \\ (v, w, x) \leftarrow \mathcal{A}(pp); \\ \mu \leftarrow \mathcal{P}(pp, v, w); \\ z \leftarrow \mathcal{P}(x); \end{array}\right] - Pr\left[\begin{array}{c} \mathcal{A}(\mu, z) \\ = 1 \end{array} \,\middle|\, \begin{array}{c} pp \leftarrow \mathcal{G}(1^\lambda); \\ (v, w, x) \leftarrow \mathcal{A}(pp); \\ (\mu, z) \leftarrow \mathcal{S}(pp, v, x); \end{array}\right] \right| \leq \mathsf{negl},$$

where the adversary $\mathcal{A}$ outputs (v, w, x) such that $(pp, v, w) \in \mathcal{R}$ and $x \in \{0,1\}^\lambda$.

2.4 Compression Technology

Bulletproofs [3] is the interactive protocol for proving arbitrary statements, which only relies on the discrete logarithm assumption. In Bulletproofs, the relation is $P = \mathbf{g}^{\mathbf{z}}$. We set $n' = \frac{n}{2}$, and divide $\mathbf{z} = (z_0, \cdots, z_{n-1})$ and $\mathbf{g} = (g_0, \cdots, g_{n-1})$ into two vectors respectively, i.e. $\mathbf{z}_{[:n']} = (z_0, \cdots, z_{n'-1})$, $\mathbf{z}_{[n':]} = (z_{n'}, \cdots, z_{n-1})$ and $\mathbf{g}_{[:n']} = (g_0, \cdots, g_{n'-1})$, $\mathbf{g}_{[n':]} = (g_{n'}, \cdots, g_{n-1})$. Then, we compute the $L = {\mathbf{g}_{[:n']}}^{\mathbf{z}_{[n':]}}$ and $R = {\mathbf{g}_{[n':]}}^{\mathbf{z}_{[:n']}}$. The process of compressing the size of $\mathbf{z}$ can be compute as $\mathbf{z'} = x \cdot \mathbf{z}_{[:n']} + x^{-1} \cdot \mathbf{z}_{[n':]}$, where x is the random sampled. In each round, $\mathcal{P}$ and $\mathcal{V}$ compute a new vector $\mathbf{g'} = \mathbf{g}_{[:n']}^{x^{-1}} \circ \mathbf{g}_{[n':]}^{x}$ to replace the original $\mathbf{g}$. The verification equation is $P' = L^{x^2} \cdot P \cdot R^{x^{-2}}$.

3 Framework Description

In this section, we show our system architecture for cloud data management and the formal definition of our $O(\log N)$ lattice-based linkable ring signature.

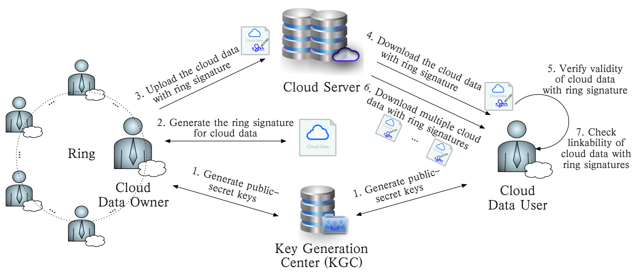

3. Upload the cloud data with ring signature
Cloud Server
4. Download the cloud data with ring signature
5. Verify validity of cloud data with ring signature
2. Generate the ring signature for cloud data
6. Download multiple cloud data with ring signatures
Ring
7. Check linkability of cloud data with ring signatures
Cloud Data Owner
1. Generate public-secret keys
1. Generate public-secret keys
Cloud Data User
Key Generation Center (KGC)

Fig. 1. Cloud Data Management System Architecture.

3.1 System Architecture

Our cloud data management system architecture is depicted in Fig. 1, which consists of four entities: Key Generation Center (KGC), Cloud Data Owner, Cloud Data User, and Cloud Server. The specific workflow of each entity as:

– **Key Generation Center (KGC)**: The KGC generates public-secret key pairs for cloud data owners and cloud data users. Note that the secret keys of cloud data owners and cloud data users are kept confidential.
– **Cloud Data Owner**: The cloud data owner owns a cloud data μ and a public-secret keys pk_j, sk_j. In our system, the cloud data owner conceals her identity within the ring to safeguard her privacy, generating a ring signature σ for the cloud data μ, and uploading σ to the cloud server for storage to guarantee the unforgeability of the cloud data.
– **Cloud Data User**: The cloud data user owns a public-secret keys pk_i, sk_i. In our system, the cloud data user can download cloud data from the cloud server and verify its validity. Further, the cloud data user can download multiple cloud data from the cloud server and verify their linkability.
– **Cloud Server**: The cloud server stores the the cloud data μ with its ring signature σ from the cloud data owner. Upon receiving requests from cloud data user, it returns the cloud data μ with its ring signature σ.

3.2 Formal Definition

Our proposed logarithmic-size lattice-based linkable ring signature consist of five algorithms: Ring-Setup, Ring-KeyGen, Ring-Sign, Ring-Verify, Ring-Link. The syntax of these algorithms are as follows.

- Ring-Setup: Give the security parameter λ, this algorithm returns the public parameter $\mathcal{P}$.
- Ring-KeyGen: Given the public parameters $\mathcal{P}$, this PPT algorithm returns the public key pk_i and secret key sk_i of user i.
- Ring-Sign: Given the message μ, a ring $\mathcal{R}$ of $[N]$ members associated the public keys set L_{pk}, and signer's secret key sk_l, this PPT algorithm returns the ring signature σ with the link label L.
- Ring-Verify: Given the message μ, a ring $\mathcal{R}$ of $[N]$ members with the public keys set L_{pk}, and a ring signature σ with link label L, this deterministic algorithm verify the validity of ring signature. If the ring signature is valid, this algorithm returns *accept*; otherwise it returns *reject*.
- Ring-Link: Given two ring signatures σ_1 and σ_2 with the link labels L_1 and L_2, this deterministic algorithm check if the link labels L_1, L_2 are equal. If $L_1 = L_2$, it returns *link*; otherwise it returns *unlink*.

4 Logarithmic-size Lattice-based Linkable Ring Signature

In this section, we present our logarithm size lattice-based linkable ring signature for cloud data management.

4.1 Generalized Lattice-based Bulletproofs Compression Technique

To begin with, we convert the original Bulletproofs to the form of Σ-protocol, called the generalized Bulletproofs compression technique. In particular, the relation for applying the Bulletproofs compression is $P = \mathbf{g}^{\boldsymbol{\zeta}}$. In one round compression, the prover sets $N' = \frac{N}{2}$, and divides $\boldsymbol{\zeta} = (\zeta_0, \cdots, \zeta_{N-1})$ and $\mathbf{g} = (g_0, \cdots, g_{N-1})$ into two vectors respectively, i.e. $\boldsymbol{\zeta}_{[:N']} = (\zeta_0, \cdots, \zeta_{N'-1})$, $\boldsymbol{\zeta}_{[N':]} = (\zeta_{N'}, \cdots, \zeta_{N-1})$ and $\mathbf{g}_{[:N']} = (g_0, \cdots, g_{N'-1})$, $\mathbf{g}_{[N':]} = (g_{N'}, \cdots, g_{N-1})$. Then, the prover computes the cross-terms $\boldsymbol{\xi_1} = \boldsymbol{g}_{[\boldsymbol{N'}:]}^{\boldsymbol{\zeta}_{[:\boldsymbol{N'}]}}$, $\boldsymbol{\eta_1} = \boldsymbol{g}_{[:\boldsymbol{N'}]}^{\boldsymbol{\zeta}_{[\boldsymbol{N'}:]}}$ and sends that to the verifier. The verifier randomly samples $c \overset{\$}{\leftarrow} \mathcal{C}$, and then sends c to the prover. After receiving the c, the prover computes the $\mathbf{g_1} = \mathbf{g}_{[:\mathbf{N'}]}{}^{c_2^{-1}} \circ \mathbf{g}_{[\mathbf{N'}:]}{}^{c_2}$ and $\boldsymbol{\zeta_1} = \boldsymbol{\zeta}_{[:\boldsymbol{N'}]} \cdot c_2 + \boldsymbol{\zeta}_{[\boldsymbol{N'}:]} \cdot c_2^{-1}$ to achieve the goal of compression.

However, for the goal of against the quantum computing attacks, we have to show how to transfer the generalized Bulletproofs compression under the discrete logarithm to the lattice setting in the form of Σ-protocol.

In particular, the prover sets $\mathbf{g}$ and $\boldsymbol{\zeta}$ to $\mathbf{g} = [\mathbf{g_1}\ \mathbf{g_2}]$ and $\boldsymbol{\zeta} = [\begin{smallmatrix}\boldsymbol{\zeta_1}\\ \boldsymbol{\zeta_2}\end{smallmatrix}]$, where $\mathbf{g_1}, \mathbf{g_2}$ and $\boldsymbol{\zeta_1}, \boldsymbol{\zeta_2}$ are $m \times N'$ matrices, $N' = \frac{N}{2}$. After that, the prover lets ξ

and η as $\xi = \boldsymbol{g_1} \cdot \boldsymbol{\zeta_2}$ and $\eta = \boldsymbol{g_2} \cdot \boldsymbol{\zeta_1}$, and transmits them to verifier. The verifier randomly selects the challenge $c \overset{\$}{\leftarrow} \{\mathcal{C}\}$ and sends it to the prover. The $\mathbf{g}'$ and $\boldsymbol{\zeta'}$ used in the next round of compression are calculated as $\mathbf{g}' = c \cdot \mathbf{g_1} + \mathbf{c^{-1}} \cdot \mathbf{g_2}$ and $\boldsymbol{\zeta'} = c^{-1} \cdot \boldsymbol{\zeta_1} + c \cdot \boldsymbol{\zeta_2}$. Then, for all c, we have $(c \cdot \mathbf{g_1} + \mathbf{c^{-1}} \cdot \mathbf{g_2})(c^{-1} \cdot \boldsymbol{\zeta_1} + c \cdot \boldsymbol{\zeta_2}) = t + c^2 \cdot \xi + c^{-2} \cdot \eta$, where $t = \boldsymbol{g} \cdot \boldsymbol{\zeta}$. The relationship authenticated by the verifier is $\mathbf{g}' \cdot \boldsymbol{\zeta'} \overset{?}{=} t + c^2 \cdot \xi + c^{-2} \cdot \eta$. If the compression is continued, $\boldsymbol{\zeta}'$ and $\mathbf{g}'$ are used as the input instead of $\boldsymbol{\zeta}$ and $\mathbf{g}$ to continue the compression until optimization.

4.2 Σ-Protocol for Our Lattice-based Ring Signature

Subsequently, we transform the scheme in [1] into the Σ-protocol to be compatible with the above lattice-based generalized Bulletproofs compression technique. Note that in the original signing algorithm, the challenge e is generated by the hash function $\mathcal{H}$. While converting it to the interactive protocol, by the Fiat-Sharmir technique, the original hash function is assumed by the verifier and the challenge e is randomly sampled.

Following the above works, we can implement the generalize lattice-based Bulletproofs compression technique to the ring signature scheme in the form of Σ-protocol. Our proposed interactive lattice-based ring signature protocol with one round of compression using Bulletproofs compression is shown in Fig. 2.

In traditional case, using Bulletproofs to compress the signature, its size decreases linearly with the number of rounds, and we can reduce the size of the signature to 1 by $\log N$ rounds of compression. However, in the setting of the lattice, after a certain number of rounds of compression, its size may instead go back up. Therefore, we need to compress the signature to a minimum to achieve optimal results by multiple rounds, not necessarily $\log N$ rounds.

4.3 Our Logarithmic-size Lattice-based Linkable Ring Signature for Cloud Data Management

Finally, utilizing the protocol depicted in Fig. 2 along with our designed link label and the Fiat–Shamir heuristic, we propose an efficient logarithmic-size lattice-based linkable ring signature scheme for cloud data management.

4.3.1 Cloud System Initialization We introduce the cloud system initialization stage first. We take the security parameter λ as the input, and then set the integers n, m, p, hash functions $\mathcal{H}, \mathcal{H}_l$ and parameter S, as Algorithm 1.

Algorithm 1 Ring-Setup(1^λ)

```
1: Set $n$ as a power of 2, $m = (3 + 2c/3)\log n$ and $p$ as a prime larger than $n^4$
2: Set hash functions $\mathcal{H} : \{0, 1\}$, $\mathcal{H}_l : \{0, 1\}$
3: Set $S \leftarrow \mathcal{R}, S \neq 0$
4: return $\mathcal{P} = (\lambda, n, m, p, S, \mathcal{H}, \mathcal{H}_l)$
```

Prover $(\mathcal{P};(\hat{\boldsymbol{\omega}}, S))$ **Verifier** $(\mathcal{P};(\mathbf{g}, S))$

$\hat{\mathbf{g}} = (g_1, g_2, \cdots, g_n)$

for all $i \in [N]$

if $i \neq j; \hat{\boldsymbol{v}}_i \leftarrow \mathcal{R}_z^m$

else if $i = j; \hat{\boldsymbol{v}}_j \leftarrow \mathcal{R}_v^m$

$\psi \leftarrow Com(\hat{\boldsymbol{v}})$

$$\xrightarrow{\psi}$$

$e \xleftarrow{\$} \mathcal{C}^*$

$$\xleftarrow{e}$$

for $i = j; \hat{\boldsymbol{\zeta}}_j \leftarrow \hat{\boldsymbol{\omega}}_j \cdot e + \hat{\boldsymbol{v}}_j$

if $\hat{\boldsymbol{\zeta}}_j \notin \mathcal{R}_z^m$; go back to re-sample

for $i \neq j; \hat{\boldsymbol{\zeta}}_i = \hat{\boldsymbol{v}}_i$

$\hat{\boldsymbol{\zeta}} \leftarrow \hat{\boldsymbol{\omega}} \cdot e + v$

Set $\hat{\mathbf{g}}^{(0)} = \hat{\mathbf{g}}$ and $\hat{\boldsymbol{\zeta}}^{(0)} = \hat{\boldsymbol{\zeta}}$

Compute cross-term

1: parse $\hat{\mathbf{g}}^{(0)} = [\hat{\mathbf{g_1}}^{(0)} \ \hat{\mathbf{g_2}}^{(0)}]$

2: parse $\hat{\boldsymbol{\zeta}}^{(0)} = \begin{bmatrix}\hat{\boldsymbol{\zeta_1}}^{(0)} \\ \hat{\boldsymbol{\zeta_2}}^{(0)}\end{bmatrix}$

3: $\xi^{(1)} = \hat{\mathbf{g_1}}^{(0)} \cdot \hat{\boldsymbol{\zeta_2}}^{(0)}$

4: $\eta^{(1)} = \hat{\mathbf{g_2}}^{(0)} \cdot \hat{\boldsymbol{\zeta_1}}^{(0)}$

$$\xrightarrow{\xi, \eta}$$

$c^{(1)} \xleftarrow{\$} \mathcal{C}^*$

$$\xleftarrow{c^{(1)}}$$

Compute $\mathbf{g}^{\hat{(\mathbf{1})}}$

1: $\mathbf{g}^{\hat{(\mathbf{1})}} = c^{(1)} \cdot \hat{\mathbf{g_1}}^{(0)} + {c^{(1)}}^{-1} \cdot \hat{\mathbf{g_2}}^{(0)}$

$\boldsymbol{\zeta}^{\hat{(\mathbf{1})}} = {c^{(1)}}^{-1} \cdot \hat{\boldsymbol{\zeta_1}}^{(0)} + c^{(1)} \cdot \hat{\boldsymbol{\zeta_2}}^{(0)}$

$$\xrightarrow{\boldsymbol{\zeta}^{\hat{(\mathbf{1})}}}$$

check if $\boldsymbol{\zeta}^{\hat{(\mathbf{1})}} \in \mathcal{R}_z^m$

$\mathbf{g}^{\hat{(\mathbf{1})}} \cdot \boldsymbol{\zeta}^{\hat{(\mathbf{1})}} \overset{?}{=} \psi + S \cdot e + {c^{(1)}}^2 \cdot \xi^{(1)} + {c^{(1)}}^{-2} \cdot \eta^{(1)}$

Fig. 2. Our Logarithmic-size Lattice-based Ring Signature Protocol.

4.3.2 Cloud Owner/User Registration Each cloud data owner and cloud data user have to register in the KGC to obtain their public key and secret key. For user i, the public key pk_i and the secret key sk_i satisfy the relation $pk_i \cdot sk_i = S$. The registration process is shown in Algorithm 2.

4.3.3 Generating Ring Signature for Cloud Data Now, we demonstrate how the cloud data owner j can generate a ring signature for cloud data μ to ensure cloud data owner privacy and cloud data unforgeability. Specifically, we first use the Fiat–Shamir heuristic to transfer the protocol shown in Fig. 2 to the non-interactive one. Then, we design the link label as $L = \mathcal{H}_l(S, sk_j)$. Our generalized lattice-based Bulletproofs compression is used to compress the size of $\hat{\boldsymbol{\zeta}}$. After r_o round compression, cloud data owner obtains the ring signature $\sigma = (\boldsymbol{\zeta}^{\hat{(\boldsymbol{r_o})}}, \mathbf{g}^{\hat{(\mathbf{r_o})}}, \psi, \xi^{(i)_{i\in[r_o]}}, \eta^{(i)_{i\in[r_o]}}, L)$ and uploads the cloud data μ with its ring signature σ to the cloud server, shown in Algorithm 3.

4.3.4 Verifying Ring Signature for Cloud Data Following this, we show how the cloud data user can verify the validity of cloud data μ with its ring signature σ after downloading them from the cloud server. The cloud data user first have to check the bound of $\boldsymbol{\zeta}^{\hat{(\boldsymbol{r_o})}}$ and then check if the equation $\boldsymbol{\zeta}^{\hat{(\boldsymbol{r_o})}} \cdot \mathbf{g}^{\hat{(\mathbf{r_o})}} \overset{?}{=} \psi + S \cdot e + {c^{(i)}}^2 \cdot \xi^{(i)} + {c^{(i)}}^{-2} \cdot \eta^{(i)}$ is hold, shown in Algorithm 4.

4.3.5 Checking Linkability of Ring Signatures for Cloud Data Lastly, our scheme also supports the cloud data user checking if multiple cloud data from the same cloud data owner by the link labels, as shown in Algorithm 5.

5 Security Analysis

In this section, the correctness, anonymity, unforgeability and linkability have be proven. In the security analysis, we showcase the one round compression.

5.1 Correctness

Firstly, we prove that our proposed lattice-based linkable ring signature satisfies the correctness. In particular, the relation used the Bulletproofs [3] is $P = \hat{\boldsymbol{\zeta}} \cdot \hat{\mathbf{g}}$, which is equal to $\psi + S \cdot e$. The correctness of Bulletproofs can be expressed as $P' = P + {c^{(i)}}^2 \cdot \xi^{(i)} + {c^{(i)}}^{-2} \cdot \eta^{(i)}$, i.e., $\boldsymbol{\zeta}^{\hat{(\boldsymbol{r_o})}} \cdot \mathbf{g}^{\hat{(\mathbf{r_o})}} = \psi + S \cdot e + {c^{(i)}}^2 \cdot \xi^{(i)} + {c^{(i)}}^{-2} \cdot \eta^{(i)}$.

Algorithm 2 Ring-KeyGen($\mathcal{P}$)

```
1: Randomly choose $\hat{\boldsymbol{\omega}}_{\boldsymbol{i}} \leftarrow \mathcal{R}_{s,c}^m$
2: if $\hat{\boldsymbol{\omega}}_{\boldsymbol{i}}$ is not invertible then
3:     go back to setp. 1
4: end if
5: Compute $\hat{\mathbf{g}}_{\mathbf{i}} \cdot \hat{\boldsymbol{\omega}}_{\boldsymbol{i}} = S$ ($S$ is fixed)
6: return $(pk, sk) = (\hat{\mathbf{g}}_{\mathbf{i}}, \hat{\boldsymbol{\omega}}_{\boldsymbol{i}})$
```

Algorithm 3 Ring-Sign$(\hat{\boldsymbol{\omega}}_{\boldsymbol{j}}, \mu, \mathcal{R}, L_{pk})$

```
1: Compute $L = \mathcal{H}_l(S, \omega_l)$
2: for all $i \in [N]$ do
3:     if $i \neq j$ then
4:         Randomly choose $\hat{\boldsymbol{v}}_i \leftarrow \mathcal{R}_z^m$
5:     end if
6:     if $i = j$ then
7:         Randomly choose $\hat{\boldsymbol{v}}_j \leftarrow \mathcal{R}_v^m$
8:     end if
9: end for
10: Compute $\psi \leftarrow \hat{\mathbf{g}}_{\mathbf{i}} \cdot \hat{\boldsymbol{v}}_{\boldsymbol{i}}$
11: Compute $e \leftarrow \mathcal{H}(\psi, L_{pk}, \mu)$
12: for all $i \in [N]$ do
13:     if $i = j$ then
14:         Compute $\hat{\boldsymbol{\zeta}}_j \leftarrow \hat{\boldsymbol{\omega}}_j \cdot e + \hat{\boldsymbol{v}}_j$
15:         if $\hat{\boldsymbol{\zeta}}_j \notin \mathcal{R}_z^m$ then
16:             go back to re-sample
17:         end if
18:     end if
19:     if $i \neq j$ then
20:         Compute $\hat{\boldsymbol{\zeta}}_i = \hat{\boldsymbol{v}}_i$
21:     end if
22: end for
23: Set $\hat{\mathbf{g}}^{(0)} = \hat{\mathbf{g}}$ and $\hat{\boldsymbol{\zeta}}^{(0)} = \hat{\boldsymbol{\zeta}}$
24: for $i$ from 1 to $r_o$ do
25:     Parse $\hat{\mathbf{g}}^{(i-1)} = [\hat{\mathbf{g}_1}^{(i-1)}\ \hat{\mathbf{g}_2}^{(i-1)}]$
26:     Parse $\hat{\boldsymbol{\zeta}}^{(i-1)} = [\begin{matrix}\hat{\boldsymbol{\zeta}_1}^{(i-1)} \\ \hat{\boldsymbol{\zeta}_2}^{(i-1)}\end{matrix}]$
27:     Compute $\xi^{(i)} = \hat{\mathbf{g}_1}^{(i-1)} \cdot \hat{\boldsymbol{\zeta}_2}^{(i-1)}$
28:     Compute $\eta^{(i)} = \hat{\mathbf{g}_2}^{(i-1)} \cdot \hat{\boldsymbol{\zeta}_1}^{(i-1)}$
29:     Compute $c^{(i)} \leftarrow \mathcal{H}(\xi^{(i)}, \eta^{(i)})$
30:     Compute $\hat{\mathbf{g}^{(\mathbf{i})}} = c^{(i)} \cdot \hat{\mathbf{g}_1}^{(i-1)} + {c^{(i)}}^{-1} \cdot \hat{\mathbf{g}_2}^{(i-1)}$
31:     Compute $\hat{\boldsymbol{\zeta}^{(\boldsymbol{i})}} = {c^{(i)}}^{-1} \cdot \hat{\boldsymbol{\zeta}_1}^{(i-1)} + c^{(i)} \cdot \hat{\boldsymbol{\zeta}_2}^{(i-1)}$
32: end for
33: return $\sigma = (\hat{\boldsymbol{\zeta}^{(\boldsymbol{r_o})}}, \hat{\mathbf{g}^{(\mathbf{r_o})}}, \psi, {\xi^{(i)}}_{i \in [r_o]}, {\eta^{(i)}}_{i \in [r_o]}, L)$
```

Algorithm 4 Ring-Verify(μ, L_{pk}, σ)

```
1: Check the bound $\boldsymbol{\zeta}^{(\hat{r_o})} \in \mathcal{R}_z^m$
2: Compute $e \leftarrow \mathcal{H}(\psi, L_{pk}, \mu)$ and $c^{(i)} \leftarrow \mathcal{H}(\xi^{(i)}, \eta^{(i)})$
3: if $\boldsymbol{\zeta}^{(\hat{r_o})} \cdot \mathbf{g}^{(\hat{\mathbf{r_o}})} = \psi + S \cdot e + {c^{(i)}}^2 \cdot \xi^{(i)} + {c^{(i)}}^{-2} \cdot \eta^{(i)}$ then
4:     return 'accept'
5: end if
6: if $\boldsymbol{\zeta}^{(\hat{r_o})} \cdot \mathbf{g}^{(\hat{\mathbf{r_o}})} \neq \psi + S \cdot e + {c^{(i)}}^2 \cdot \xi^{(i)} + {c^{(i)}}^{-2} \cdot \eta^{(i)}$ then
7:     return 'reject'
8: end if
```

Algorithm 5 Ring-Link(σ_1, σ_2)

```
1: Parse $\sigma_1 = ({\boldsymbol{\zeta}^{(\hat{r_o})}}_1, {\mathbf{g}^{(\hat{\mathbf{r_o}})}}_1, \psi_1, {\xi_1^{(i)}}^{i \in [r_o]}, {\eta_1^{(i)}}^{i \in [r_o]}, L_1)$
2: Parse $\sigma_2 = ({\boldsymbol{\zeta}^{(\hat{r_o})}}_2, {\mathbf{g}^{(\hat{\mathbf{r_o}})}}_2, \psi_2, {\xi_2^{(i)}}^{i \in [r_o]}, {\eta_2^{(i)}}^{i \in [r_o]}, L_2)$
3: if $L_1 = L_2$ then
4:     return 'link'
5: end if
6: if $L_1 \neq L_2$ then
7:     return 'unlink'
8: end if
```

Then, we need to prove the correctness of the relation verified by the verifier. We prove this by taking a one-round compression as an example, i.e.

$$\begin{aligned}
\mathbf{g}^{\hat{(1)}} \cdot \boldsymbol{\zeta}^{\hat{(1)}} &= ({\mathbf{g}^{\hat{(0)}}}_1 \cdot c^{(1)} + {\mathbf{g}^{\hat{(0)}}}_2 \cdot {c^{(1)}}^{-1}) \cdot ({\boldsymbol{\zeta}^{\hat{(0)}}}_1 \cdot {c^{(1)}}^{-1} + {\boldsymbol{\zeta}^{\hat{(0)}}}_2 \cdot c^{(1)}) \\
&= {\mathbf{g}^{\hat{(0)}}}_1 \cdot {\boldsymbol{\zeta}^{\hat{(0)}}}_1 \cdot c^{(1)} \cdot {c^{(1)}}^{-1} + {\mathbf{g}^{\hat{(0)}}}_1 \cdot {\boldsymbol{\zeta}^{\hat{(0)}}}_2 \cdot c^{(1)} \cdot c^{(1)} \\
&\quad + {\mathbf{g}^{\hat{(0)}}}_2 \cdot {\boldsymbol{\zeta}^{\hat{(0)}}}_1 \cdot {c^{(1)}}^{-1} \cdot {c^{(1)}}^{-1} + {\mathbf{g}^{\hat{(0)}}}_2 \cdot {\boldsymbol{\zeta}^{\hat{(0)}}}_2 \cdot {c^{(1)}}^{-1} \cdot c^{(1)} \\
&= ({\mathbf{g}^{\hat{(0)}}}_1 \cdot {\boldsymbol{\zeta}^{\hat{(0)}}}_1 + {\mathbf{g}^{\hat{(0)}}}_2 \cdot {\boldsymbol{\zeta}^{\hat{(0)}}}_2) + {\mathbf{g}^{\hat{(0)}}}_1 \cdot {\boldsymbol{\zeta}^{\hat{(0)}}}_2 \cdot {c^{(1)}}^2 + {\mathbf{g}^{\hat{(0)}}}_2 \cdot {\boldsymbol{\zeta}^{\hat{(0)}}}_1 \cdot {c^{(1)}}^{-2} \\
&= \psi + S \cdot e + \xi^{(1)} \cdot {c^{(1)}}^2 + \eta^{(1)} \cdot {c^{(1)}}^{-2}
\end{aligned}$$

For linking correctness, the cloud data owner j signs two cloud data μ_1 and μ_2 using the same secret key sk_j to generate two signatures σ_1 and σ_2 containing the link labels $L_1 = \mathcal{H}_l(S, sk_j)$ and $L_2 = \mathcal{H}_l(S, sk_j)$, respectively. Since L_1, L_2 are generated with the same parameter S, if the user signs the cloud data μ_1, μ_2 with the same secret key sk_j, then it must be the case that $L_1 = L_2$.

5.2 Anonymity

Theorem 1 *If our proposed scheme satisfies HVZK, i.e., there exists an efficient simulator $\mathcal{S}$ which generates a tuple $(\boldsymbol{\zeta}^{\hat{(1)}}, \mathbf{g}^{(1)}, \psi, \xi^{(1)},\ eta^{(1)})$ that is in-*

distinguishable from the transcript generated by the real protocol run between the prover and the verifier, then we say our scheme satisfies anonymity.

Proof. In HVZK, we require the verifier is honest. Then, we construct the simulator $\mathcal{S}$ that outputs the same distribution by running the protocol "in reverse" and imitating a forgery signature without the secret key, as in Algorithm 6.

Algorithm 6 Simulator $\mathcal{S}$

```
1: Randomly choose $\boldsymbol{\zeta}_1^{\hat{(0)}}, \boldsymbol{\zeta}_2^{\hat{(0)}}$ and parameter $c^{(1)}$
2: Set $\boldsymbol{\zeta}^{\hat{(0)}} = [\begin{smallmatrix}\boldsymbol{\zeta}_1^{\hat{(0)}} \\ \boldsymbol{\zeta}_2^{\hat{(0)}}\end{smallmatrix}]$, $\mathbf{g}^{\hat{(0)}} = [\mathbf{g}_1^{\hat{(0)}}\ \mathbf{g}_2^{\hat{(0)}}]$
3: Calculate $\boldsymbol{\zeta}^{\hat{(1)}} = \boldsymbol{\zeta}_1^{\hat{(0)}} \cdot {c^{(1)}}^{-1} + \boldsymbol{\zeta}_2^{\hat{(0)}} \cdot c^{(1)}$, $\mathbf{g}^{\hat{(\mathbf{1})}} = c^{(1)} \cdot \hat{\mathbf{g_1}}^{(0)} + {c^{(1)}}^{-1} \cdot \hat{\mathbf{g_2}}^{(0)}$
4: Calculate the $\xi^{(1)} = \mathbf{g}_1^{\hat{(0)}} \cdot \boldsymbol{\zeta}_2^{\hat{(0)}}$, $\eta^{(1)} = \mathbf{g}_2^{\hat{(0)}} \cdot \boldsymbol{\zeta}_1^{\hat{(0)}}$
5: Randomly choose $e \leftarrow \mathcal{C}^*$ and $S$
6: Calculate the $\psi \leftarrow \mathbf{g}^{\hat{(1)}} \cdot \boldsymbol{\zeta}^{\hat{(1)}} - S \cdot e - \xi^{(1)} \cdot {c^{(1)}}^2 - \eta^{(1)} \cdot {c^{(1)}}^{-2}$
7: return the tuple $(\boldsymbol{\zeta}^{\hat{(1)}}, \mathbf{g}^{\hat{(1)}}, \psi, \xi^{(1)}, \eta^{(1)})$
```

5.3 Unforgeability

Theorem 2 *In the random oracle model, if the our scheme satisfies the soundness, then we say our scheme satisfies unforgeability.*

We assume that there exists a PPT adversary $\mathcal{A}$ who tries to break the unforgeability of our scheme and a challenger $\mathcal{C}$ is able to respond to queries of $\mathcal{A}$. We introduce four queries which can be used to prove the unforgeability.

- **Registration Query**: $\mathcal{A}$ queries the public key pk_i of user i, $\mathcal{C}$ calls the Ring-KeyGen algorithm and returns the corresponding public key to the $\mathcal{A}$, which can be performed at most N times.
- **Corruption Query**: $\mathcal{A}$ randomly choose one public key pk_i and queries the corresponding secret key sk_i. $\mathcal{C}$ returns the secret key sk_i if $i \neq j$. Otherwise, the query is aborted.
- **Signing Query**: $\mathcal{A}$ queries the signature which was signed by the signer i. If $i \neq j$, $\mathcal{C}$ calls the Ring-Sign algorithm to obtain the ring signature. If $i = j$, the secret key is missing, but this query cannot be aborted. In this situation, $\mathcal{C}$ needs to run the simulator $\mathcal{S}$ to get the signature.
- **Random Oracle Query**: We model the hash function $\mathcal{H}$ to the random oracle. If the input has been queried before, the corresponding value is output. If the input has never been queried, the output is calculated and recorded.

We use the simulator $\mathcal{S}$ and the extractor $\mathcal{E}$ to prove soundness, i.e. unforgeability. The function of extractor $\mathcal{E}$ is to extract the knowledge, i.e., the secret key, which is shown in Algorithm 7. In addition, a ring signature can be generated without a secret key by the simulator $\mathcal{S}$ shown in Algorithm 6.

Algorithm 7 Extractor $\mathcal{E}$

```
1: Prover randomly choose $\hat{\boldsymbol{v}}$ and send to the extractor
2: Extractor randomly choose the challenge $e'$, send to prover
3: Prover compute the response $\hat{\boldsymbol{\zeta}'}$, send to the extractor
4: Extracotr rewind from step. 2
5: Extractor randomly choose the challenge $e''$ different with $e'$, send to prover
6: Prover compute the response again to have $\hat{\boldsymbol{\zeta}''}$, send to the extractor
7: return the secret key rewind by $(e', e'', \hat{\boldsymbol{\zeta}'}, \hat{\boldsymbol{\zeta}''})$
```

The secret key is required to be short and the challenge is irretrievable, we use the relax proof technique to norm check and relax the verification relationship.

Theorem 3 *If our proposed scheme satisfies the unforgeability, then we say our scheme satisfies linkability.*

Proof. We assume that there exists a PPT adversary $\mathcal{A}$ who tries to break the linkability of our scheme and a challenger $\mathcal{C}$ is able to respond to queries of $\mathcal{A}$. The queries are same as the unforgeability game. As a result, $\mathcal{A}$ outputs two tuples $(\hat{\boldsymbol{\zeta}^{(1)}}_1, \hat{\mathbf{g}^{(1)}}_1, \psi_1, \xi_1^{(1)}, \eta_1^{(1)}, L_1)$, $(\hat{\boldsymbol{\zeta}^{(1)}}_2, \hat{\mathbf{g}^{(1)}}_2, \psi_2, \xi_2^{(1)}, \eta_2^{(1)}, L_2)$.

We assume that $\mathcal{A}$ generates two ring signatures σ_1, σ_2 with non-negligible probability while only owns one secret key. Besides, both σ_1, σ_2 can pass the verification. Since our scheme satisfies the unforgeability, σ_1, σ_2 can pass Ring-Verify algorithm only if the $\mathcal{A}$ honestly generates two signatures.

When $\mathcal{A}$ generates the two signatures, we have two link labels $L_1 = \mathcal{H}_l(S, sk_1)$, $L_2 = \mathcal{H}_l(S, sk_2)$, respectively. Since $\mathcal{A}$ only has one secret key, then $sk_1 = sk_2$. Moreover, since the public parameter S is the same, we get $L_1 = L_2$. It shows that the two tuples of $\mathcal{A}$ verified by the Ring-Link algorithm will return 'link', which contradicts the assumption of the linkability game. Hence, the advantage of $\mathcal{A}$ is negligible and our scheme is linkable.

6 Performance Evaluation

In this section, we perform a comparative performance evaluation of our proposed $O(\log N)$ lattice-based linkable ring signature with existing lattice-based ring signature schemes [1, 4, 8].

Firstly, we present the theoretical analysis of the public key size, secret key size, and signature size comparison between our scheme and existing schemes [1, 4, 8], as shown in Table 1. Specifically, the sizes of our scheme's public key and secret key are similar to schemes [1, 4, 8], while the signature size of our scheme is superior to the other three schemes [1, 4, 8].

Table 1. Comparison of the Public Key, Secret Key, and Signature Sizes.

Schemes	Public Key	Secret Key	Signature
Scheme [8]	$nm \log p$	$2nm \log p$	$2Nnm \log(mn^{1.5} \log n - \sqrt{n} \log n)$ $+2Nn + (m+1)n \log p$
Scheme [1]	$nm \log p$	$2nm \log p$	$2Nnm \log(mn^{1.5} \log n - \sqrt{n} \log n) + 2n$
Scheme [4]	$nm \log p$	$2nm \log p$	$2Nnm \log(mn^{1.5} \log n - \sqrt{n} \log n) + 2n$
Ours	$nm \log p$	$2nm \log p$	$2nm \log(mn^{1.5} \log n - \sqrt{n} \log n)$ $+2n + nm \log p + 4n \log N$

Then, we show the comparison of the complexity of signature size and security requirement between ours and existing schemes [1, 4, 8], as Table 2. All schemes satisfy anonymity and unforgeability, while scheme [4] and ours satisfy linkability. More importantly, among schemes [1, 4, 8] and ours, only ours has $O(\log N)$ signature size, showing the advantage of ours in both signature size and security.

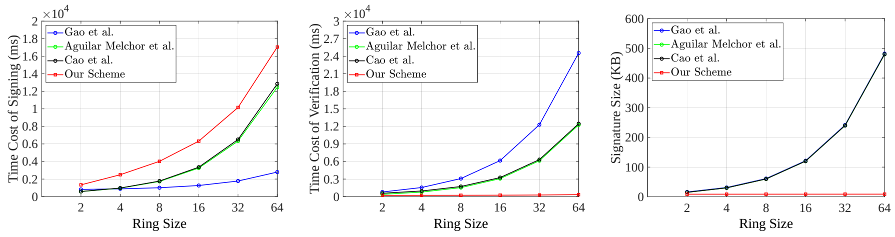
Table 2. Comparison of Signature Size and Security.

Schemes	Signature Size	Security		
		Anonymity	Unforgeability	Linkability
Scheme [8]	$O(N)$	✓	✓	×
Scheme [1]	$O(N)$	✓	✓	×
Scheme [4]	$O(N)$	✓	✓	✓
Ours	$O(\log N)$	✓	✓	✓

Finally, we set the parameters $n = 64$, $m = 30$, $\log(mn^{1.5} \log n - \sqrt{n} \log n) \approx 16$ and $\log p \approx 5$ to measure the performance of signature size and signature generation and verification time costs between our scheme and other existing schemes [1, 4, 8], shown in Fig. 3. It can be observed that the time cost for signature generation of our scheme is the highest among schemes [1, 4, 8], while the time cost for signature verification is generally lower than the other schemes [1,4,8]. Our scheme has the smallest signature size among all these schemes [1,4,8] and provides linkability. Therefore, the higher time cost for signature generation is an acceptable trade-off considering the signature size and security.

7 Conclusions

In this paper, we construct a logarithmic-size lattice-based linkable ring signature for cloud data management. To achieve this, we first generalize the Bulletproofs compression for the Σ-protocol and setting that under the lattices, to propose our generalized lattice-based Bulletproofs compression. Then, we implement our generalized lattice-based Bulletproofs compression to the existing lattice-based



(a) Signature Generation. (b) Signature Verification. (c) Signature Size.

Fig. 3. Comparison of Time Cost and Signature Size.

ring signature to propose our lattice-based ring signature protocol for one round compression. After that, utilizing the above protocol along with our designed link label and the Fiat–Shamir heuristic, we propose an efficient logarithmic-size lattice-based linkable ring signature. Our $O(\log N)$ lattice-based linkable ring signature is suitable for cloud data management, ensuring the cloud data owner's privacy and cloud data unforgeability and enabling checking of whether multiple cloud data are from the same user. Through the security analysis, our scheme satisfied correctness, anonymity, unforgeability and linkability. The performance evaluation demonstrates that our scheme outperforms existing schemes.

Acknowledgment

This work was supported by HKU-SCF FinTech Academy, Shenzhen-Hong Kong-Macao Science and Technology Plan Project (Category C Project: SGDX2021082 3103537030), Theme-based Research Scheme T35-710/20-R, the NSFC (No. 62302485), CAS Special Research Assistant Program and the Key Research Project of Chinese Academy of Sciences (No. RCJJ-145-24-21).

References

1. Aguilar Melchor, C., Bettaieb, S., Boyen, X., Fousse, L., Gaborit, P.: Adapting lyubashevsky's signature schemes to the ring signature setting. In: International Conference on Cryptology in Africa. pp. 1–25. Springer (2013)
2. Ajtai, M.: Generating hard instances of lattice problems. In: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing. pp. 99–108 (1996)
3. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: Short proofs for confidential transactions and more. In: 2018 IEEE symposium on security and privacy (SP). pp. 315–334 (2018)
4. Cao, C., You, L., Hu, G.: A novel linkable ring signature on ideal lattices. Entropy **25**(2), 237 (2023)
5. Chen, X., Xu, S., Gao, S., Guo, Y., Yiu, S.M., Xiao, B.: Fs-llrs: Lattice-based linkable ring signature with forward security for cloud-assisted electronic medical records. IEEE Transactions on Information Forensics and Security **19** (2024)

6. Fu, Z., Wang, Y., Sun, X., Zhang, X.: Semantic and secure search over encrypted outsourcing cloud based on bert. Frontiers of Computer Science **16**(2) (2022)
7. Gan, Q., Liu, J.K., Wang, X., Yuan, X., Sun, S.F., Huang, D., Zuo, C., Wang, J.: Verifiable searchable symmetric encryption for conjunctive keyword queries in cloud storage. Frontiers of Computer Science **16**(6), 166820 (2022)
8. Gao, W., Chen, L., Hu, Y., Newton, C.J., Wang, B., Chen, J.: Lattice-based deniable ring signatures. International Journal of Information Security **18** (2019)
9. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing. pp. 212–219 (1996)
10. Guo, Y., Xi, Y., Wang, H., Wang, M., Wang, C., Jia, X.: Fededb: Building a federated and encrypted data store via consortium blockchains. IEEE TKDE (2023)
11. Langlois, A., Stehlé, D.: Worst-case to average-case reductions for module lattices. Designs, Codes and Cryptography **75**(3), 565–599 (2015)
12. Li, X., Zhu, Y., Xu, R., Wang, J., Zhang, Y.: Indexing dynamic encrypted database in cloud for efficient secure k-nearest neighbor query. Frontiers of Computer Science **18**(1), 181803 (2024)
13. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM **21**(2), 120–126 (1978)
14. Schaper, J.: Cloud services. In: 4th IEEE International Conference on Digital Ecosystems and Technologies. pp. 91–91. IEEE (2010)
15. Shacham, H., Waters, B.: Efficient ring signatures without random oracles. In: International Workshop on Public Key Cryptography. pp. 166–180. Springer (2007)
16. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM review **41**(2), 303–332 (1999)
17. Tian, M., Zhang, Y., Zhu, Y., Wang, L., Xiang, Y.: Divrs: Data integrity verification based on ring signature in cloud storage. Computers & Security **124** (2023)
18. Wang, M., Guo, Y., Zhang, C., Wang, C., Huang, H., Jia, X.: Medshare: A privacy-preserving medical data sharing system by using blockchain. IEEE Transactions on Services Computing **16**(1), 438–451 (2021)
19. Wang, M., Miao, Y., Guo, Y., Huang, H., Wang, C., Jia, X.: Aesm2: Attribute-based encrypted search for multi-owner and multi-user distributed systems. IEEE Transactions on Parallel and Distributed Systems (2022)
20. Xu, S., Cao, Y., Chen, X., Guo, Y., Yang, Y., Guo, F., Yiu, S.M.: Post-quantum searchable encryption supporting user-authorization for outsourced data management. In: Proceedings of the 33rd ACM International Conference on Information and Knowledge Management. pp. 2702–2711 (2024)
21. Xu, S., Cao, Y., Chen, X., Zhao, Y., Yiu, S.M.: Post-quantum public-key authenticated searchable encryption with forward security: General construction, and applications. In: International Conference on Information Security and Cryptology. pp. 274–298. Springer (2023)
22. Xu, S., Chen, X., Guo, Y., Yang, Y., Wang, S., Yiu, S.M., Cheng, X.: Lattice-based forward secure multi-user authenticated searchable encryption for cloud storage systems. IEEE Transactions on Computers (2025)
23. Zhang, C., Xie, Q., Wang, M., Guo, Y., Jia, X.: Optimal compression for encrypted key-value store in cloud systems. IEEE Transactions on Computers **73**(3) (2024)
24. Zhang, J., Cui, J., Zhong, H., Gu, C., Liu, L.: Secure and efficient certificateless provable data possession for cloud-based data management systems. In: Database Systems for Advanced Applications: DASFAA 2021. pp. 71–87. Springer (2021)