Certificateless Aggregate Signature Without Trapdoor for Cloud Storage

Yingjie Dong[1], Yongjian Liao[1(✉)], Zhishuo Zhang[1], and Wen Huang[2]

[1] School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu, China
liaoyj@uestc.edu.cn
[2] Sichuan University, Chengdu, China
wen@scu.edu.cn

Abstract. In this paper, we propose a lattice-based certificateless signature scheme and certificateless aggregate signature which can be used to verify a large number of signatures simultaneously efficiently. To improve the security of our scheme, we have implemented our signature algorithm without trusted third parties. The security of our proposed signature algorithm can be reduced to the SIS problem under the random oracle model. Moreover, our construction only needs matrix multiplication and rejection sampling operation, so the algorithm is naturally simple and efficient. Besides, we've done some experiments using the NTL library, indicating our scheme has less time overhead and storage overhead than other schemes. To our best knowledge, we propose the first certificateless aggregate signature without trapdoors.

Keywords: Certificateless · Aggregate Signature · Lattice

1 Introduction

Cloud data storage is currently a mainstream way of storing data due to its inherent properties of scalability, large storage space, strong flexibility, etc. Scholars have proposed various schemes to protect data integrity. Certificateless schemes [1,2] ensure the data integrity, but cannot efficiently verify integrity on large batches of files. Therefore, we propose an aggregate signature algorithm for cloud storage that can perfectly solve these issues. Meanwhile, to solve the problem of the untrustworthiness of the cloud, we remove the requirement of the trusted third party and construct our certificateless one.

Signature is widely applied in various scenarios. Considering the requirements of practical applications, we need to balance security and algorithm efficiency. There are various signature schemes under standard model [3–5], while they are less efficient than schemes under random oracle models [6]. Hence, to make sure that our scheme can be executed effectively, we construct our scheme under random oracle models.

Because of the natural superiority of anti-quantum attacks compared with other traditional hard assumptions, such as discrete logarithm, lattice-based

H. Duan et al. (Eds.): SecureComm 2023, LNICST 567, pp. 350–368, 2025.
https://doi.org/10.1007/978-3-031-64948-6_18

cryptography has been obtaining more and more research. Constructing the first lattice-based secure signature is not easy. In 2008, Gentry et al. [7] proposed a signature scheme based on the worst-case lattice-based hard problems. In 2014, Abdalla et al. [8] proposed a signature scheme based on the SVP problem in ideal lattices. To satisfy demands in different application scenarios, various signature algorithms with special properties have been proposed accordingly, for example, certificateless signature, aggregate signature, ring signature, blind signature, etc.

1.1 Related Work

Many scholars have conducted research on lattice signatures. In Gentry's scheme [7], the pivotal idea is that given a lattice $\boldsymbol{A}$ along with its basis $\boldsymbol{S}$ with short norm, sample lattice points from discrete Gaussian distribution. The main procedure is that by calling the algorithm $Samplepre$ with input $(\boldsymbol{A}, \boldsymbol{S})$ and $H(msg)$, output a short vector $\boldsymbol{v}$ satisfying $\boldsymbol{Av} = H(msg)$. Alwen and Peikert [9] proposed a new method to generate basis in 2009. Due to the complexity and high time overhead of the Trapdoor generation algorithm and pre-image sampling function, it is significant to build the lattice scheme without trapdoors. In 2012, Lyubashevsky [10] proposed a signature scheme without trapdoors by using rejection sampling, it sets $(\boldsymbol{A}, \boldsymbol{AS})$ as the public key and keeps $\boldsymbol{S}$ as the secret key. Its scheme only includes matrix multiplication and matrix addition which is more efficient than other schemes. To untie the connection between ciphertext and private key, it uses a core technique named rejection sampling which will be introduced later. In 2013, Ducas et al. [11] improved the signature scheme by sampling from bimodal Gaussian. In 2018, Lyubashevsky and Micciancio [12] proposed a lattice-based one-time signature that can be converted from certain types of linear collision-resistant hash functions.

To satisfy different practical needs, various signature schemes with different functionalities are proposed. We mainly introduce certificateless signatures and aggregate signatures below. Boneh et al. [13] proposed the first aggregate signature using pairing. It allows the third party to aggregate multiple signatures into a single short signature. In 2014, Bansarkhani and Buchmann [14] proposed the first lattice-based sequential aggregate signature scheme under the random oracle model. And there are many aggregate signatures in recent years [15–18] which require the existence of trusted KGC.

In 2009, Zhang et al. [19] proposed the certificateless aggregate signature using bilinear pairing. Later, many researchers have been investigating the topic [20,21]. Tian and Huang [22] proposed the first lattice-based certificateless signature in 2015 that uses the GPV algorithm to generate key pairs which is inefficient. In 2019, Xie et al. [23] proposed the first certificateless aggregate signature with trapdoors. Various schemes have been proposed recently [24–26], which are both certificateless signature schemes but unable to aggregate the signature. In 2020, Yao et al. [27] proposed a lattice-based sequential aggregate signature scheme under the standard model, while the computation overhead is relatively high.

In this paper, we propose a certificateless signature scheme and the first lattice-based certificateless aggregate signature without trapdoors that satisfies the following properties:

1. Certificateless: Our scheme has overcome the limitation of trusted third parties.
2. Aggregation: Efficiently aggregate a user's signatures to improve holistic verification efficiency. It's extremely suitable for cloud storage and users can verify the integrity of their data stored in the cloud.
3. Simpleness: We only need to calculate matrix multiplication and addition avoiding generations of trapdoors and other stuff. Furthermore, we've made theoretical and experimental analysis, showing that our scheme is more suitable for the cloud storage system.
4. Security: We give formal security proofs which show that our scheme is secure against both type I and II adversaries.

1.2 Overview of Our Scheme

In Sect. 3, we elaborate on our construction of the signature scheme based on the hardness of the SIS problem. And our scheme is the modification of Lyubashevsky [10].

The user's private key consists of two parts: a partial private key generated by the KGC and a secret value chosen by themselves. The first part of keys generated by KGC will use KGC's private key by computing $\boldsymbol{s_1} = \boldsymbol{BY}$ and $\boldsymbol{R}G(id, \boldsymbol{s_1}) + \boldsymbol{Y}$. When receiving this part, the user can verify its validity. The second part of the secret key is $\boldsymbol{P}$ with a short norm. We'll use both parts to generate the signature.

Choosing when to output the signature $\theta = (\boldsymbol{v}, \boldsymbol{z}, \boldsymbol{z_2})$, we use rejection sampling on $\boldsymbol{z}$ and $\boldsymbol{z_2}$ respectively, because they are independent. The procedure of rejection sampling is that given two distributions f, g and a constant $M \in \mathbb{R}$ satisfying $f(x) \leq Mg(x)$. If users sample x from g and output it with probability $1/Mg(x)$, the resulting distribution of x is exactly f.

The crux of the security proof is that if there exists a type I or II adversary who successfully forges a signature, then we could solve SIS problems by using forking lemma. Finally, considering our scheme, the length of parameters is relatively large. We believe that shortening the space cost of our signature scheme is needed.

1.3 Organization of The Paper

In Sect. 2, we review some fundamental knowledge that will be used in our scheme. In Sect. 3, we construct our basic certificateless signature scheme and we rigorously prove its security based on the SIS problem. In Sect. 4, we present our certificateless aggregate signature scheme and analyze its security. In Sect. 5, we make theoretical and experimental comparisons with other related schemes. In Sect. 6, we construct the cloud storage system using our scheme. Finally, we make a conclusion in Sect. 7.

2 Preliminaries

2.1 Notations

Now we sketch notations that will be used later. We denote the real number set by $\mathbb{R}$ and the integer set by $\mathbb{Z}$. The notation $\mathbb{Z}_q$ represents the set that elements are in the range $[-\frac{q-1}{2}, \frac{q-1}{2}]$. By convention, column vectors are typed in bold lower-case letters, e.g. $\boldsymbol{a}$, and a_i is the i-th component of the vector $\boldsymbol{a}$. The length of a vector $\boldsymbol{a}$ is its Eculidean norm (l_2 norm) defined as $\|\boldsymbol{a}\| = \sqrt{\sum_i a_i^2}$. Matrices are denoted by bold capital letters, e.g. $\boldsymbol{A}$, and $\boldsymbol{a_i}$ is the i-th column vector of $\boldsymbol{A}$. The length of a matrix $\boldsymbol{A}$ is the l_2 norm of its longest vector. All matrix and vector calculations will end up with modulo q.

2.2 Lattice

Definition 1. Let $\boldsymbol{B} = \{\boldsymbol{b_1}, \ldots, \boldsymbol{b_n} \in \mathbb{R}^n\}$ be composed of n linearly independent vectors. The n dimensional lattice Λ is defined as linear combinations of vectors in $\boldsymbol{B}$, denoted by:

$$\Lambda = \{\boldsymbol{Bc} = \sum_{i \in \{1,\ldots,n\}} \boldsymbol{b_i} \cdot c_i, \boldsymbol{c} \in \mathbb{Z}^n\}$$

In this paper, we only use integer lattices, indicating that $\boldsymbol{B} = \{\boldsymbol{b_1}, \ldots, \boldsymbol{b_n} \in \mathbb{Z}^n\}$.

2.3 The SIS Problem

The security of our scheme is based on the SIS problem. We'll give a description of SIS and its variants.

Definition 2. Given a matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$, find a short vector $\boldsymbol{v} \in \mathbb{Z}^m$ satisfying $\boldsymbol{Av} = 0$ and $\|\boldsymbol{v}\| \leq \sqrt{m} a^{n/m}$.

Definition 3. Given a pair $(\boldsymbol{A}, \boldsymbol{t})$ where $\boldsymbol{t} = \boldsymbol{Av}$, find a vector $\boldsymbol{v} \in \{-d, \ldots, 0, \ldots, d\}^m$ satisfying $\boldsymbol{Av} = \boldsymbol{t}$.

The lemma below is important when we prove the security using the forking lemma [28,29].

Lemma 1 [10]. For any matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$ where $m > 64 + n \cdot logq/log(2d+1)$, and a randomly chosen vector $\boldsymbol{s} \leftarrow \{-d, \ldots, 0, \ldots, d\}^m$, there exists another $\boldsymbol{s'} \leftarrow \{-d, \ldots, 0, \ldots, d\}^m$ such that $\boldsymbol{As} = \boldsymbol{As'}$ with probability $1 - 2^{-100}$.

2.4 Gaussian on Lattices

We mainly review Gaussian distributions following the prior works [30,31].

Definition 4 [30]. For any m dimensional vector $\boldsymbol{x}$, the continuous Gaussian function centered in $\boldsymbol{c}$ scaled by a factor of σ is defined by the function

$$\rho_{\boldsymbol{c},\sigma}(\boldsymbol{x}) = e^{-\pi\|(\boldsymbol{x}-\boldsymbol{c})/\sigma\|^2}$$

When $\boldsymbol{c} = 0$, we represent it by the notation $\rho_\sigma(\boldsymbol{x})$.
We can get the total measure of $\rho_{\boldsymbol{c},\sigma}$

$$\int_{\boldsymbol{x}\in\mathbb{R}^m} \rho_{\boldsymbol{c},\sigma}(\boldsymbol{x})d\boldsymbol{x} = \sigma^m$$

The probability density function is defined as follows:

$$\forall \boldsymbol{x} \in \mathbb{R}^m, D^m_{\boldsymbol{c},\sigma}(\boldsymbol{x}) = \frac{\rho_{\boldsymbol{c},\sigma}(\boldsymbol{x})}{\sigma^m}$$

Let A be a countable subset of $\mathbb{Z}^m$, the notation $\rho_{\boldsymbol{c},\sigma}(A)$ represents $\rho_{\boldsymbol{c},\sigma}(A) = \sum_{\boldsymbol{x}\in A}\rho_{\boldsymbol{c},\sigma}(\boldsymbol{x})$.

Definition 5 [32]. The discrete Gaussian distribution over lattice Λ, centered at $\boldsymbol{c}$, is defined as follows using the notation $D^m_{\Lambda,\boldsymbol{c},\sigma}$:

$$\forall \boldsymbol{x} \in \Lambda, D^m_{\Lambda,\boldsymbol{c},\sigma}(\boldsymbol{x}) = \frac{D^m_{\boldsymbol{c},\sigma}(\boldsymbol{x})}{D^m_{\boldsymbol{c},\sigma}(\Lambda)} = \frac{\rho_{\boldsymbol{c},\sigma}(\boldsymbol{x})}{\rho_{\boldsymbol{c},\sigma}(\Lambda)}$$

Now we give some lemmas from [33,34].

Lemma 2.

1. $Pr[|z| > \omega(\sigma\sqrt{\log m}); z \leftarrow D^1_\sigma] = 2^{-\omega(\log m)}$, and more specifically, $Pr[|z| > 12\sigma; z \leftarrow D^1_\sigma] < 2^{-100}$.
2. $Pr[||z|| > 2\sigma\sqrt{m}; \boldsymbol{z} \leftarrow D^m_\sigma(\mathbf{z})] \leq 2^{-m+1}$.

2.5 Rejection Sampling

The following theorem shows that the vector generated by rejection sampling is indistinguishable from the one that samples from the Gaussian distribution.

Theorem 1 [10]. V is a subset of $\mathbb{Z}^m$ in which all elements have norms less than T. Choose the parameter $\sigma = \omega(T\sqrt{logm})$. h is a distribution $h : V \to \mathbb{R}$. Then there exists a constant $M = O(1)$ such that the statistical distance of the output of the following two algorithms is less than $\frac{2^{-100}}{M}$.

1. $v \leftarrow h, \boldsymbol{z} \leftarrow D^m_{v,\sigma}$,
   output $(\boldsymbol{z}, v)$ with probability $\min(\frac{D^m_\sigma(z)}{MD^m_{v,\sigma}(\boldsymbol{z})}, 1)$.
2. $v \leftarrow h, \boldsymbol{z} \leftarrow D^m_\sigma$,
   output $(\boldsymbol{z}, v)$ with probability $1/M$.

2.6 Signature Scheme

Definition 6. A certificateless aggregate signature scheme is composed of the following 8 algorithms:

- **Setup**: The algorithm takes as input the security parameter. Then it outputs the public parameters $params$ and the master key.
- **Partial-Private-Key**: The algorithm takes as input a user's identity id and the master key. Then outputs the partial private key for the user.
- **Set-Private-Key**: The user inputs the partial private key and chooses the secret value. Then it outputs the user's private key.
- **Set-Public-Key**: After generating the secret key, the user runs this algorithm to output the corresponding public key.
- **Sign**: On Inputting the signer's secret key and a message, the algorithm outputs a signature.
- **Verify**: On inputting the public key and a signature, the verifier verifies the validity of the signature.
- **AggSign**: After inputting a user's request string $s \in \{-1, 0, 1\}^l$ and the user's signatures, the cloud generates the aggregate signature.
- **AggVerify**: After receiving the aggregate signature, the user checks whether the aggregate signature corresponds to the submitted s and verifies the correctness of the signature.

We omit the input of public parameters $params$ in each algorithm.

2.7 Security Models

We will consider the following security games for Type I and Type II adversaries. Note that for Type I adversary, KGC plays the role of the challenger, while for Type II, KGC is the adversary.

Type I adversary has the ability to know the secret value of any entity and replace the associated public key.

Type II adversary has the ability to produce the partial private key but does not know the associated secret value.

Definition 7. The certificateless aggregate signature scheme is EUF-CMA secure if there is no adversary $\mathcal{F}$ who can forge a valid signature with non-negligible advantage $Adv(\mathcal{F}) = |Pr(\mathcal{F}\ wins)|$.

Setup. The challenger runs the **Setup** algorithm to generate public parameters and the master key. Then the challenger forwards $params, mpk$ to $\mathcal{F}$.

Partial-Private-Key. $\mathcal{F}$ sends an identity to the challenger. Then the challenger returns the corresponding partial private key to $\mathcal{F}$. For the Type II adversary, he could compute any partial private keys because he owns msk.

Replace Public Key. If Type I adversary $\mathcal{F}$ sends an identity and the public key to the challenger, the challenger will replace the public key of the corresponding identity.

Signing Queries. $\mathcal{F}$ issues queries with massage msg and the identity, the challenger runs the **Sign** algorithm to generate a signature and sends it to $\mathcal{F}$.

Forgery. $\mathcal{F}$ outputs a signature for the identity id. $\mathcal{F}$ wins the game when the following states hold:

1. The signature generated by $\mathcal{F}$ was never issued in **Signing** queries.
2. For the Type I adversary, the partial private key on id was never queried.
3. For the Type II adversary, the public key replacement on id was never queried.

3 Basic Signature Scheme Based on SIS

3.1 Our Scheme

In this section, we present our signature scheme based on SIS as follows and demonstrate the correctness of the single signature. Now we explain the concrete workings of our scheme.

Setup. Given the system parameter n, the system initialization performs as follows:

1. Choose the parameter $q = poly(n)$, m and d according to Lemma 1.
2. Define the hash functions $H : \{0,1\}^* \rightarrow \{-d, \ldots, 0, \ldots, d\}^n$, $H_1 : \{0,1\}^* \rightarrow \{-1,0,1\}^k$, $G : \{0,1\}^* \rightarrow \{-1,0,1\}^{n \times k}$, and $G_2 : \{0,1\}^* \rightarrow \{-1,0,1\}^{n \times m}$.
3. Randomly choose a matrix $\boldsymbol{B} \in \mathbb{Z}_q^{n \times m}$.
4. Choose a matrix with short l_2 norm $\boldsymbol{R} \in \mathbb{Z}_2^{m \times n}$.
5. Compute the matrix $\boldsymbol{T} = \boldsymbol{BR}$.
6. Output the public parameters $params = \{\boldsymbol{T}, \boldsymbol{B}, H, H_1, G, G_2\}$ and keep the matrix $\boldsymbol{R}$ as master secret key.

Partial-Private-Key. On inputting the user's identity id, public parameters, and the master secret key, KGC runs the following algorithm:

1. Choose $\boldsymbol{Y} \in D_\sigma^{m \times k}$ with a short norm.
2. Compute $\boldsymbol{s_1} = H(\boldsymbol{BY})$.
3. Compute the vector $\boldsymbol{F} = G(id, \boldsymbol{s_1})$.
4. Compute $\boldsymbol{S_2} = \boldsymbol{RF} + \boldsymbol{Y}$.
5. Finally, return $(\boldsymbol{s_1}, \boldsymbol{S_2})$ as its partial private key.

Set-Private-Key. On inputting the partial private key $(\boldsymbol{s_1}, \boldsymbol{S_2})$, the user's identity id, the user generates its private key as follows:

1. The user verifies that $\|\boldsymbol{S_2}\| \leq n + 2\sigma\sqrt{m}$ and $H(\boldsymbol{BS_2} - \boldsymbol{T}G(id, \boldsymbol{s_1})) = \boldsymbol{s_1}$. If it fails, then the algorithm aborts. Otherwise, the user executes the subsequent step.
2. The user randomly chooses the matrix $\boldsymbol{P} \in \mathbb{Z}_2^{m \times k}$.
3. Set $(\boldsymbol{S_2}, \boldsymbol{P})$ as his private key.

Set-Public-Key. On inputting a user's identity id and the corresponding secret key $(\boldsymbol{S_2}, \boldsymbol{P})$, the user generates its public key:

1. Compute the corresponding matrix $\boldsymbol{A} = G_2(id) \in \mathbb{Z}_q^{n \times m}$.
2. Compute $\boldsymbol{K} = \boldsymbol{AP}$.
3. Compute $\boldsymbol{K_2} = \boldsymbol{BS_2} - \boldsymbol{T}G(id, \boldsymbol{s_1})$.
4. Set $\{id, \boldsymbol{K}, \boldsymbol{K_2}\}$ as its public key.

Sign. When the user needs to generate a signature for the message msg, the user runs the algorithm along with the input of the user's identity id and its corresponding secret key $(\boldsymbol{S_2}, \boldsymbol{P})$.

1. Randomly choose $\boldsymbol{y_1}, \boldsymbol{y_2} \leftarrow D_\sigma^m$.
2. $\boldsymbol{u} = G_2(id)\boldsymbol{y_1} + \boldsymbol{By_2} + H(msg)$.
3. Generate the hash value $\boldsymbol{v} = H_1(\boldsymbol{u})$.
4. Compute $\boldsymbol{z} = \boldsymbol{Pv} + \boldsymbol{y_1}$.
5. Compute $\boldsymbol{z_2} = \boldsymbol{S_2 v} + \boldsymbol{y_2}$.
6. Accept $\boldsymbol{z}$ with probability $min(\frac{D_\sigma^m(z)}{MD^m_{\boldsymbol{Pv},\sigma}(z)}, 1)$ and accept $\boldsymbol{z_2}$ with probability $min(\frac{D_\sigma^m(\boldsymbol{z_2})}{MD^m_{\boldsymbol{S_2 v},\sigma}(\boldsymbol{z_2})}, 1)$. Finally, output $\theta = (\boldsymbol{v}, \boldsymbol{z}, \boldsymbol{z_2})$.

Verify. After receiving a signature θ, the verifier checks whether it's a legal signature using the corresponding public key.

1. Check $\|\boldsymbol{z}\| \leq 2\sigma\sqrt{m}$ and $\|\boldsymbol{z_2}\| \leq 2\sigma\sqrt{m}$, output 0 if not satisfied.
2. Verify the equation $H_1(G_2(id)\boldsymbol{z} - \boldsymbol{Kv} + \boldsymbol{Bz_2} - (\boldsymbol{T}G(id, H(\boldsymbol{K_2})) + \boldsymbol{K_2})\boldsymbol{v} + H(msg)) = \boldsymbol{v}$. Output 0 if it fails, otherwise output 1.

Correctness:
In the **Verify** algorithm, the verifier inputs the signer's public key, the signature along with the corresponding message to compute $\boldsymbol{u'}$.

$$
\begin{aligned}
&G_2(id)\boldsymbol{z} - \boldsymbol{Kv} + \boldsymbol{Bz_2} - (\boldsymbol{T}G(id, H(\boldsymbol{K_2})) + \boldsymbol{K_2})\boldsymbol{v} + H(msg) \\
&= G_2(id)(\boldsymbol{Pv} + \boldsymbol{y_1}) - \boldsymbol{Kv} \\
&+\boldsymbol{B}(\boldsymbol{S_2 v} + \boldsymbol{y_2}) - (\boldsymbol{T}G(id, H(\boldsymbol{K_2})) + \boldsymbol{K_2})\boldsymbol{v} \\
&+H(msg) \\
&= K\boldsymbol{v} + G_2(id)\boldsymbol{y_1} - \boldsymbol{Kv} + (\boldsymbol{T}G(id, H(\boldsymbol{K_2})) + \boldsymbol{K_2})\boldsymbol{v} + \boldsymbol{By_2} \\
&-(\boldsymbol{T}G(id, H(\boldsymbol{K_2})) + \boldsymbol{K_2})\boldsymbol{v} + H(msg) \\
&= G_2(id)\boldsymbol{y_1} + \boldsymbol{By_2} + H(msg) \\
&= \boldsymbol{u'}
\end{aligned}
$$

After getting $\boldsymbol{u'}$, verifier checks the following equation $H(\boldsymbol{u'}) = \boldsymbol{v}$. If it fails, output 0. Otherwise, output 1.

3.2 Security Proof of Basic Scheme

In the certificateless scheme, there exist two types of adversaries denoted by type I adversary and type II adversary. We'll illustrate concrete security proof respectively.

Theorem 2. Suppose that there exists a Type II adversary who makes at most h times hash queries and s times signing queries, and succeeds in forging a signature with probability δ. Then there exists an algorithm that given a matrix $\boldsymbol{A}$, finds a vector $\boldsymbol{s} \leq 4\sigma\sqrt{m}+2dk$ satisfying $\boldsymbol{As} = 0$ with non-negligible probability.

Proof. For the type II adversary, he could get the partial private key $\boldsymbol{S_2}$ but cannot get $\boldsymbol{P}$ generated by a user. The security of the signature is totally based on $\boldsymbol{z}$ and we can ignore $\boldsymbol{z_2}$ because it can be seen as a constant. Let $D_H = \{\boldsymbol{c} : \boldsymbol{c} \in \{-1, 0, 1\}^k\}$ denote the image range of the random oracle H_1. The forger $\mathcal{F}$ calls hash query h times and signing query s times. Let $t = h + s$ denote the total number of the hash queries called during the attack. Then the challenger $\mathcal{A}$ picks $\boldsymbol{r_1}, \ldots, \boldsymbol{r_t} \leftarrow D_H$. When getting a new hash query, $\mathcal{A}$ returns the first $\boldsymbol{r_i}$ in $\boldsymbol{r_1}, \ldots, \boldsymbol{r_t}$ which has not yet been used as response. $\mathcal{A}$ will store all information of the hash query in a table which will be used later. So when getting a repeated hash query, $\mathcal{A}$ returns the same $\boldsymbol{r_i}$. After finishing queries, $\mathcal{F}$ outputs a forged signature.

$\mathcal{F}$ outputs a valid signature $(\boldsymbol{v}, \boldsymbol{z})$ with probability δ. Notice that the probability that $\mathcal{F}$ outputs a signature that is irrelevant to all queries and satisfies the equation $H_1(\boldsymbol{Az} - \boldsymbol{Kv} + H(msg)) = \boldsymbol{v}$ is $\frac{1}{|D_H|}$. Therefore, $\boldsymbol{v}$ must be one of $\boldsymbol{r_i}$ in $\mathcal{A}$'s table with probability $1 - \frac{1}{|D_H|}$. Pay attention that $\boldsymbol{r_i}$ could be returned as a response either during a hash query or in a signing query. Now we discuss these two queries in detail.

We discuss the case that $\boldsymbol{r_i}$ is used during a signing query. When receiving a signing query. $\mathcal{A}$ chooses $\boldsymbol{z} \in D_\sigma^m$ and $\boldsymbol{v} \in \{-1, 0, 1\}^k$, and record $(\boldsymbol{Az} - \boldsymbol{Kv}, \boldsymbol{v}, msg)$ in the table. Because $\boldsymbol{z_2}$ can be seen as a constant for type II adversary, $\mathcal{A}$ then returns $(\boldsymbol{z}, \boldsymbol{v})$ as response.

After t times queries, $\mathcal{F}$ generates a valid forged signature $(\boldsymbol{v}, \boldsymbol{z})$. Then in $\mathcal{A}$'s hash table, we could find $H_1(\boldsymbol{Az'} - \boldsymbol{Kv} + H(msg')) = H_1(\boldsymbol{Az} - \boldsymbol{Kv} + H(msg))$. If $\boldsymbol{Az'} - \boldsymbol{Kv} \neq \boldsymbol{Az} - \boldsymbol{Kv}$ or $H(msg') \neq H(msg)$, only when $\mathcal{F}$ gets the pre-image of $\boldsymbol{r_i}$ can he achieves the equality. So when $H(msg') = H(msg)$ and $\boldsymbol{Az'} - \boldsymbol{Kv} \neq \boldsymbol{Az} - \boldsymbol{Kv}$, we get the equality $\boldsymbol{A(z' - z)} = 0$. Because $\|z\| \leq 2\sigma\sqrt{m}$, so we get the short vector satisfying $\|z' - z\| \leq 4\sigma\sqrt{m}$.

Now we turn to the hash query made by $\mathcal{F}$. $\mathcal{A}$ records a signature $(\boldsymbol{z}, \boldsymbol{r_j})$ on m from $\mathcal{F}$, the $\mathcal{A}$ generates new elements $\boldsymbol{r'_1}, \ldots, \boldsymbol{r'_t} \leftarrow D_H$. Then $\mathcal{A}$ repeats queries again with input $\boldsymbol{r_1}, \ldots, \boldsymbol{r_{j-1}}, \ldots, \boldsymbol{r'_t}$. By using the forking lemma, the probability that $\boldsymbol{r_j} \neq \boldsymbol{r'_j}$ and $\mathcal{F}$ uses $\boldsymbol{r'_j}$ in his forgery is

$$(\delta - \frac{1}{|D_H|})(\frac{\delta - 1/|D_H|}{t} - \frac{1}{|D_H|})$$

Therefore, $\mathcal{F}$ outputs a signature $(\boldsymbol{z'}, \boldsymbol{r'_j})$ and $\boldsymbol{Az'} - \boldsymbol{Kr'_j} + H(msg)) = \boldsymbol{Az} - \boldsymbol{Kr_j} + H(msg))$ where $\boldsymbol{v} = \boldsymbol{r_j}$ and $\boldsymbol{v'} = \boldsymbol{r'_j}$. Then we get:

$$\boldsymbol{A}(\boldsymbol{z} - \boldsymbol{z'} + \boldsymbol{Pv'} - \boldsymbol{Pv}) = 0$$

The norm of vector satisfies

$$(\boldsymbol{z} - \boldsymbol{z'} + \boldsymbol{Pv'} - \boldsymbol{Pv}) \leq 4\sigma\sqrt{m} + 2dk$$

Now, we could even ignore the hardness of secret key $\boldsymbol{s}$ and illustrate the security proof based on $\boldsymbol{P}$. By Lemma 1, we know that there exists another $\boldsymbol{P}$ satisfying $\boldsymbol{AP} = \boldsymbol{AP'}$ with as least $1 - 2^{-100}$ probability. If

$$(\boldsymbol{z} - \boldsymbol{z'} + \boldsymbol{Pv'} - \boldsymbol{Pv}) = 0$$

then

$$(\boldsymbol{z} - \boldsymbol{z'} + \boldsymbol{P'v'} - \boldsymbol{P'v}) \neq 0$$

Because $\mathcal{A}$ doesn't know these secret keys and $\mathcal{F}$ doesn't know which secret key $\mathcal{A}$ uses. Besides, $\mathcal{A}$ uses these two secret keys with equal probability. Therefore, with probability 1/2, $\mathcal{A}$ can get a non-zero result.

Theorem 3. Suppose that there exists a Type I adversary who makes at most h times hash queries and s times signing queries, and succeeds in forging a signature with probability δ. Then there exists an algorithm that given a matrix $\boldsymbol{A}$, finds a vector $\boldsymbol{s} \leq (4\sigma + 4k\sigma)\sqrt{m} + 2nk$ such that $\boldsymbol{As} = 0$ with non-negligible probability.

Proof. For the type I adversary, he could get the secret key $\boldsymbol{P}$ generated by a user but he cannot get the partial private key $\boldsymbol{S_2}$. The proof is similar to II adversary. The security of the signature is totally based on $\boldsymbol{z_2}$ and we can ignore $\boldsymbol{z}$ because it can be seen as a constant. Let $D_H = \{\boldsymbol{c} : \boldsymbol{c} \in \{-1, 0, 1\}^k\}$ denote the image range of the random oracle H_1. The forger $\mathcal{F}$ calls hash query h times and signing query s times. Let $t = h + s$ denote the total number of the hash queries called during the attack. Then the challenger $\mathcal{A}$ picks $\boldsymbol{r_1}, \ldots, \boldsymbol{r_t} \leftarrow D_H$. When getting a new hash query, $\mathcal{A}$ returns the first $\boldsymbol{r_i}$ in $\boldsymbol{r_1}, \ldots, \boldsymbol{r_t}$ which has not yet been used as response. $\mathcal{A}$ will store all information of the hash query in a table which will be used later. So when getting a repeated hash query, $\mathcal{A}$ returns the same $\boldsymbol{r_i}$. After finishing queries, $\mathcal{F}$ outputs a forged signature.

$\mathcal{F}$ outputs a valid signature $(\boldsymbol{v}, \boldsymbol{z_2})$ with probability δ. Notice that the probability that $\mathcal{F}$ outputs a signature which is irrelevant to all queries and satisfies the equation $H_1(\boldsymbol{Bz_2} - (\boldsymbol{T}G(id, \boldsymbol{K_2}) + \boldsymbol{K_2})\boldsymbol{v} + H(msg)) = \boldsymbol{v}$ is $\frac{1}{|D_H|}$. Therefore, $\boldsymbol{v}$ must be one of $\boldsymbol{r_i}$ in $\mathcal{A}$'s table with probability $1 - \frac{1}{|D_H|}$. Pay attention that $\boldsymbol{r_i}$ could be returned as a response either during hash query or in signing query. Now we discuss these two queries in detail.

We discuss the case that $\boldsymbol{r_i}$ is used during the signing query. When receiving a signing query. $\mathcal{A}$ chooses $\boldsymbol{z_2} \in D_\sigma^m$ and $\boldsymbol{v} \in \{-1, 0, 1\}^k$, and record $(\boldsymbol{Az} -$

$(\boldsymbol{TG}(id, \boldsymbol{K_2}) + \boldsymbol{K_2})\boldsymbol{v}, \boldsymbol{v}, msg)$ in the table. Because $\boldsymbol{z}$ can be seen as a constant for type I adversary, $\mathcal{A}$ then returns $(\boldsymbol{z_2}, \boldsymbol{v})$ as response.

After t times queries, $\mathcal{F}$ generates a valid forged signature $(\boldsymbol{v}, \boldsymbol{z_2})$. Then in $\mathcal{A}$'s hash table, we could find $H_1(\boldsymbol{Bz'_2} - (\boldsymbol{TG}(id, \boldsymbol{K_2}) + \boldsymbol{K_2})\boldsymbol{v} + H(msg')) = H_1(\boldsymbol{Bz_2} - (\boldsymbol{TG}(id, \boldsymbol{K_2}) + \boldsymbol{K_2})\boldsymbol{v} + H(msg))$. If $\boldsymbol{Bz'_2} - (\boldsymbol{TG}(id, \boldsymbol{K_2}) + \boldsymbol{K_2})\boldsymbol{v} \neq \boldsymbol{Bz_2} - (\boldsymbol{TG}(id, \boldsymbol{K_2}) + \boldsymbol{K_2})\boldsymbol{v}$ or $H(msg') \neq H(msg)$, only when $\mathcal{F}$ gets the pre-image of $\boldsymbol{r_i}$ can he achieves the equality. So when $H(msg') = H(msg)$ and $\boldsymbol{Bz'_2} - (\boldsymbol{TG}(id, \boldsymbol{K_2}) + \boldsymbol{K_2})\boldsymbol{v} \neq \boldsymbol{Bz_2} - (\boldsymbol{TG}(id, \boldsymbol{K_2}) + \boldsymbol{K_2})\boldsymbol{v}$, we get the equality $\boldsymbol{B(z'_2 - z_2)} = 0$. Because $\|z_2\| \leq 2\sigma\sqrt{m}$, so we get the short vector satisfying $\|z'_2 - z_2\| \leq 4\sigma\sqrt{m}$.

Now we turn to the hash query made by $\mathcal{F}$. $\mathcal{A}$ records a signature $(\boldsymbol{z}, \boldsymbol{r_j})$ on m from $\mathcal{F}$, the $\mathcal{A}$ generates new elements $\boldsymbol{r'_1}, \ldots, \boldsymbol{r'_t} \leftarrow D_H$. Then $\mathcal{A}$ repeats queries again with input $\boldsymbol{r_1}, \ldots, \boldsymbol{r_{j-1}}, \ldots, \boldsymbol{r'_t}$. By using the forking lemma, the probability that $\boldsymbol{r_j} \neq \boldsymbol{r'_j}$ and $\mathcal{F}$ uses $\boldsymbol{r'_j}$ in his forgery is

$$(\delta - \frac{1}{|D_H|})(\frac{\delta - 1/|D_H|}{t} - \frac{1}{|D_H|})$$

Therefore, $\mathcal{F}$ outputs a signature $(\boldsymbol{z'_2}, \boldsymbol{r'_j})$ and $\boldsymbol{Bz'_2} - (\boldsymbol{TG}(id, \boldsymbol{K_2}) + \boldsymbol{K_2})\boldsymbol{r'_j} + H(msg)) = \boldsymbol{Bz_2} - (\boldsymbol{TG}(id, \boldsymbol{K_2}) + \boldsymbol{K_2})\boldsymbol{r_j} + H(msg))$ where $\boldsymbol{v} = \boldsymbol{r_j}$ and $\boldsymbol{v'} = \boldsymbol{r'_j}$. Then we get:

$$\boldsymbol{B}(\boldsymbol{z_2 - z'_2 + (RG(id, K_2) + Y)v' - (RG(id, K_2) + Y)v}) = 0$$

The norm of vector satisfies

$$\begin{aligned}(\boldsymbol{z_2 - z'_2 + (RG(id, K_2) + Y)v' - (RG(id, K_2) + Y)v}) \\ \leq (4\sigma + 4k\sigma)\sqrt{m} + 2nk\end{aligned}$$

.

Now, we could even ignore the hardness of secret key $\boldsymbol{s}$ and illustrate the security proof based on $\boldsymbol{R}$. By Lemma 1, we know that there exists another $\boldsymbol{P}$ such that $\boldsymbol{BR} = \boldsymbol{BR'}$ with as least $1 - 2^{-100}$ probability. If

$$(\boldsymbol{z_2 - z'_2 + (RG(id, K_2) + Y)v' - (RG(id, K_2) + Y)v}) = 0$$

then

$$(\boldsymbol{z_2 - z'_2 + (R'G(id, K_2) + Y)v' - (R'G(id, K_2) + Y)v}) \neq 0$$

Because $\mathcal{A}$ doesn't know these secret keys and $\mathcal{F}$ doesn't know which secret key $\mathcal{A}$ uses. Besides, $\mathcal{A}$ uses these two secret keys with equal probability. Therefore, with probability 1/2, $\mathcal{A}$ can get a non-zero result.

4 Aggregate Signature Scheme Based on SIS

4.1 Our Scheme

The basic signature scheme is the same as above. We now present the aggregation process.

AggSign. On input a user's id request string $s \in \{-1, 0, 1\}^l$, where l is the total number of message blocks stored in the cloud, then the cloud server generates the aggregate signatures θ which are generated using the above basic signature $\theta_i = (\boldsymbol{v_i}, \boldsymbol{z_i}, \boldsymbol{z_{2i}})$.

1. $\boldsymbol{u_i} = G_2(id)\boldsymbol{z_i} - \boldsymbol{K}\boldsymbol{v_i} + \boldsymbol{B}\boldsymbol{z_{2i}} - (\boldsymbol{T}G(id, H(\boldsymbol{K_2})) + \boldsymbol{K_2})\boldsymbol{v_i} + H(msg_i)$.
2. Compute $\boldsymbol{u} = \sum_{i=1}^{l} s_i \boldsymbol{u_i}$.
3. $\boldsymbol{v} = \sum_{i=1}^{l} s_i H_1(\boldsymbol{u_i}) + H_1(\boldsymbol{u})$.
4. $\boldsymbol{z} = \sum_{i=1}^{l} s_i \boldsymbol{z_i}$.
5. $\boldsymbol{z_2} = \sum_{i=1}^{l} s_i \boldsymbol{z_{2i}}$.
6. Finally, output the aggregate signature $\theta = (\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{z}, \boldsymbol{z_2})$.

AggVerify. The verifier verifies the aggregate signature and the corresponding message.

1. Check $\|\boldsymbol{z}\| \leq 2l\sigma\sqrt{m}$ and $\|\boldsymbol{z_2}\| \leq 2l\sigma\sqrt{m}$, output 0 if not satisfied.
2. The user verifies the equation:

$$G_2(id)\boldsymbol{z} - \boldsymbol{K}\boldsymbol{v} + \boldsymbol{K}H_1(\boldsymbol{u}) + \boldsymbol{B}\boldsymbol{z_2} - (\boldsymbol{T}G(id, H(\boldsymbol{K_2})) + \boldsymbol{K_2})\boldsymbol{v} = \boldsymbol{u} - \sum s_i H(msg_i)$$

Output 0 if fails, otherwise output 1.

Correctness:
After getting the aggregate signature $\theta = (\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{z}, \boldsymbol{z_2})$, the user then compute $\boldsymbol{u'}$ as follows to check whether $\boldsymbol{u'} = \boldsymbol{u}$. If it fails, output 0. Otherwise, output 1.

$$\begin{aligned}
&G_2(id)\boldsymbol{z} - \boldsymbol{K}\boldsymbol{v} + \boldsymbol{B}\boldsymbol{z_2} - (\boldsymbol{T}G(id, H(\boldsymbol{K_2})) + \boldsymbol{K_2})\boldsymbol{v} + \sum s_i H(msg_i) \\
&= G_2(id)\sum(\boldsymbol{P}\boldsymbol{v_i} + \boldsymbol{y_{1i}}) - \sum \boldsymbol{K}\boldsymbol{v_i} - \boldsymbol{K}H_1(\boldsymbol{u}) + \boldsymbol{K}H_1(\boldsymbol{u}) \\
&+ \sum \boldsymbol{B}(\boldsymbol{S_2}\boldsymbol{v_i} + \boldsymbol{y_{2i}}) - \sum(\boldsymbol{T}G(id, H(\boldsymbol{K_2})) + \boldsymbol{K_2})\boldsymbol{v_i} + \sum s_i H(msg_i) \\
&= G_2(id)(\boldsymbol{P}\sum \boldsymbol{v_i} + \sum \boldsymbol{y_{1i}}) - \sum \boldsymbol{K}\boldsymbol{v_i} + \boldsymbol{B}(\boldsymbol{S_2}\sum \boldsymbol{v_i} + \sum \boldsymbol{y_{2i}}) \\
&- (\boldsymbol{T}G(id, H(\boldsymbol{K_2}) + \boldsymbol{K_2})\sum \boldsymbol{v_i} + \sum s_i H(msg_i) \\
&= \boldsymbol{K}\sum \boldsymbol{v_i} + G_2(id)\sum \boldsymbol{y_{1i}} - \boldsymbol{K}\sum \boldsymbol{v_i} + (\boldsymbol{T}G(id, H(\boldsymbol{K_2})) + \boldsymbol{K_2})\sum \boldsymbol{v_i} \\
&+ \sum \boldsymbol{B}\boldsymbol{y_{2i}} - (\boldsymbol{T}G(id, H(\boldsymbol{K_2})) + \boldsymbol{K_2})\sum \boldsymbol{v_i} + \sum s_i H(msg_i) \\
&= G_2(id)\sum \boldsymbol{y_{1i}} + \sum \boldsymbol{B}\boldsymbol{y_{2i}} + \sum s_i H(msg_i) \\
&= \sum \boldsymbol{u_i} \\
&= \boldsymbol{u'}
\end{aligned}$$

4.2 Security Proof of Aggregate Signature Scheme

In the basic scheme, the purpose of not directly providing $\boldsymbol{u}$ is to shorten the signature length. However, the security of this signature does not depend on the privacy of $\boldsymbol{u}$, but on the unforgeability of $\boldsymbol{z}$ and $\boldsymbol{z_2}$. So in the aggregate scheme, for the Type I adversary, he cannot forge a signature with short $\boldsymbol{z_2}$. And for the Type II adversary, he cannot forge a signature with short $\boldsymbol{z}$. The details of the proof are consistent with the basic scheme above. We only provide the theorem and omit the specific proof process.

Theorem 4. Suppose that there exists a Type II adversary who makes at most h times hash queries and s times signing queries, and succeeds in forging a signature with probability δ. Then there exists an algorithm that given a matrix $\boldsymbol{A}$, find a vector $\boldsymbol{s} \leq 4l\sigma\sqrt{m} + (2ld+1)k$ satisfying $\boldsymbol{As} = 0$ with non-negligible probability.

Theorem 5. Suppose that there exists a Type I adversary who makes at most h times hash queries and s times signing queries, and succeeds in forging a signature with probability δ. Then there exists an algorithm that given a matrix $\boldsymbol{A}$, find a vector $\boldsymbol{s} \leq (4l\sigma + 4(l+1)k\sigma)\sqrt{m} + 2nk(l+1)$ such that $\boldsymbol{As} = 0$ with non-negligible probability.

Table 1. Functionality comparisons.

Scheme	Certificateless	Aggregation	Trapdoors
Li et al. [1]	✓	✗	✓
Salvakkam et al. [2]	✓	✗	✓
Prajapat et al. [25]	✓	✗	✗
Yang et al. [26]	✓	✗	✓
Yao et al. [27]	✗	✓	✓
Ours	✓	✓	✗

5 Evaluation

In this section, we compare with other certificateless signature schemes [1,2,25, 26] and the aggregate signature scheme [27]. Schemes [1,2] are used for cloud storage which have more comparative significance.

5.1 Theoretical Comparison

Table 1 demonstrates the properties of our scheme compared with related schemes in the aspect of certificateless, aggregation, and trapdoors. Note that the notation ✗ represents that the scheme cannot achieve the feature or doesn't

Table 2. Storage Cost.

Scheme	Public Key	Private Key	Signature
Li et al. [1]	$2nm\|\mathbb{Z}\|$	$2m^2\|\mathbb{Z}\|$	$2ml\|\mathbb{Z}\|$
Salvakkam et al. [2]	$mk\|\mathbb{Z}\|$	$mk\|\mathbb{Z}\|$	$lnm\|\mathbb{Z}\|$
Prajapat et al. [25]	$m\|\mathbb{Z}\|$	$4m\|\mathbb{Z}\|$	$5lm\|\mathbb{Z}\|$
Yang et al. [26]	$nk\|\mathbb{Z}\|$	$mklog(2s) + mklog(2d+1)$	$2lmlog(12\sigma)$
Yao et al. [27]	$(nm+n)\|\mathbb{Z}\|$	$m^2\|\mathbb{Z}\|$	$m\|\mathbb{Z}\|$
Ours	$2nk\|\mathbb{Z}\|$	$mk + mk\|\mathbb{Z}\|$	$n\|\mathbb{Z}\| + 2k + 2mlog(12\sigma)$

Table 3. Computation cost.

Scheme	Signing	Verify
Li et al. [1]	$2lnmT_{mul} + T_{sap} + (n+l)T_H$ $+nlT_{mul}$	$(2nm + n + n^2 + 2nm)T_{mul}$ $+(n+l)T_H$
Salvakkam et al. [2]	$l(3nT_{mul} + T_H + T_{sap})$	$l(3nT_{mul} + T_H)$
Prajapat et al. [25]	$4lm^2T_{mul} + lT_H$	$4lm^2T_{mul} + lT_H$
Yang et al. [26]	$4lmkT_{mul} + lT_H$	$4lmkT_{mul} + 2lT_H$
Yao et al. [27]	$nm^2T_{mul} + T_{bd} + 2T_H$	$n^2m^4T_{mul} + T_{sap} + 2T_H$
Ours	$l(2nm + 2nk)T_{mul} + 3lT_H$	$(2nm + 2nk + n^2)T_{mul} + 3T_H$

use this part and the notation ✓ represents that the scheme achieves this feature or uses this part. Aggregate signatures can verify the integrity of a large amount of data at once, and compared to the scheme [27], our scheme does not require trapdoors, making the algorithm more efficient.

Table 2 illustrates the storage cost. Note that l represents the number of the message blocks. $|\mathbb{Z}|$ represent the number of the element in group $\mathbb{Z}$. In our scheme, the private key size is $mk + mk|\mathbb{Z}|$ which is smaller than [1,2], and the public key size is $2nk|\mathbb{Z}|$ which is smaller than [1,25–27]. When a user requests signatures for l message blocks to check their validity, the length of the aggregate signature is constant, while the length of other schemes is linear to the parameter l.

Table 3 demonstrates the computation cost among these schemes. T_{mul} represents the modular multiplication in the group $\mathbb{Z}_q$. T_H denotes the running time of the hash function. Let T_{sap} and T_{bd} denote the running time of the algorithm $Samplepre$ and $BasisDel$, respectively. Compared with other schemes using $Trapgen$ and $Samplepre$ algorithm, our scheme only uses matrix multiplication and addition, which is more concise and efficient, and can use parallel computing to accelerate efficiency. Table 3 shows the time cost of generating signatures for l message blocks and the cost of verifying these signatures. It indicates that the time cost of signing is smaller than other schemes. And the time cost of verification is smaller than other schemes.

5.2 Experimental Comparison

To evaluate the performance of our scheme in experiments, in this section, we will make simulation experiments with other related schemes which are shown above. The simulation environment in which we deploy the cryptography libraries is Ubuntu 16.04 (default gcc/g++ version 5.4.0) with 8 GB RAM and CPU i7-8750H. All schemes are implemented by using NTL library [35]. We've tested the time cost of algorithms *Samplepre*. It cost about 59.2s. So the time cost of scheme [1,2] is relatively high. The computation overhead for [27] is relatively high, so we omit it. Now we give a concrete description.

Note that when signing l message blocks, the time cost for signing and verification of other schemes is related to l just as Fig. 1 and Fig. 2. Figure 1 shows the time cost of each scheme related to the parameter l. Schemes [1,2] include the algorithm *Samplepre* which is time-consuming, so we omit these two schemes. Figure 2 illustrates that the time cost of verifying l message blocks is constant. For a cloud storage system, users can verify the validity of messages in constant time, so it is extremely suitable for cloud storage verification.

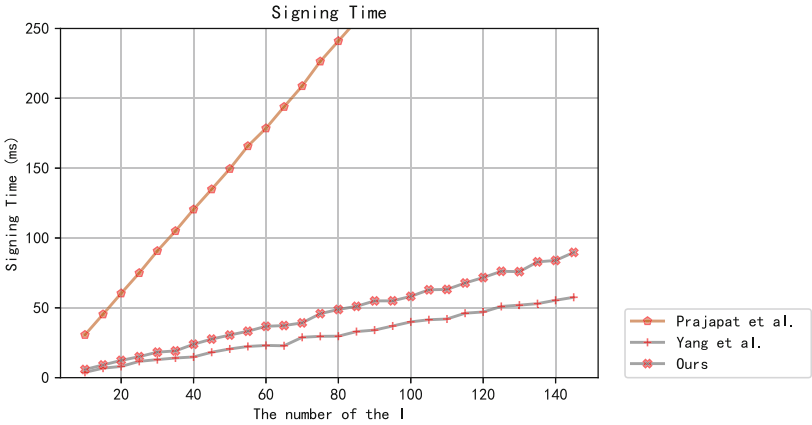

Fig. 1. Time Comparison of Signing.

6 Application

The aggregate signature protocol enables users to check the integrity of the data in the cloud. Besides, we use the certificateless scheme to prevent the cloud from forging signatures. When stored data is lost, the cloud cannot prevent it from concealing or evading responsibility, improving the security and reliability of user-stored data. We briefly describe the application of our algorithm in a cloud storage system as shown in Fig. 3.

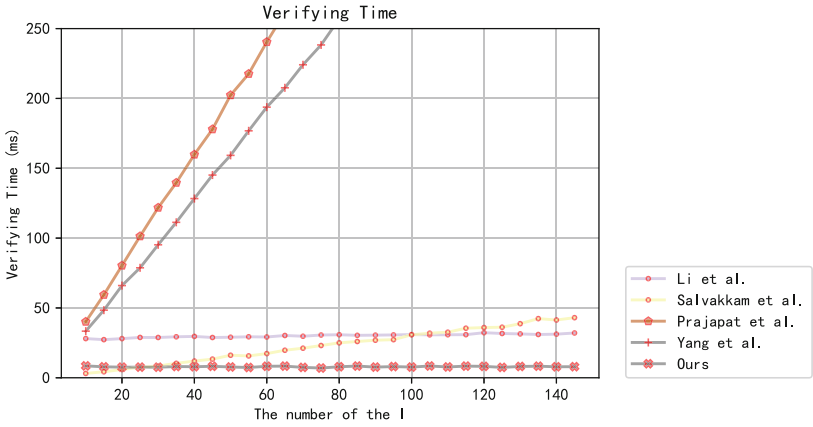

Fig. 2. Time Comparison of Verifying.

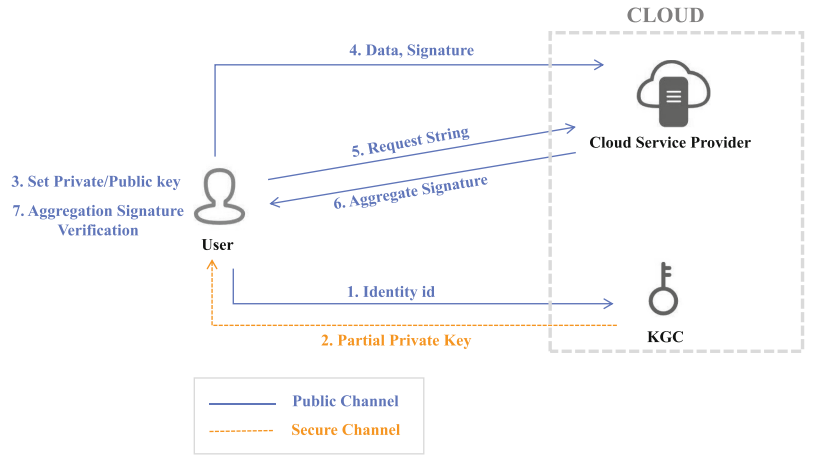

Fig. 3. Workflow of Cloud Storage.

There are two main entities, the cloud user and the Cloud. The user processes a large number of files. Users generate a signature for each data block and then store the two parts (message, signature) in the cloud to release local storage resources. Cloud is mainly used for data storage due to its massive amount of storage space and computation resources. The cloud has two main roles, data storage and partial private key generation.

The cloud storage protocol consists of the following processes.

1. Initialization: The cloud server initializes the system to generate public parameters *params* and master key and it could be KGC to generate the partial private key for the user.

2. Key Generation: Key generation corresponds to step 1,2,3 in the above graph. When the user requires a partial private key for his identity, KGC generates and returns it. Then the user chooses a secret value and generates its secret keys $(\boldsymbol{S_2}, \boldsymbol{P})$. Finally, the user outputs the public key $\{id, \boldsymbol{K}, \boldsymbol{K_2}\}$.
3. Signing and Storage: When the user wants to upload data to the cloud, he'll generate the signature for each message block msg_i, and store the message and the corresponding signature on the cloud as illustrated in the fourth step above. We omit the specific storage methods of data in the cloud, as this is not what users are concerned about.
4. Aggregation: When any user wants to check the validity of messages in the cloud, he generates a string s that each bit s_i of the string corresponds to a message msg_i. Then the cloud server checks the validity of each signature. When all signatures are valid and integral. The server generates and outputs the aggregate signature θ, otherwise, the cloud server informs users of data issues. It corresponds to steps 5,6 in the above graph.
5. Verification: As shown in the 7th step of the above graph, the user checks whether the aggregate signature θ corresponds to his request string s and verifies the signature. If the verification fails, it indicates that an error occurred for the requesting data.

7 Conclusion

We constructed a certificateless signature scheme and certificateless aggregate signature scheme without trapdoors and proved its security in the random oracle model. Besides, we applied this scheme to specific cloud storage system which can ensure data integrity and solves the cloud security problems. After making the experimental comparison, it shows that our scheme is more efficient. And it would be interesting to improve our scheme by adapting it to RLWE.

References

1. Li, H., et al.: PSCPAC: post-quantum secure certificateless public auditing scheme in cloud storage. J. Inf. Secur. Appl. **61**, 102927 (2021). https://doi.org/10.1016/j.jisa.2021.102927
2. Salvakkam, D.B., Pamula, R.: An improved lattice based certificateless data integrity verification techniques for cloud computing. J. Ambient Intell. Humaniz. Comput. **14**, 1868–5145 (2023). https://doi.org/10.1007/s12652-023-04608-7
3. Cash, D.: Bonsai trees, or how to delegate a lattice basis. J. Cryptol. **25**, 601–639 (2012). https://doi.org/10.1007/s00145-011-9105-2
4. Boyen, X.: Lattice mixing and vanishing trapdoors: a framework for fully secure short signatures and more. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 499–517. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13013-7_29
5. Lyubashevsky, V., Nguyen, N.K., Plancon, M., Seiler, G.: Shorter lattice-based group signatures via almost free encryption and other optimizations. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021. LNCS, vol. 13093, pp. 218–248. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-92068-5_8

6. Espitau, T., Tibouchi, M., Wallet, A., Yu, Y.: Shorter hash-and-sign lattice-based signatures. In: Dodis, Y., Shrimpton, T. (eds.) Advances in Cryptology – CRYPTO 2022. CRYPTO 2022. LNCS, vol. 13508, pp. 245–275. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-15979-4_9
7. Gentry, C., Peikert, C., Vaikuntanathan, V.: How to use a short basis: trapdoors for hard lattices and new cryptographic constructions, Electronic Colloquium on Computational Complexity (ECCC), vol. 14, September 2008
8. Abdalla, M., Fouque, P.-A., Lyubashevsky, V., Tibouchi, M.: Tightly-secure signatures from lossy identification schemes. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 572–590. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_34
9. Alwen, C., Peikert, J.: Generating shorter bases for hard random lattices. Theory Comput. Syst. **48** (2011)
10. Lyubashevsky, V.: Lattice signatures without trapdoors. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 738–755. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_43
11. Ducas, L., Durmus, A., Lepoint, T., Lyubashevsky, V.: Lattice signatures and bimodal gaussians. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 40–56. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_3
12. Lyubashevsky, V., Micciancio, D.: Asymptotically efficient lattice-based digital signatures. J. Cryptol. **31**, 774–797 (2018). https://doi.org/10.1007/s00145-017-9270-z
13. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 416–432. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-39200-9_26
14. El Bansarkhani, R., Buchmann, J.: Towards lattice based aggregate signatures. In: Pointcheval, D., Vergnaud, D. (eds.) AFRICACRYPT 2014. LNCS, vol. 8469, pp. 336–355. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-06734-6_21
15. Lu, S., Ostrovsky, R., Shacham, H., Waters, B.: Sequential aggregate signatures, multisignatures, and verifiably encrypted signatures without random oracles. J. Cryptol. **26**, 340–373 (2013)
16. Li, Q., Luo, M., Hsu, C., Wang, L., He, D.: A quantum secure and noninteractive identity-based aggregate signature protocol from lattices. IEEE Syst. J. **16**(3), 4816–4826 (2022). https://doi.org/10.1109/JSYST.2021.3112555
17. Sato, S., Shikata, J.: Identity-based interactive aggregate signatures from lattices. In: Seo, S.H., Seo, H. (eds.) Information Security and Cryptology – ICISC 2022. ICISC 2022. LNCS, vol. 13849, pp. 408–432. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-29371-9_20
18. Boudgoust, K., Roux-Langlois, A.: Compressed linear aggregate signatures based on module lattices, IACR Cryptol. ePrint Arch. (2021). 263. https://eprint.iacr.org/2021/263
19. Zhang, L., Zhang, F.: New certificateless aggregate signature scheme. Comput. Commun. **32**(6), 1079–1085 (2009). cited by: 151. https://doi.org/10.1016/j.comcom.2008.12.042
20. Xiong, H., Guan, Z., Chen, Z., Li, F.: An efficient certificateless aggregate signature with constant pairing computations. Inf. Sci. **219**, 225–235 (2013). https://doi.org/10.1016/j.ins.2012.07.004. https://www.sciencedirect.com/science/article/pii/S0020025512004689

21. Zhang, L., Qin, B., Wu, Q., Zhang, F.: Many-to-one authentication with certificateless aggregate signatures. Comput. Netw. **54**(14), 2482–2491 (2010). cited by: 111. https://doi.org/10.1016/j.comnet.2010.04.008
22. Tian, M., Huang, L.: Certificateless and certificate-based signatures from lattices. Secur. Commun. Netw. **8**(8), 1575–1586 (2015). https://doi.org/10.1002/sec.1105
23. Xie, J., Hu, Y., Gao, J., Jiang, M.: Certificateless sequential aggregate signature scheme on NTRU lattice. Chin. J. Electron. **28**(2), 294–300 (2019). https://doi.org/10.1049/cje.2019.01.019
24. Hung, Y.-H., Tseng, Y.-M., Huang, S.-S.: Lattice-based revocable certificateless signature. Symmetry **9**(10) (2017). https://doi.org/10.3390/sym9100242. https://www.mdpi.com/2073-8994/9/10/242
25. Prajapat, S., Kumar, P., Sharma, V.: An efficient CL-signature scheme over NTRU lattices. In: 2022 4th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N), 2022, pp. 1220–1224 (2022). https://doi.org/10.1109/ICAC3N56670.2022.10074591
26. Yang, Q., Li, D.: Provably secure lattice-based self-certified signature scheme. In: Security and Communication Networks, Hindawi, Cham, pp. 1939–0114 (2021)
27. Yao, Y., Li, Z., Guo, H.: A unified framework of identity-based sequential aggregate signatures from 2-level HIBE schemes. Inf. Sci. **516**, 505–514 (2020). https://doi.org/10.1016/j.ins.2019.12.076
28. Bellare, M., Neven, G.: Multi-signatures in the plain public-key model and a general forking lemma. In: Proceedings of the ACM Conference on Computer and Communications Security, pp. 390–399 (2006).https://doi.org/10.1145/1180405.1180453
29. Pointcheval, D.: Security arguments for digital signatures and blind signatures. J. Cryptol. **13**(3), 361–396 (2000)
30. Micciancio, D., Regev, O.: Worst-case to average-case reductions based on gaussian measures. In: 45th Annual IEEE Symposium on Foundations of Computer Science, pp. 372–381 (2004). https://doi.org/10.1109/FOCS.2004.72
31. Regev, O.: New lattice-based cryptographic constructions. J. ACM **51**(6), 899–942 (2004). https://doi.org/10.1145/1039488.1039490
32. Micciancio, D.: Almost perfect lattices, the covering radius problem, and applications to Ajtai's connection factor. SIAM J. Comput. **34**, 118–169 (2004)
33. Banaszczyk, W.: New bounds in some transference theorems in the geometry of numbers. Math. Ann. **296**, 625–635 (1993)
34. Peikert, C., Rosen, A.: Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 145–166. Springer, Heidelberg (2006). https://doi.org/10.1007/11681878_8
35. Ntl (2021). https://www.shoup.net/ntl