Lattice-based Forward Secure Certificateless Encryption Scheme for Cloud Data Management

Shiyuan Xu[1], Xue Chen[1(✉)], Yue Wang[2], Tianrun Xu[3], Fangda Guo[4(✉)], Xiaoli Zhang[5(✉)], 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, smyiu@cs.hku.hk
[2] School of Artificial Intelligence, Beijing Normal University, Beijing, China
yuewang@mail.bnu.edu.cn
[3] Department of Automation, Tsinghua University, Beijing, China
xtr24@mails.tsinghua.edu.cn
[4] CAS Key Laboratory of AI Safety, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
guofangda@ict.ac.cn
[5] University of Science and Technology Beijing, Beijing, China
xiaoli.z@ustb.edu.cn

Abstract. Data encryption has been widely studied in cloud data management to protect data security. Most encryption schemes only works with fully trusted cloud services and have issues with key-escrow problems, which result in high storage overhead for key certificates and the security is relatively low. To address these concerns, certificateless encryption was formalized. However, in practical cloud applications, secret key exposure threat delegates a severe and realistic concern, potentially leading to privacy disclosure. In addition, most schemes are susceptible to resist quantum computing attacks. As such, research focusing on designing a forward secure certificateless encryption scheme while enjoying quantum-safety is far-reaching. In this paper, we propose FSCE, the first lattice-based forward secure certificateless encryption scheme, which can reduce the damage from key exposure in a quantum setting. We employ the lattice basis sampling algorithms to construct the certificateless encryption. Moreover, unlike former schemes, FSCE utilizes lattice basis extension algorithm together with binary tree structure to achieve one-way secret key evolution, allowing users to update their secret keys periodically. Furthermore, our scheme is proven to be secure in IND-CPA under two types of adversaries. Comprehensive experimental evaluations indicate that FSCE is effective and practical for cloud data management.

Keywords: Cloud data management · Certificateless encryption · Forward security · Data security and privacy · Lattice.

1 Introduction

Cloud computing has emerged as an attractive paradigm for big data applications, offering numerous well-known benefits such as flexible and convenient

storage, cost-effective performance, and rapid application deployment [2, 12, 19, 22, 33]. Meanwhile, concerns regard to cloud data security and privacy continue to obstruct its adoption in real-world applications [20, 24, 29]. To protect cloud data management security, data encryption appears to be a straightforward solution [10, 13, 14, 21]. Unfortunately, managing and storing public key certificates is cumbersome and significantly increases the communication overhead [18, 30]. Although leveraging identity-based encryption (IBE) can reduce the communication overhead of public keys, it brings up the key-escrow problem, threatening system security [25]. As such, it is crucial to safeguard cloud data security while minimizing communication costs.

Al-Riyami and Paterson formalized the concept of certificateless public key encryption [1], addressing the challenge of managing public key certificates without relying on a trusted third party and avoiding key escrow issues. Unlike traditional public key encryption, this approach introduces a semi-honest server with a master secret key, which is solely responsible for generating partial secret keys for users [31]. However, with the advancement of quantum computers [8], classical cryptographic primitives based on discrete logarithm (DL) have become vulnerable to quantum computing attacks, thereby compromising the security of cloud data [3,27]. To bridge the gap, Jiang et al. developed the first post-quantum certificateless encryption primitive [16]. Since then, numerous certificateless encryption schemes have been proposed [5, 17, 28].

In the aforementioned schemes, their secret keys are constant, so that if the user's secret key is leaked due to insufficient storage or malicious behavior of an adversary, the security of the cloud data may be endangered [7, 15]. The exposure of the secret key poses serious security threat undoubtedly, and it is easy to happen in the cloud server. To address this concern, several scholars introduced the notation of forward security into cryptographic primitives, where the secret key is updated periodically [6]. In this way, even if a secret key is compromised in one period, its security of other periods remains intact [26]. To the best of our knowledge, none of the existing certificateless encryption schemes can resist the secret key leakage problem in a quantum setting.

In this work, we propose FSCE, a lattice-based forward secure certificateless encryption scheme for cloud data management. Our solution eliminates the need to rely on fully trusted third parties for key generation and management, while we perfectly mitigate the risk of secret key leakage to enhance data security. The primary difference between traditional encryption and certificateless encryption lies in the method of key generation. In certificateless encryption, the user's key comprises two components: a partial secret key generated by an authority and a key generated by the user. Specifically, we employ the lattice basis sampling algorithms to generate both the master secret key and the partial secret key, which serves as a cornerstone towards to our design. Different from previous schemes [32, 34], our approach integrates the lattice basis extension algorithm with a binary tree structure to implement a one-way secret key update mechanism. Additionally, we incorporate the minimal cover set technique to facilitate periodic secret key updates. In essence, our scheme ensures that even if the cur-

rent secret key has been compromised, an adversary remains unable to decrypt the ciphertext. Our contributions can be summarized as follows:

- We propose FSCE, the first lattice-based forward-secure certificateless encryption framework for cloud data management. Our solution eliminates the reliance on trusted third parties for user key generation and resists to the secret key leakage attacks in the context of quantum security.
- We formalize and prove the IND-CPA security (with two types of adversaries) of FSCE by reducing to the Learning with Errors (LWE) hardness. Our design resists both external and internal attacks, ensuring a high level of security for cloud computing applications.
- Through comprehensive performance evaluation results, our FSCE scheme outperforms existing lattice-based schemes in communication overhead. Our scheme excels in encryption cost compared to others. The decryption cost of our scheme is better than most prior arts and slightly higher than Li et al. [17], which is deemed acceptable due to our superior security features.

2 Building Blocks

We now introduce some fundamental knowledge. We adopt lowercase bold letters to represent vectors (e.g. $\mathbf{e}$) and uppercase bold letters to represent matrices (e.g. $\mathbf{E}$). We adopt $[\mathbf{A}|\mathbf{B}]$ to denote the concatenation of matrices $\mathbf{A}$ with $\mathbf{B}$, and '$\leftarrow$' to denote sampling values. Given a matrix $\mathbf{M} = (\mathbf{m}_1, \mathbf{m}_2, \cdots, \mathbf{m}_m)$ composing m linearly independent vectors, we define a lattice Λ as: $\Lambda = \Lambda(\mathbf{M}) = \{x_1 \cdot \mathbf{m}_1 + x_2 \cdot \mathbf{m}_2 + \cdots + x_m \cdot \mathbf{m}_m | x_i \in \mathbb{Z}, i \in [m]\}$, where $\mathbf{M}$ is a basis of Λ.

Lemma 1 *[11] Given three positive integers n, m, and q. A probabilistic polynomial time (PPT) algorithm* $\mathsf{TrapGen}(n, m, q)$ *calculates a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and its basis $\mathbf{T_A} \in \mathbb{Z}_q^{m \times m}$ for $\Lambda_q^{\perp}(\mathbf{A})$, where $\mathbf{A}$ is statistically close to uniform distribution on $\mathbb{Z}^{n \times m}$, and $\|\widetilde{\mathbf{T_A}}\| \leq m\omega(\sqrt{\log m})$. A PPT algorithm* $\mathsf{SamplePre}(\mathbf{A}, \mathbf{T_A}, \mathbf{u}, \sigma)$ *calculates a vector $\mathbf{e} \in \mathbb{Z}_q^m$, statistically close to $\mathcal{D}_{\Lambda_q^u(\mathbf{A}),\sigma}$,* s.t. $\mathbf{A} \cdot \mathbf{e} = \mathbf{u} \bmod q$, *where $\sigma \leq \|\widetilde{\mathbf{T_A}}\| \cdot \omega(\sqrt{\log m})$.*

Lemma 2 *[4] Given two matrices $\mathbf{A}_1 \in \mathbb{Z}_q^{n \times m_1}, \mathbf{A}_2 \in \mathbb{Z}_q^{n \times m_2}$, and a basis $\mathbf{S}_1 \in \mathbb{Z}_q^{m_1 \times m_1}$ of $\Lambda_q^{\perp}(\mathbf{A}_1)$, the* $\mathsf{ExtBasis}(\mathbf{A}, \mathbf{S}_1)$ *algorithm calculates a matrix $\mathbf{S}$ of $\Lambda_q^{\perp}(\mathbf{A}) \subseteq \mathbb{Z}_q^{m_1 \times m}$ as its basis, where $m = m_1 + m_2$, $\mathbf{A} = \mathbf{A}_1 || \mathbf{A}_2$, and $\|\widetilde{\mathbf{S}_1}\| = \|\widetilde{\mathbf{S}}\|$.*

3 Problem Formulation

3.1 System Architecture

The system architecture of our scheme is depicted in Fig. 1, which consists of four entities: cloud data owner, cloud data user, cloud server, and authority. The specific workflow of each entity is as follows.

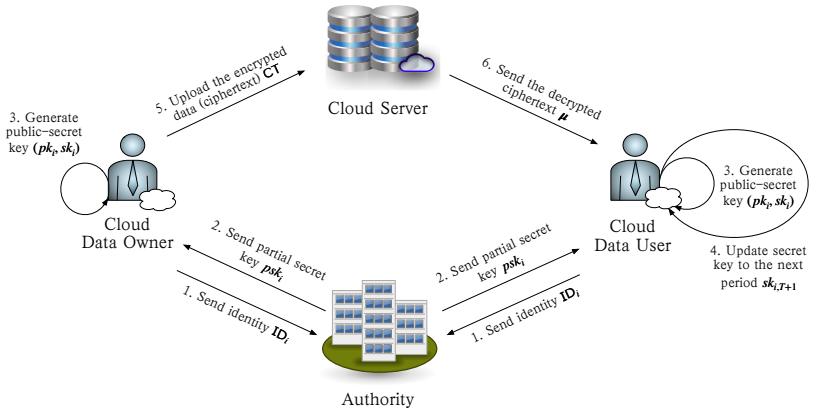


Fig. 1. System Architecture of FSCE.

- **Cloud data owner**: Each data owner has a collection of cloud data records $\boldsymbol{\mu}$ and generates its public-secret keys (pk_i, sk_i) through its identity ID_i. It encrypts the cloud data through public key pk_i, and then uploads the ciphertext CT to the cloud server for storage. It is considered as *fully trusted*.
- **Cloud data user**: Each data user generates its own public-secret keys (pk_i, sk_i) according to its identity ID_i. It could update its secret key $sk_{i,T}$ from the time period T to the next period $T+1$ as $sk_{i,T+1}$. Finally, a data user downloads the ciphertext CT from the cloud server, and decrypts it by the secret key to get the cloud data $\boldsymbol{\mu}$. It can be assumed as *malicious*.
- **Cloud server**: The server stores the the encrypted cloud data CT from the data owner. Upon receiving access requests from data user, it returns the ciphertext. It is considered as *honest-but-curious*.
- **Authority**: The authority serves as a system manager. It calculates the public parameter pp and master secret key MSK. Note that the master secret key is kept confidential. Furthermore, it generates the partial secret keys psk_i based on the identity ID_i and then transmits psk_i to the data owner/data user confidentially. It can be assumed as *honest-but-curious*.

3.2 Formal Definition

Our proposed scheme incorporates six algorithms: Setup, $\mathsf{PskExtract}$, KeyGen, $\mathsf{SkUpdate}$, $\mathsf{DataEnc}$, $\mathsf{CiphDec}$. The syntax of these algorithms are as follows.

- $\mathsf{Setup}(\lambda, d)$: Given a security parameter λ and a depth d of a binary tree, this algorithm returns a public parameter pp and a master secret key MSK.
- $\mathsf{PskExtract}(pp, \mathsf{MSK}, \mathsf{ID}_i)$: Given a public parameter pp, a master secret key MSK, and a user identity ID_i, this algorithm returns a partial secret key psk_i.
- $\mathsf{KeyGen}(pp, \mathsf{ID}_i, psk_i)$: Given a public parameter pp, a user identity ID_i, and a partial secret key psk_i, this algorithm returns a public key pk_i, and a secret key sk_i for ID_i.

- $\mathsf{SkUpdate}(pp, \mathsf{ID}_i, d, sk_{i,T})$: Given a public parameter pp, a user identity ID_i, a depth d of a binary tree, and a secret key $sk_{i,T}$ on the time period T, this algorithm returns a updated secret key $sk_{i,T+1}$ on the timer period $T+1$.
- $\mathsf{DataEnc}(pp, \boldsymbol{\mu}, \mathsf{ID}_i, pk_i)$: Given a public parameter pp, cloud data $\boldsymbol{\mu}$, a user identity ID_i, and a public key pk_i, this algorithm returns a ciphertext CT.
- $\mathsf{CiphDec}(pp, \mathsf{ID}_i, \mathsf{CT}, sk_{i,T})$: Given a public parameter pp, a user identity ID_i, a ciphertext CT, and a secret key $sk_{i,T}$ on the time period T, this algorithm returns the cloud data $\boldsymbol{\mu}$.

4 The Design of FSCE

4.1 Cloud System Initialization

The cloud system initialization stage of our scheme is conducted by the authority, which takes a security parameter λ and binary tree depth d as input and then sets several parameters $\sigma = m\omega(\sqrt{\log n})$, $m = \lceil 6n \log q \rceil$, $\alpha < \frac{1}{82nm^{1.5}\sigma \log q}$. Then it generates the public parameter pp and delivers it to other entities. It also calculates the master secret key MSK and keeps it in secret. The detailed algorithm (Setup) are elaborated in Algorithm 1.

Algorithm 1: Cloud System Initialization Algorithm

Input: Security parameter λ, and depth d of a binary tree.
Output: Public parameter pp, and master secret key MSK.

1 Invoke the $\mathsf{TrapGen}(n, m, q)$ algorithm to calculate a matrix $\mathbf{A}_0 \in \mathbb{Z}_q^{n \times m}$ together with its basis $\mathbf{T}_{\mathbf{A}_0} \in \mathbb{Z}_q^{m \times m}$;
2 Define a hash function: $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_q^{n \times m}$;
3 Uniformly random select several matrices: $\mathbf{M}_1^{(0)}, \mathbf{M}_1^{(1)}, \mathbf{M}_2^{(0)}, \mathbf{M}_2^{(1)}, \cdots, \mathbf{M}_d^{(0)}, \mathbf{M}_d^{(1)} \in \mathbb{Z}_q^{n \times m}$;
4 Set the public parameter $pp = (\mathbf{A}_0, \mathcal{H}, \mathbf{M}_1^{(0)}, \mathbf{M}_1^{(1)}, \mathbf{M}_2^{(0)}, \mathbf{M}_2^{(1)}, \cdots, \mathbf{M}_d^{(0)}, \mathbf{M}_d^{(1)} \in \mathbb{Z}_q^{n \times m})$;
5 Set the master secret key $\mathsf{MSK} = \mathbf{T}_{\mathbf{A}_0}$;
6 Return pp and MSK.

4.2 User Registration

There are two steps in the user registration phase, including user partial secret key extraction ($\mathsf{PskExtract}$) and user key generation (KeyGen). As illustrated in Algorithm 2, the authority invokes the $\mathsf{SamplePre}$ algorithm to compute the partial secret key psk_i for both data owner and data user. In certificateless cryptographic primitives, both data owner and data user calculates the public key pk_i and secret key sk_i by itself. It leverages the $\mathsf{TrapGen}$ algorithm to obtain a matrix $\mathbf{M}_i$ and its basis $\mathbf{T}_{\mathbf{M}_i}$. Finally, the public key pk_i is a combination of $\mathbf{M}_i$ and a hash value p_i, while the secret key sk_i is a combination of $\mathbf{T}_{\mathbf{M}_i}$ and partial secret key psk_i, as described in Algorithm 3.

Algorithm 2: User Partial Secret Key Extraction Algorithm

Input: Public parameter pp, master secret key MSK, and user identity ID_i.
Output: Partial secret key psk_i of ID_i.

1 Invoke the $\mathsf{SamplePre}(\mathbf{A}_0, \mathbf{T}_{\mathbf{A}_0}, \mathcal{H}(\mathsf{ID}_i), \sigma)$ algorithm to calculate a sample $\mathbf{E}_i$, where $\mathbf{A}_0 \cdot \mathbf{E}_i = \mathcal{H}(\mathsf{ID}_i)$;
2 Set the partial secret key $psk_i = \mathbf{E}_i \in \mathbb{Z}_q^{m \times m}$ for the user with identity ID_i;
3 Return psk_i.

Algorithm 3: User Key Generation Algorithm

Input: Public parameter pp, user identity ID_i, and partial secret key psk_i.
Output: Public key pk_i, and secret key sk_i of ID_i.

1 Invoke the $\mathsf{TrapGen}(n, m, q)$ algorithm to calculate a matrix $\mathbf{M}_i \in \mathbb{Z}_q^{n \times m}$ together with its basis $\mathbf{T}_{\mathbf{M}_i} \in \mathbb{Z}_q^{m \times m}$;
2 Set a parameter $p_i = \mathcal{H}(\mathbf{M}_i \| \mathsf{ID}_i)$;
3 Calculate a public key $pk_i = (\mathbf{M}_i, p_i)$ for the user with identity ID_i;
4 Calculate $sk_i = (\mathbf{T}_{\mathbf{M}_i}, psk_i)$ for the user with identity ID_i as its secret key;
5 Return pk_i and sk_i.

4.3 User Secret Key Update

In our design, the secret keys of cloud data users can be updated periodically, to address the secret key leakage problems and thereby achieve forward security. A data user ID_i takes a public parameter pp and its secret key $sk_{i,T}$ on the current time period T into the $\mathsf{SkUpdate}$ algorithm. Then, it performs the following procedures as specified in Algorithm 4 to update the secret key to the next time period $T + 1$. As depicted in Fig. 2, it describes the time period expression of a binary tree with $d = 3$. We show the examples of $\mathcal{L}(T)$ when the depth is three as: $\mathcal{L}(0)$ is the root node, $\mathcal{L}(1) = \{001, 01, 1\}$, $\mathcal{L}(2) = \{01, 1\}$, $\mathcal{L}(3) = \{011, 1\}$, $\mathcal{L}(4) = \{1\}$, $\mathcal{L}(5) = \{101, 11\}$, $\mathcal{L}(6) = \{11\}$, and $\mathcal{L}(7) = \{111\}$.

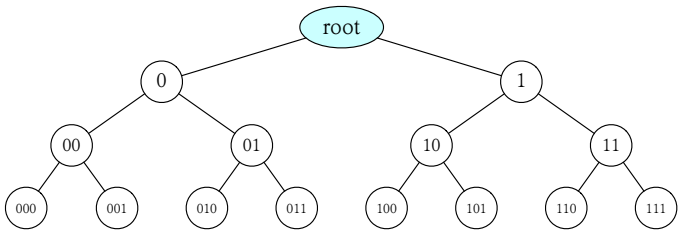


Fig. 2. Binary Tree with Depth $d = 3$ Time Period Expression.

4.4 Cloud Data Encryption

We now illustrate the cloud data encryption procedure ($\mathsf{DataEnc}$) of our scheme, which is conducted by the cloud data owner. The concrete steps are described in Algorithm 5. After a cloud data owner ID_i inputs the public parameter pp, the cloud data $\boldsymbol{\mu}$ together with its public key pk_i, it initially checks the validity of the public key and then randomly selects several matrices to encrypt the cloud data. The ciphertext CT includes two elements: CT_1 and CT_2.

Algorithm 4: User Secret Key Update Algorithm

Input: Public parameter pp, cloud data user identity ID_i, depth d of a binary tree, and secret key $sk_{i,T}$ on the time period T.

Output: Updated secret key $sk_{i,T+1}$ on the time period $T+1$.

The time period T is represented in binary as $T = (T_1 T_2 \cdots T_j)$, where $T_j \in \{0,1\}$ and $i \in \{1,2,\cdots,d\}$;

Let $\Psi^{(j)} = (\psi_1 \psi_2 \cdots \psi_j) \in \mathcal{L}(T)$, where $\psi_j \in \{0,1\}$;

`/* ` $\mathcal{L}(T)$ `includes the ancestor of all leaves in` $\{T, \cdots, 2^d - 1\}$`, while excluding any ancestor of leaves in` $\{0, 1, \cdots, T-1\}$ `*/`

Define a matrix $\mathbf{M}_{\Psi^{(j)}} = [\mathbf{A}_0 | \mathbf{M}_1^{\psi_1} | \mathbf{M}_2^{\psi_2} | \cdots | \mathbf{M}_j^{\psi_j}] \in \mathbb{Z}_q^{n \times (1+d)m}$ as the corresponding matrix of $\Psi^{(j)}$;

Define $\mathbf{T}_{\Psi^{(j)}}$ be a basis of node $\mathbf{M}_{\Psi^{(j)}}$ in the binary tree;

Update the secret key element $sk_{i,T}$ on the time period T to $sk_{i,T+1}$ on the next period $T+1$ based on the following two methods:

`/* Through any ancestor's basis or use the initial (root node) secret key` $\mathbf{T}_{\mathbf{A}_0}$ `*/`

if use any ancestor's basis $\mathbf{T}_{\Psi^{(h)}}$ $(h < j)$ **then**

  Invoke the $\mathsf{ExtBasis}(\mathbf{M}_{\Psi^{(j)}}, \mathbf{T}_{\Psi^{(h)}})$ algorithm to calculate a new basis $\mathbf{T}_{\Psi^{(j)}}$, where $\Psi^{(h)} = (\psi_1 \psi_2 \cdots \psi_h \cdots \psi_j)$;

else

  Invoke the $\mathsf{ExtBasis}(\mathbf{M}_{\Psi^{(j)}}, \mathbf{T}_{\mathbf{A}_0})$ algorithm to calculate a new basis $\mathbf{T}_{\Psi^{(j)}}$;

end

Determine the minimal cover set $\mathcal{L}(T+1)$ by computing all basis of nodes in $\mathcal{L}(T+1) \backslash \mathcal{L}(T)$ and deleting basis of nodes in $\mathcal{L}(T) \backslash \mathcal{L}(T+1)$;

Set the update secret key $sk_{i,T+1} = (\mathbf{T}_{\Psi^{(j)}}, psk_i)$ on the time period $T+1$ for the user with identity ID_i;

Return $sk_{i,T+1}$.

4.5 Ciphertext Decryption

The last procedure of our design is the ciphertext decryption phase ($\mathsf{CiphDec}$), as mentioned in the Algorithm 6 concretely. This algorithm is executed by the cloud data user, with public key pp, ciphertext CT, and secret key $sk_{i,T}$ on the timer period T as input. It computes a matrix for decryption and check if the equation $\|2\mathbf{T}_{\mathbf{M}_i}^{\top} \cdot \mathbf{P} + \mathbf{T}_{\mathbf{M}_i}^{\top} \cdot \boldsymbol{\mu}\|_{\infty} < \frac{q}{2}$ holds. Eventually, if the condition is met, then it returns the cloud data $\boldsymbol{\mu}$; Otherwise, it returns $\perp$.

Algorithm 5: Cloud Data Encryption Algorithm

Input: Public parameter pp, cloud data $\boldsymbol{\mu}$, cloud data owner identity ID_i, and public key pk_i.
Output: Ciphertext CT.
`/* Verify the validity of a public key` pk_i `*/`

if the equation $p_i \stackrel{?}{=} \mathcal{H}(\mathbf{M}_i \| \mathsf{ID}_i)$ holds **then**
  Uniformly random select a matrix $\mathbf{Q} \stackrel{\$}{\leftarrow} \mathbb{Z}_q^{n \times m}$;
  Uniformly random select two matrices $\mathbf{E}_1, \mathbf{E}_2$ over the distribution $\chi^{m \times m}$;
  Calculates two elements of the ciphertext $\mathsf{CT}_1 = \mathbf{A}_0^\top \cdot \mathbf{Q} + 2\mathbf{E}_1 \in \mathbb{Z}_q^{m \times m}$ and $\mathsf{CT}_2 = \mathcal{H}(\mathsf{ID}_i)^\top \cdot \mathbf{Q} + \boldsymbol{\mu} + \mathbf{M}_i^\top \cdot \mathbf{Q} + 2\mathbf{E}_2 \in \mathbb{Z}_q^{m \times m}$, where $\boldsymbol{\mu} \in \mathbb{Z}_q^{m \times m}$ is the cloud data;
  Let the ciphertext $\mathsf{CT} = (\mathsf{CT}_1, \mathsf{CT}_2)$;
else
  Return $\perp$;
end
Return CT.

4.6 Correctness Analysis

We hereby analyze the correctness of our design. With regard to the ciphertext CT, a user with identity ID_i uses its secret key sk_i to compute the following equations:

$$\begin{aligned}
\mathbf{T}_{\mathbf{M}_i}^\top \cdot \mathbf{C} &= \mathbf{T}_{\mathbf{M}_i}^\top (\mathsf{CT}_2 - \mathbf{E}_i^\top \cdot \mathsf{CT}_1) \\
&= \mathbf{T}_{\mathbf{M}_i}^\top (\mathcal{H}(\mathsf{ID}_i)^\top \cdot \mathbf{Q} + \boldsymbol{\mu} + \mathbf{M}_i^\top \cdot \mathbf{Q} + 2\mathbf{E}_2 - 2\mathbf{A}_0^\top \cdot \mathbf{Q} - 2\mathbf{E}_1) \\
&= \mathbf{T}_{\mathbf{M}_i}^\top (2(\mathbf{E}_2 - \mathbf{E}_i^\top \cdot \mathbf{E}_1) + \mathbf{M}_i^\top \cdot \mathbf{Q} + \boldsymbol{\mu}) \\
&= \mathbf{T}_{\mathbf{M}_i}^\top (2\mathbf{P} + \mathbf{M}_i^\top \cdot \mathbf{Q} + \boldsymbol{\mu}) \\
&= 2\mathbf{T}_{\mathbf{M}_i}^\top \cdot \mathbf{P} + \mathbf{T}_{\mathbf{M}_i}^\top \cdot \mathbf{M}_i^\top \cdot \mathbf{Q} + \mathbf{T}_{\mathbf{M}_i}^\top \cdot \boldsymbol{\mu} \\
&= 2\mathbf{T}_{\mathbf{M}_i}^\top \cdot \mathbf{P} + \mathbf{T}_{\mathbf{M}_i}^\top \cdot \boldsymbol{\mu}.
\end{aligned}$$

If $\|2\mathbf{T}_{\mathbf{M}_i}^\top \cdot \mathbf{P} + \mathbf{T}_{\mathbf{M}_i}^\top \cdot \boldsymbol{\mu}\|_\infty < \frac{q}{2}$, then we have $(\mathbf{T}_{\mathbf{M}_i}^\top)^{-1} (\mathbf{T}_{\mathbf{M}_i}^\top \cdot \mathbf{C}) \bmod 2 = \boldsymbol{\mu}$.

5 Security Analysis

It is essential to demonstrate the security when designing a certificateless encryption scheme for the cloud data management [9,23]. We consider two types of adversaries $\mathcal{A}$ in our design, including Type-I ($\mathcal{A}_I$), and Type-II ($\mathcal{A}_{II}$). Specifically, $\mathcal{A}_I$ is an external attacker without knowing any information about the master secret key MSK, and it has the ability to replace any user's public key to the value in the public key list $pk_i \in \mathcal{PK}$. $\mathcal{A}_{II}$ is assumed to be a curious authority knowing the master secret key MSK, and it also has the ability to replace any user's public key to the value in the public key list $pk_i \in \mathcal{PK}$ except for the target user. Both of the adversaries are interacted with a challenger $\mathcal{C}$ to simulate the security experiment.

Algorithm 6: Ciphertext Decryption Algorithm

Input: Public parameter pp, cloud data user identity ID_i, ciphertext CT, and secret key $sk_{i,T}$ on the time period T.

Output: Cloud data $\boldsymbol{\mu}$.

Calculate a matrix $\mathbf{C} = \mathsf{CT}_2 - \mathbf{E}_i^{\top} \cdot \mathsf{CT}_1$;

Calculate $\boldsymbol{\mu} = (\mathbf{T}_{\mathbf{M}_i^{\top}})^{-1} \cdot (\mathbf{T}_{\mathbf{M}_i^{\top}} \cdot \mathbf{C}) \bmod 2$;

Define a matrix $\mathbf{P} = \mathbf{E}_2 - \mathbf{E}_i^{\top} \cdot \mathbf{E}_1$;

if the equation $\|2\mathbf{T}_{\mathbf{M}_i}^{\top} \cdot \mathbf{P} + \mathbf{T}_{\mathbf{M}_i}^{\top} \cdot \boldsymbol{\mu}\|_{\infty} \geq \frac{q}{2}$ holds **then**

  Return $\perp$;

else

  Return the cloud data $\boldsymbol{\mu}$.

end

5.1 Security Model

We formally define the IND-CPA security model of our scheme as follows.

- **Initialization.** To begin with, a challenger $\mathcal{C}$ executes the Setup algorithm with λ and d as input. $\mathcal{C}$ then transmits pp to an adversary $\mathcal{A}_I$ and $\mathcal{A}_{II}$, while it also sends MSK to $\mathcal{A}_{II}$.
- **Query Phase 1.** The adversary performs a polynomially bounded number of queries, and $\mathcal{C}$ returns the answer as follows.
  - Public key oracle $\mathcal{O}_{pk}$: $\mathcal{C}$ maintains a public key list $\mathcal{PK}$, which is empty initially. $\mathcal{A}$ enquires the public key with the identity ID_i as input, $\mathcal{C}$ answers as follows. $\mathcal{C}$ searches the pk_i in the list $\mathcal{PK}$. If it doesn't exist in the list, $\mathcal{C}$ invokes the $\mathsf{PskExtract}(pp, \mathsf{MSK}, \mathsf{ID}_i)$ and $\mathsf{KeyGen}(pp, \mathsf{ID}_i, psk_i)$ algorithms to calculate pk_i, and returns it to $\mathcal{A}$ while adds it in the list; Otherwise, it returns $pk_i \in \mathcal{PK}$ to $\mathcal{A}$.
  - Public key replace oracle $\mathcal{O}_{pkr}$: After $\mathcal{A}$ sends a new public key pk_i' for the user with identity ID_i to $\mathcal{C}$, $\mathcal{C}$ then updates pk_i to pk_i' in the list $\mathcal{PK}$.
  - Partial secret key oracle $\mathcal{O}_{psk}$: $\mathcal{C}$ maintains a partial secret key list $\mathcal{PSK}$, which is empty initially. $\mathcal{A}_I$ enquires the partial secret key with the identity ID_i as input, $\mathcal{C}$ answers as follows. $\mathcal{C}$ searches the psk_i in the list $\mathcal{PSK}$. If it doesn't exist in the list, $\mathcal{C}$ invokes the $\mathsf{PskExtract}(pp, \mathsf{MSK}, \mathsf{ID}_i)$ algorithm to calculate psk_i, and returns it to $\mathcal{A}_I$ while adds it in the list; Otherwise, it returns $psk_i \in \mathcal{PSK}$ to $\mathcal{A}_I$. Note that this query is not compatible with $\mathcal{A}_{II}$ as it can generate the partial secret key through MSK directly.
  - Secret key oracle $\mathcal{O}_{sk}$: $\mathcal{C}$ maintains a secret key list $\mathcal{SK}$, which is empty initially. $\mathcal{A}$ enquires the secret key with the identity ID_i as input, $\mathcal{C}$ answers as follows. $\mathcal{C}$ searches the sk_i in the list $\mathcal{SK}$. If it doesn't exist in the list, $\mathcal{C}$ invokes the $\mathsf{PskExtract}(pp, \mathsf{MSK}, \mathsf{ID}_i)$ and $\mathsf{KeyGen}(pp, \mathsf{ID}_i, psk_i)$ algorithms to calculate sk_i, and returns it to $\mathcal{A}$ while adds it in the list; Otherwise, it returns $sk_i \in \mathcal{SK}$ to $\mathcal{A}$.

  - Secret key update oracle $\mathcal{O}_{sku}$: After $\mathcal{A}$ sends a secret key $sk_{i,T}$ on the timer period T of the user with identity ID_i to $\mathcal{C}$, $\mathcal{C}$ updates $sk_{i,T}$ to the next period $sk_{i,T+1}$ unless the current period is the final period, and returns it to $\mathcal{A}$.
- **Challenge.** $\mathcal{A}$ sends two challenge cloud data $\boldsymbol{\mu}_0$ and $\boldsymbol{\mu}_1$ together with the challenge user's identity ID_i^* to $\mathcal{C}$. Then, $\mathcal{C}$ performs these procedures.
  - Type-I $\mathcal{A}_I$: If the public key associated with ID_i^* has been replaced with an invalid public key, or if it has been replaced with a valid public key and its corresponding partial secret key psk_i^* has been queried, $\mathcal{C}$ will abort the game. Otherwise, $\mathcal{C}$ randomly selects a bit $b \in \{0,1\}$ and executes $\mathsf{DataEnc}(pp, \boldsymbol{\mu}, \mathsf{ID}_i^*, pk_i^*)$ algorithm to calculate CT^*, and then returns it to $\mathcal{A}_I$.
  - Type-II $\mathcal{A}_{II}$: If the public key associated with ID_i^* has been replaced, or the secret key sk_i^* has been queried, $\mathcal{C}$ will abort the game. Otherwise, $\mathcal{C}$ randomly selects a bit $b \in \{0,1\}$ and executes $\mathsf{DataEnc}(pp, \boldsymbol{\mu}, \mathsf{ID}_i^*, pk_i^*)$ algorithm to calculate CT^*, and then returns it to $\mathcal{A}_{II}$.
- **Query Phase 2.** $\mathcal{A}$ continues to perform the queries as in Query Phase 1 with the following limitations: $\mathcal{A}_I$ is forbidden to query $\mathcal{O}_{psk}$ oracle and $\mathcal{A}_{II}$ is forbidden to query $\mathcal{O}_{sk}$ oracle with inputting ID_i^*.
- **Guess.** Ultimately, $\mathcal{A}$ outputs a guess bit $b' = \{0,1\}$ and it wins the game if $b' = b$. The advantage for $\mathcal{A}$ to attack the scheme can be defined as: $Adv_{\mathcal{A}}^{\text{IND-CPA}}(\lambda) = 2|\Pr[b = b'] - \frac{1}{2}|$.

Definition 1 (IND-CPA security) *Our lattice-based FSCE scheme is* IND-CPA secure, if for any PPT adversary $\mathcal{A}_I$ or $\mathcal{A}_{II}$, the advantage $Adv_{\mathcal{A}}^{\text{IND-CPA}}(\lambda)$ is negligible.

5.2 Security Proof

Theorem 1 *Our lattice-based* FSCE *scheme is* IND-CPA *secure assuming the* LWE *hardness holds. For a polynomial* Type-I *adversary $\mathcal{A}_I$, if $\mathcal{A}_I$ has the ability to win the game by performing limited public key and partial secret key queries, then there exists an algorithm $\mathcal{B}$ which can solve the* LWE *assumption.*

Proof. **Initialization.** $\mathcal{B}$ calculates MSK and pp, and sends pp to $\mathcal{A}_I$.

Query Phase 1. $\mathcal{A}_I$ performs a polynomially bounded number of queries, and $\mathcal{B}$ responds the answer while records the data in the corresponding lists.

- Public key oracle $\mathcal{O}_{pk}$: $\mathcal{A}_I$ inputs ID_i, and $\mathcal{B}$ returns pk_i if it exists in $\mathcal{PK}$; Otherwise, $\mathcal{B}$ transmits ID_i to $\mathcal{C}$. If $\mathsf{ID}_i = \mathsf{ID}_t$, $\mathcal{C}$ answers a tuple $(\mathsf{ID}_i^*, \mathcal{H}(\mathsf{ID}_i^*), \perp)$ to $\mathcal{B}$; Otherwise, $\mathcal{C}$ answers a tuple $(\mathsf{ID}_i, \mathcal{H}(\mathsf{ID}_i), \mathbf{E}_i)$ to $\mathcal{B}$. Then, $\mathcal{B}$ invokes $\mathsf{TrapGen}(n, m, q)$ to calculate $(\mathbf{M}_i, \mathbf{T}_{\mathbf{M}_i})$, and records the tuple $(\mathsf{ID}_i, \mathcal{H}(\mathsf{ID}_i), \perp, \mathbf{M}_i, \mathbf{T}_{\mathbf{M}_i})$ or $(\mathsf{ID}_i, \mathcal{H}(\mathsf{ID}_i), \mathbf{E}_i, \mathbf{M}_i, \mathbf{T}_{\mathbf{M}_i})$ in the list $\mathcal{PK}$. Eventually, it returns $(\mathsf{ID}_i, \mathcal{H}(\mathsf{ID}_i), \mathbf{M}_i)$ to $\mathcal{A}_I$.
- Public key replace oracle $\mathcal{O}_{pkr}$: Upon received a new public key $\mathbf{M}_i'$ of the user ID_i from $\mathcal{A}_I$, $\mathcal{B}$ replaces $(\mathsf{ID}_i, \mathcal{H}(\mathsf{ID}_i), \mathbf{E}_i, \mathbf{M}_i, \mathbf{T}_{\mathbf{M}_i})$ to $(\mathsf{ID}_i, \mathcal{H}(\mathsf{ID}_i), \mathbf{E}_i, \mathbf{M}_i', \perp)$ in the list $\mathcal{PK}$.

- Partial secret key oracle $\mathcal{O}_{psk}$: After $\mathcal{A}_I$ inputting ID_i, $\mathcal{B}$ searches the list $\mathcal{PK}$ and returns $\mathbf{E}_i$ if $\mathsf{ID}_i \neq \mathsf{ID}_i^*$; Otherwise, $\mathcal{B}$ returns $\perp$.
- Secret key oracle $\mathcal{O}_{sk}$: $\mathcal{A}_I$ sends ID_i to $\mathcal{B}$. If its public key has been replaced, $\mathcal{B}$ returns $\perp$; Otherwise, $\mathcal{B}$ returns $\mathbf{T}_{\mathbf{M}_i}$.
- Secret key update oracle $\mathcal{O}_{sku}$: Upon received the $sk_{i,T}$ update query from $\mathcal{A}_I$, $\mathcal{B}$ updates $sk_{i,T}$ to $sk_{i,T+1}$ at the next period unless it is the last period.

Challenge. $\mathcal{A}_I$ sends two challenge cloud data $\boldsymbol{\mu}_0$ and $\boldsymbol{\mu}_1$ and the challenge identity ID_i^* to $\mathcal{B}$. Then, $\mathcal{B}$ carries out steps as follows when $\mathsf{ID}_i^* = \mathsf{ID}_t^*$. If the public key $\mathbf{M}^*$ of ID_i^* has been replaced with a valid public key $\mathbf{M}^{*\prime}$: $\mathcal{B}$ sends $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$ to $\mathcal{C}$ and computes $(V_1, V_2 + \mathbf{M}^* \cdot \mathbf{S}^*)$, where V_1 and V_2 are ciphertexts and $\mathbf{S}^* \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$. Otherwise: $\mathcal{B}$ returns $\perp$.

Query Phase 2. This phase is the same as Query Phase 1.

Guess. $\mathcal{A}_I$ outputs a guess bit b' to $\mathcal{B}$ and $\mathcal{C}$. If $\mathcal{A}_I$ uses an invalid pk_i to query $\mathcal{O}_{pkr}$, then it will fail the game. In this way, we can assume that $\mathcal{A}_I$ replace it to a valid public key. The requirements to break this game are: When accessing the $\mathcal{O}_{psk}$ oracle, it has $\mathsf{ID}_i \neq \mathsf{ID}_i^*$ and $\mathsf{ID}_i^* = \mathsf{ID}_t^*$. As the IBE scheme has been proven secure in IND-CPA security under the LWE hardness, the advantage of $\mathcal{A}_I$ to break the FSCE scheme is negligible.

Theorem 2 *Our lattice-based* FSCE *scheme is* IND-CPA *secure assuming the* LWE *hardness holds. For a polynomial* Type-II *adversary* $\mathcal{A}_{II}$*, if* $\mathcal{A}_{II}$ *can win the game by performing limited public key, public key replace, and secret key queries, then there exists an algorithm* $\mathcal{B}$ *which can solve the* LWE *assumption.*

Proof. **Initialization.** $\mathcal{C}$ sends $\mathbf{M}^* \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$ to $\mathcal{B}$. $\mathcal{B}$ then invokes $\mathsf{TrapGen}(n, m, q)$ to calculate $(\mathbf{A}_0, \mathbf{T}_{\mathbf{A}_0})$ and sends them to $\mathcal{A}_{II}$.

Query Phase 1. $\mathcal{A}_{II}$ performs a polynomially bounded number of queries, and $\mathcal{B}$ responds the answer while records the data in the corresponding lists.

- Public key oracle $\mathcal{O}_{pk}$: $\mathcal{A}_{II}$ inputs ID_i, and $\mathcal{B}$ returns $(\mathbf{M}_i, \mathbf{T}_{\mathbf{M}_i})$ if it exists in $\mathcal{PK}$. If it does not exist in $\mathcal{PK}$, $\mathcal{C}$ sends $(\mathbf{M}_i, \mathbf{T}_{\mathbf{M}_i})$ to $\mathcal{B}$ if $\mathsf{ID}_i \neq \mathsf{ID}_t$; Otherwise, $\mathcal{B}$ sets $\mathbf{M}_i = \mathbf{M}^*$ and $\mathbf{T}_{\mathbf{M}_i} = \perp$. Eventually, $\mathcal{B}$ records $(\mathbf{M}_i, \mathbf{T}_{\mathbf{M}_i})$ in the list $\mathcal{PK}$.
- Public key replace oracle $\mathcal{O}_{pkr}$: Same as Theorem 1.
- Secret key oracle $\mathcal{O}_{sk}$: $\mathcal{A}_I$ sends ID_i to $\mathcal{B}$. If $i \neq t$, $\mathcal{B}$ searches in the list $\mathcal{SK}$ and returns $\mathbf{T}_{\mathbf{M}_i}$; Otherwise, $\mathcal{B}$ returns $\perp$.
- Secret key update oracle $\mathcal{O}_{sku}$: Same as Theorem 1.

Challenge. $\mathcal{A}_{II}$ sends two challenge cloud data $\boldsymbol{\mu}_0$ and $\boldsymbol{\mu}_1$ and the challenge identity ID_i^* to $\mathcal{B}$. Then, $\mathcal{B}$ carries out steps as follows if $\mathsf{ID}_i^* = \mathsf{ID}_t^*$. $\mathcal{B}$ searches the list to obtain $(\mathsf{ID}_i^*, \mathbf{M}^*, \perp)$, and executes the Algorithm 2 with inputting $(pp, \mathsf{MSK}, \mathsf{ID}_i^*)$ to calculate $\mathbf{E}^*$. After that, $\mathcal{B}$ sends $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$ to $\mathcal{C}$, and computes $(\mathbf{A}_0^\top \cdot \mathbf{S} + 2E_1, \mathcal{H}(\mathsf{ID}_t)^\top \cdot \mathbf{S} + 2E_2 + V)$, where V is a ciphertext, $\mathbf{S} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$, and $E_1, E_2 \xleftarrow{\$} \chi^{m\times m}$.

Query Phase 2. This phase is the same as Query Phase 1.

Guess. $\mathcal{A}_{II}$ outputs a guess bit b' to $\mathcal{B}$ and $\mathcal{C}$. The requirements to break this game are: When accessing $\mathcal{O}_{pkr}$ and $\mathcal{O}_{sk}$, it has $\mathsf{ID}_i \neq \mathsf{ID}_i^*$ and $\mathsf{ID}_i^* = \mathsf{ID}_t^*$. As the IBE scheme has been proven secure in IND-CPA security under the LWE hardness, the advantage of $\mathcal{A}_{II}$ to break the FSCE scheme is negligible.

6 Performance Evaluation and Comparison

In this sector, we conduct a comprehensive performance evaluation of our design and then give a thorough comparison with other state-of-the-art schemes [5], [17], [26]. We perform a comparative analysis of our scheme with others in both computational overhead and communication overhead. All experiments were conducted in a system environment with a processor of Apple M2 and memory of 16.0 GB. In general, we evaluate the overhead in three types parameter setting environments, i.e. $n = 64$, $n = 128$, $n = 256$. For example, when $n = 128$, we set security parameters $q = 2^{11}$, $m = 6n \log q = 8448$, $N = 2n \log q = 2816$, $M = (n + 1) \log q + 2n = 1675$, $\rho = 10$, $\kappa = 10$, the length of a keyword $\ell = 10$, and the depth of a binary tree $d = 3$.

In the computational cost analysis, we compare the time required for the encryption and decryption processes. Firstly, we provide a theoretical comparison of the computational overhead of our scheme and previous studies [5], [17], [26], as illustrated in Table 1. We denote T_{mul} as the multiplication operation, T_h as the hash function operation, T_{sl} as the SampleLeft algorithm, T_{sp} as the SamplePre algorithm. We observe that the encryption and decryption time costs of Xu et al.'s work [26] are significantly higher compared to others due to its searchable function. Therefore, we will exclude it from further comparison.

Table 1. Theoretical Comparison of Computation Overheads.

Schemes	Encryption	Decryption
Chen et al. [5]	$(M+N)nT_{mul} + (M+n)nT_{mul}$	$(N+n)nT_{mul}$
Li et al. [17]	$3nT_{mul}$	$3mT_{mul}$
Xu et al. [26]	$(\rho+1)T_h + (\kappa(m+n+1)$ $+\rho(\frac{(d+3)^2m^2}{4} + (d+3)nm$ $+2n+\ell+1))T_{mul}$	$2T_h + (\kappa(m+n+1)$ $+\ell)T_{mul} + T_{sl} + T_{sp}+$ $(d+3)\rho mT_M$
Ours	$3nT_{mul}$	$3mT_{mul}$

Then, we analyze and compare the ciphertext size and security requirements of our scheme with existing schemes [5], [17], [26]. As illustrated in Table 2, our scheme's ciphertext size is significantly smaller than Xu et al.'s work [26], slightly smaller than Chen et al.'s work [5], and equal to Li et al.'s work [17]. In terms of security analysis, our scheme additionally provides forward security, which makes it superior to the scheme proposed by Li et al. [17]. Although Xu et al.'s work [26] also provides forward security, it does not applicable the property

of IND-CPA. Similar to the comparison of computational cost, we will exclude Xu et al.'s work [26] from further comparison.

Table 2. Theoretical Comparison of Ciphertext Size and Security.

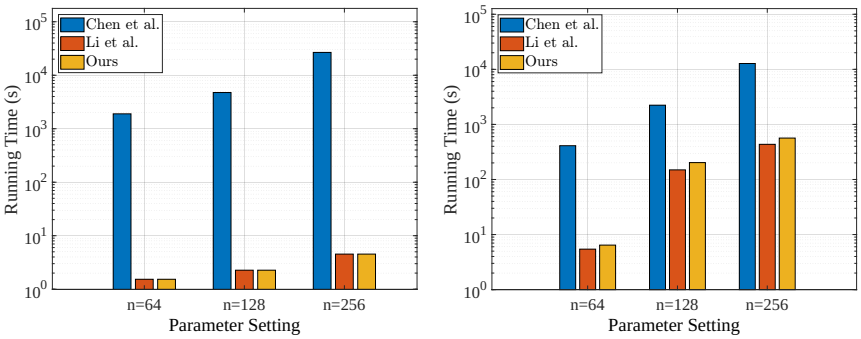
Schemes	Ciphertext Size	Security		
		Quantum Resistance	IND-CPA	Forward Secure
Chen et al. [5]	$2(n^{1.5}+n^2)$	✓	✓	×
Li et al. [17]	$2n^{1.5}$	✓	✓	×
Xu et al. [26]	$\rho(q+(d+3)mq+1)$	✓	∘	✓
Ours	$2n^{1.5}$	✓	✓	✓

Furthermore, we conducted a comparative experimental analysis of encryption time cost, decryption time cost, and ciphertext size across three different parameter settings. The time cost of the encryption process is illustrated in Fig. 3(a), where it is evident that our scheme is almost equivalent to Li et al. [17] and significantly lower than Chen et al. [5]. In the decryption time cost, as shown in Fig. 3(b), our scheme is slightly higher than Li et al. [17] but remains significantly lower than Chen et al. [5]. The comparison of ciphertext size, presented in Fig. 3(c), demonstrates that our scheme is comparable to Li et al. [17] and markedly lower than Chen et al. [5] across all three parameter settings.

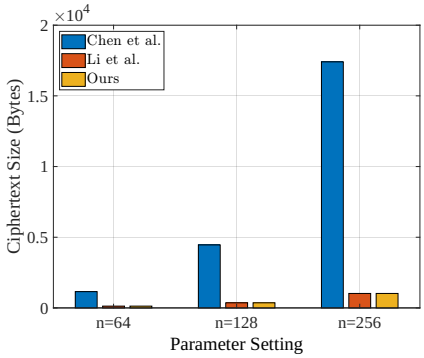
In summary, the time costs and ciphertext size of our scheme are significantly lower than Chen et al.'s scheme [5]. Compared to Li et al.'s scheme [17], our scheme has a slightly higher decryption time overhead. However, this is an acceptable trade-off, as our scheme additionally provides forward security, which Li et al.'s scheme [17] does not offer.

7 Conclusion

In this work, we present a lattice-based forward secure certificateless encryption scheme for privacy-preserving in the context of cloud data management. Our scheme is designed based on the lattice hardness and mitigates the threat of secret key exposure. Combining with several lattice basis sampling algorithms, we construct the fundamental certificateless encryption framework. Then, by leveraging the philosophy of the ExtBasis algorithm and binary tree structure, we facilitate the one-way secret key evolution, enabling periodic secret key updates by users. Rigorous analysis and experimental evaluations demonstrate that FSCE is secure and practical for cloud data management. For future work, it would be interesting to design a lattice-based forward secure certificateless encryption scheme supporting keyword search.



(a) Encryption Phase. (b) Decryption Phase.



(c) Ciphertext Size.

Fig. 3. Comparison of Algorithms Time Costs and Ciphertext Size.

Acknowledgment

This work was supported by HKU-SCF FinTech Academy, Shenzhen-Hong Kong-Macao Science and Technology Plan Project (Category C Project: SGDX20210823103537030), Theme-based Research Scheme T35-710/20-R, Zhejiang Provincial Natural Science Foundation of China under Grant No. LQ23F020019, the NSFC (No. 62302452, No. 62302485), CAS Special Research Assistant Program and the Key Research Project of Chinese Academy of Sciences (No. RCJJ-145-24-21).

References

1. Al-Riyami, S.S., Paterson, K.G.: Certificateless public key cryptography. In: International conference on the theory and application of cryptology and information security. pp. 452–473. Springer (2003)
2. Bai, Z., Wang, M., Guo, F., Guo, Y., Cai, C., Bie, R., Jia, X.: Secmdp: Towards privacy-preserving multimodal deep learning in end-edge-cloud. In: IEEE 40th International Conference on Data Engineering. pp. 1659–1670. IEEE (2024)

3. Cao, Y.B., Chen, X.B., He, Y.F., Liu, L.X., Che, Y.M., Wang, X., Xiao, K., Xu, G., Chen, S.Y.: A post-quantum cross-domain authentication scheme based on multi-chain architecture. Computers, Materials & Continua **78**(2) (2024)
4. Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai trees, or how to delegate a lattice basis. Journal of Cryptology **25**, 601–639 (2012)
5. Chen, H., Hu, Y., Lian, Z.Z., Jia, H.W.: Efficient certificateless encryption schemes from lattices (in chinese). Journal of Software **27**(11), 2884–2897 (2016)
6. 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**, 8875–8891 (2024)
7. Chen, X., Xu, S., He, Y., Cui, Y., He, J., Gao, S.: Lfs-as: lightweight forward secure aggregate signature for e-health scenarios. In: ICC 2022-IEEE International Conference on Communications. pp. 1239–1244. IEEE (2022)
8. Chen, X., Xu, S., Qin, T., Cui, Y., Gao, S., Kong, W.: Aq–abs: Anti-quantum attribute-based signature for emrs sharing with blockchain. In: IEEE Wireless Communications and Networking Conference. pp. 1176–1181. IEEE (2022)
9. Cui, N., Deng, Z., Li, M., Ma, Y., Cui, J., Zhong, H.: Authenticated ranked keyword search over encrypted data with strong privacy guarantee. In: International Conference on Database Systems for Advanced Applications. pp. 644–660. Springer (2023)
10. 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)
11. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Proceedings of the fortieth annual ACM symposium on Theory of computing. pp. 197–206 (2008)
12. Geoffrey, X.Y., Wu, Z., Kossmann, F., Li, T., Markakis, M., Ngom, A., Madden, S., Kraska, T.: Blueprinting the cloud: Unifying and automatically optimizing cloud data infrastructures with brad. Proceedings of the VLDB Endowment **17**(11), 3629–3643 (2024)
13. Guo, Y., Xi, Y., Wang, H., Wang, M., Wang, C., Jia, X.: Fededb: Building a federated and encrypted data store via consortium blockchains. IEEE Transactions on Knowledge and Data Engineering (2023)
14. Guo, Y., Xie, H., Wang, C., Jia, X.: Enabling privacy-preserving geographic range query in fog-enhanced iot services. IEEE Transactions on Dependable and Secure Computing (2021)
15. Guo, Y., Zhang, C., Wang, C., Jia, X.: Towards public verifiable and forward-privacy encrypted search by using blockchain. IEEE Transactions on Dependable and Secure Computing (2022)
16. Jiang, M., Hu, Y., Lei, H., Wang, B., Lai, Q.: Lattice-based certificateless encryption scheme. Frontiers of Computer Science **8**, 828–836 (2014)
17. Li, J., Yan, M., Peng, J., Huang, H., Abd El-Latif, A.A.: A lattice-based efficient certificateless public key encryption for big data security in clouds. Future Generation Computer Systems **158**, 255–266 (2024)
18. Li, W., Susilo, W., Xia, C., Huang, L., Guo, F., Wang, T.: Secure data integrity check based on verified public key encryption with equality test for multi-cloud storage. IEEE Transactions on Dependable and Secure Computing (2024)
19. 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)

20. Li, Y., Ma, J., Miao, Y., Liu, X., Jiang, Q.: Blockchain-based encrypted image storage and search in cloud computing. In: International Conference on Database Systems for Advanced Applications. pp. 413–421. Springer (2022)
21. Liu, L., Li, X., Au, M.H., Fan, Z., Meng, X.: Metadata privacy preservation for blockchain-based healthcare systems. In: International Conference on Database Systems for Advanced Applications. pp. 404–412. Springer (2022)
22. Sha, M., Cai, Y., Wang, S., Phan, L.T.X., Li, F., Tan, K.L.: Object-oriented unified encrypted memory management for heterogeneous memory architectures. Proceedings of the ACM on Management of Data **2**(3), 1–29 (2024)
23. Tian, J., Lu, Y., Li, J.: Lightweight searchable and equality-testable certificateless authenticated encryption for encrypted cloud data. IEEE Transactions on Mobile Computing (2024)
24. Wang, H., Jiang, T., Guo, Y., Guo, F., Bie, R., Jia, X.: Label noise correction for federated learning: A secure, efficient and reliable realization. In: 2024 IEEE 40th International Conference on Data Engineering (ICDE). pp. 3600–3612. IEEE (2024)
25. 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)
26. 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)
27. 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)
28. Xu, S., Chen, X., Guo, Y., Yiu, S.M., Gao, S., Xiao, B.: Efficient and secure post-quantum certificateless signcryption with linkability for iomt. IEEE Transactions on Information Forensics and Security (2024)
29. Xu, S., Chen, X., He, Y.: Evchain: An anonymous blockchain-based system for charging-connected electric vehicles. Tsinghua Science and Technology **26**(6), 845–856 (2021)
30. Xu, S., Chen, X., He, Y., Cao, Y., Gao, S.: Vmt: Secure vanets message transmission scheme with encryption and blockchain. In: International Conference on Wireless Algorithms, Systems, and Applications. pp. 244–257. Springer (2022)
31. Yang, N., Tang, C., He, D.: A lightweight certificateless multi-user matchmaking encryption for mobile devices: Enhancing security and performance. IEEE Transactions on Information Forensics and Security (2023)
32. Yang, X., Chen, X., Huang, J., Li, H., Huang, Q.: Fs-ibeks: Forward secure identity-based encryption with keyword search from lattice. Computer Standards & Interfaces **86**, 103732 (2023)
33. Zhang, L., Li, R., Ren, C., Di, S., Liu, J., Huang, J., Underwood, R., Grosset, P., Tao, D., Liang, X., et al.: Lcp: Enhancing scientific data management with lossy compression for particles. Proceedings of the ACM on Management of Data **3**(1), 1–27 (2025)
34. Zhang, X., Xu, C., Wang, H., Zhang, Y., Wang, S.: Fs-peks: Lattice-based forward secure public-key encryption with keyword search for cloud-assisted industrial internet of things. IEEE Transactions on Dependable and Secure Computing **18**(3), 1019–1032 (2021)