Cybersecurity

RESEARCH Open Access

A fully dynamic forward-secure group signature from lattice

Zhijian Liao[1], Qiong Huang[1,2*] and Xinjian Chen[1]

Abstract

A forward-secure group signature (FSGS) ensures the unforgeability of signatures in the past time period despite signing secret key is leaked in the current time period. As we know, traditional FSGS schemes are mostly relying on number-theoretic assumptions unable to resist quantum attacks. Therefore, we present an efficient lattice-based fully dynamic (i.e. users can flexibly join or quit the group) forward-secure group signature (DFSGS) by combining an improved version of FSGS scheme proposed by Ling. Based on an efficient zero-knowledge argument, we construct argument of knowledge of the committed value and the plaintext that help with privacy protection. Our DFSGS scheme is proved to be anonymous and forward-secure traceable relying on short integer solution and learning with errors assumptions in random oracle model. Moreover, the lengths of group public key and signature of our DFSGS scheme have been improved, and the length of user secret key has no connection with the quantity of group members.

Keywords: Dynamic group signature, Forward-secure, Lattice, SIS, LWE, Zero-knowledge argument

Introduction

Group signature

With the rapid development of informatization, ordinary digital signatures can no longer meet the requirements of both authentication and privacy protection in the secure authentication protocol under the large environment of e-commerce and e-government. Subsequently, people began to research and construct other signature schemes that can meet some special requirements or properties on the basis of ordinary digital signatures. The group signature(GS) was formally proposed by Chaum and Van Heyst (1991) in 1991. Traditional group signature schemes usually require two properties: anonymity and traceability. Anonymity signifies that legal users sign the message representing the whole group, and the signer's identity is unknown to the verifier when verifying the validity of the signature. Traceability means that when a signature is disputed, the group manager could find out the signer's identity through the tracing secret key. Then the stronger security of full anonymity and full traceability was proposed by Bellare et al. (2003).

As we know, the initial group signature is static, that is, once the group system is established, new users cannot join the group. If a new user needs to be added, the group system must reinitialize the group public key and signing secret key. Meanwhile, group systems often need to add new users frequently in practical applications. Therefore, static group signatures are unsuitable for practical applications. At the same time, static group signing schemes cannot revoke group members. In group signature schemes, it is a difficult problem to realize the revocation of group members. A group manager cannot prohibit revoked group member from continuing to sign with his secret key. Therefore, it is necessary to have an effective verification algorithm and group member revocation mechanism, so that the signatures generated by the members that have not been revoked can pass the verification algorithm, while the signatures generated by the revoked members cannot pass the verification algorithm. Currently, the revocation methods of group

*Correspondence: qhuang@scau.edu.cn
[1] College of Mathematics and Informatics, South China Agricultural University, 483 Wushan Road, Guangzhou 510642, China
Full list of author information is available at the end of the article

signature mainly include the following: (1) update all the keys; (2) dynamic accumulators(DA) based (Camenisch and Lysyanskaya 2002; Nguyen 2005); (3) verifier-local revocation(VLR) model (Boneh and Shacham 2004); (4) broadcast encryption(BE) based (Libert et al. 2012a, b); (5) time period based update scheme of authorization (Song 2001). Among these methods, VLR has received wide attentions. VLR means that revocation messages are only sent to signature verifiers, who then checks the validity of the signature locally without contacting the individual signers when some user is revoked. Since then, many researchers had conducted in-depth research on dynamic group signatures (i.e. support joining and revocation mechanisms). Most group signature schemes that consider both function and security, on the other hand, have an issue with inefficiency when group members join and be revoked. As a result, group signature researches based on traditional difficulty assumptions are focused on building an efficient group signature that takes both usefulness and security into account. In the meantime, the features of group signature make it applicable in many privacy protection scenarios such as anonymous electronic voting, electronic currency and trusted computing.

Related work

In the past, most group signatures were relied on the traditional difficulty assumptions such as large integer factorization and discrete logarithms. In 1994, Shor (1994) proposed a quantum algorithm putting the security of traditional cryptographic schemes in jeopardy. Ajtai (1996) pioneered the proof that the difficulty of the lattice problem in the average case is the same as the difficulty in the worst case in 1996. This advancement establishes a theoretical foundation for the design of lattice-based cryptosystem. With the development of quantum computing, traditional group signatures (Ateniese et al. 2000; Bellare et al. 2003; Boneh and Shacham 2004; Boneh et al. 2004; Kiayias et al. 2004; Bellare et al. 2005; Kiayias and Yung 2006; Boyen and Waters 2006; Groth 2006; Boyen and Waters 2007; Groth 2007) based on number-theoretic assumptions can no longer resist quantum attacks. The lattice-based cryptosystem has gradually gained popularity as a research topic in the post-quantum cryptographic era owing to its simple structure and security against quantum computing attacks.

Gordon et al. (2010) proposed the first lattice-based GS in Asiacrypt 2010. After that, some schemes have been proposed successively, which have improved efficiency. Many of them (Laguillaumie et al. 2013; Ling et al. 2015; Nguyen et al. 2015) aim to make the key and signature smaller, varying from a linear relationship with the quantity of group members to a logarithmic relationship, and then independent of the quantity of group members. Since users need flexibly join a group and group manager should have the right to revoke illegal users when they are found to be misbehaving, group signature should support the dynamicity of group users joining and revocation mechanisms. In 2016, Libert et al. (2016) constructed a GS with a joining mechanism, but does not support dynamic revocation of group users. The first fully dynamic GS from lattice was introduced by Ling et al. (2017) which is based on Merkle hash tree. But it needs update the Merkle hash tree when revoking users and the calculation is more complicated and time-consuming. A GS scheme based on lattice that supports verifier-local revocation was put forward by Ling et al. (2018) in 2018. But it does not support the dynamic joining of group users.

Sometimes secret key leakage is unavoidable, but the loss caused by key leakage can be reduced using forward security technology. To ensure the signature's security after signing secret key is leaked, Song (2001) proposed the first FSGS scheme. Ling et al. (2019) had constructed the first FSGS scheme based on lattice. But this scheme is a static group signature, and the secret key of time period is updated through the hierarchical structure of the Bonsai tree and node select algorithm. In 2020, Kansal et al. (2020) put forward the first lattice-based dynamic forward-secure GS scheme. It includes an updating algorithm based on Hamming weight of node select algorithm. However, both of the member certificate and secret key need be updated, and the length of the public key and signature needs be further optimized.

To the best of our knowledge, there are few lattice-based GS schemes achieve both dynamics and forward security which is provably secure in RO model.

Our contributions

Aiming at the problem of insufficient dynamic flexibility and inefficiency of the existing part of the work, we propose an efficient fully dynamic forward-secure group signature from lattice with the following contributions.

1. Our DFSGS scheme achieves forward security while achieving dynamics to ensure security after the key is exposed. Compared with other dynamic group signature schemes, our DFSGS scheme is supporting forward security while the length of public key ($\tilde{\mathcal{O}}(\lambda^2 \cdot d + \lambda \cdot L)$) and secret keys ($\tilde{\mathcal{O}}(\lambda \cdot d^3)$) are not increased much, where L and d are logarithmically related to the quantity of group members and the time periods, respectively. Moreover, the length of secret key has no connection with the quantity of group members.

2. Due to the combination of efficient zero-knowledge arguments of knowledge for linear equations from Yang et al. (2019) which makes the soundness error of our scheme reach 1/poly, it can significantly reduce the times of repetition for the protocol. Thereby, we reduce the length of the signature and improve the protocol's verification efficiency. Compared with the first lattice-based dynamic FSGS scheme (Kansal et al. 2020), our DFSGS scheme's length of group public key and signature has been greatly improved.
3. Our scheme supports to revoke members through verifier-local revocation. User's revocation token will be changed when his secret key is updated, and the token in the revocation list will also be updated. If group member need to restore the user's legal identity, just remove his revocation token from the revocation list. This also implies that a user can be revoked in time period t_1 and regain the legal authority in the time period $t_2 (t_2 > t_1)$ without resetting the public key and secret key.

Application to internet of vehicles With the continuous development of the Internet of Vehicles (IoV), privacy protection has become more and more important. The DFSGS can achieve the privacy protection of vehicles in the vehicle network, and the dynamics of the group signature ensures the dynamic joining and revoking of vehicles. The system model of the Internet of Vehicles mainly includes: Trusted Authority(TA), Road Side Unit(RSU) fixed on the roadside and On Board Unit(OBU) loaded on the vehicles. The TA has sufficient communication, computation and storage capacity and interacts with the RSUs. It is responsible for the registration and revocation of vehicle users, and also acts as the tracing manager when any abnormal behavior is detected. The RSUs act as the group manager and are responsible for managing the group consisting of vehicle users. The OBUs act as the vehicle users to represent the members in the group. In the stage of OBU joining the group, OBU submits its own identity information to the group manager, who subsequently issues certificates and secret key to the vehicle for authentication after passing. In the cooperation phase, the vehicle members use their own secret key to sign their information, and use inductive sensors to send it to nearby vehicle units to realize cooperative driving with the vehicle. In the message verification phase, other vehicles receiving the information judge whether the vehicle has been revoked according to the revocation list, and then check the validity of the message without knowing the true identity of the message sender, so as to realize anonymous communication between vehicles. When a vehicle member publishes a false message that causes a traffic accident or a dispute, the tracing manager uses the tracing key to open the signature of the message and holds the corresponding vehicle member accountable. After that, the corresponding vehicle members will be added to the revocation list and be punished for a corresponding time period. After waiting for the penalty time to expire, the group manager may consider to remove them from the revocation list and reset them as legal members (Fig. 1).

Preliminaries

Notations

In the whole paper, $||\mathbf{a}||$ denotes the l_2-norm of a vector $\mathbf{a}$, and $||\mathbf{b}||_\infty$ denotes the l_∞-norm of a vector $\mathbf{b}$. Let $(\mathbf{a}||\mathbf{b})$ denote two vectors' horizontal concatenation, $(\mathbf{A}|\mathbf{B})$ denotes the concatenation of two matrices $\mathbf{A} \in \mathbb{R}^{n\times m_1}$ and $\mathbf{B} \in \mathbb{R}^{n\times m_2}$. $[N]$ indicates the set of integer $\{1, \ldots, N\}$. For matrix $\mathbf{I_n} = \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{bmatrix}_{n\times n}$ and $\mathbf{g} = (1\ 2\ \ldots\ 2^{\lfloor \log q \rceil - 1})$, we define $\mathbf{G} = \mathbf{I_n} \bigotimes \mathbf{g} = \begin{bmatrix} 1\ 2 \ldots 2^{\lfloor \log q \rceil -1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1\ 2 \ldots 2^{\lfloor \log q \rceil -1} \end{bmatrix}$ where $\mathbf{G}$ is a matrix of n times m and $m = n \cdot \lfloor \log q \rceil$.

Lattice

Let linearly independent vectors $\mathbf{b}_i \in \mathbb{R}^n$ for $i \in [m]$ and matrix $\mathbf{B} = [\mathbf{b}_1||\ldots||\mathbf{b}_m] \in \mathbb{Z}_q^{n\times m}$. The n-dimension lattice generated by $\mathbf{B}$ is denoted by $\Lambda(\mathbf{B}) = \sum_{i=1}^{m} \mathbf{b}_i x_i : x_i \in \mathbb{Z}$. The set of vectors $\{\mathbf{b}_1, \ldots, \mathbf{b}_m\}$ represents a basis of $\Lambda(\mathbf{B})$ and m is the rank of $\Lambda(\mathbf{B})$.

The security proof of our DFSGS scheme relies on the lattice hard assumptions(SIS problem and LWE problem) defined below.

Definition 1 (*SIS Problem* Ajtai 1996; Gentry et al. 2008; Peikert 2015; Micciancio and Peikert 2013) For matrix $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$, the $\mathrm{SIS}^\infty_{n,m,q,\beta}$ problem is to find a non-zero vector $\mathbf{e} \in \mathbb{Z}^m$ satisfying that $\mathbf{A} \cdot \mathbf{e} = 0 \mod q$ and $||\mathbf{e}||_\infty \le \beta$.

Let $m, \beta = poly(n)$, and for any $q > \beta\sqrt{n}$, the $\mathrm{SIS}^\infty_{n,m,q,\beta}$ problem is at least as difficult as the worst-case SIVP_γ problem for some $\gamma = \beta \cdot \tilde{O}(\sqrt{n})$.

Definition 2 (*LWE Problem* Peikert 2015; Regev 2009) $\mathrm{LWE}_{n,m,q,\chi}$ is parametrized by positive integers n, $m \ge 1$, $q \ge 2$ and probability distribution χ. For $\mathbf{s} \in \mathbb{Z}_q^n$ and χ, the distribution $A_{s,\chi}$ is obtained by sampling $\mathbf{a} \xleftarrow{\$} \mathbb{Z}_q^n$ and $e \xleftarrow{\$} \chi$, it outputs the pair $(\mathbf{a}, c = \mathbf{a}^\top \cdot \mathbf{s} + e)$. When m vectors $\mathbf{a}$ are selected and $\mathbf{e}'$ is selected from

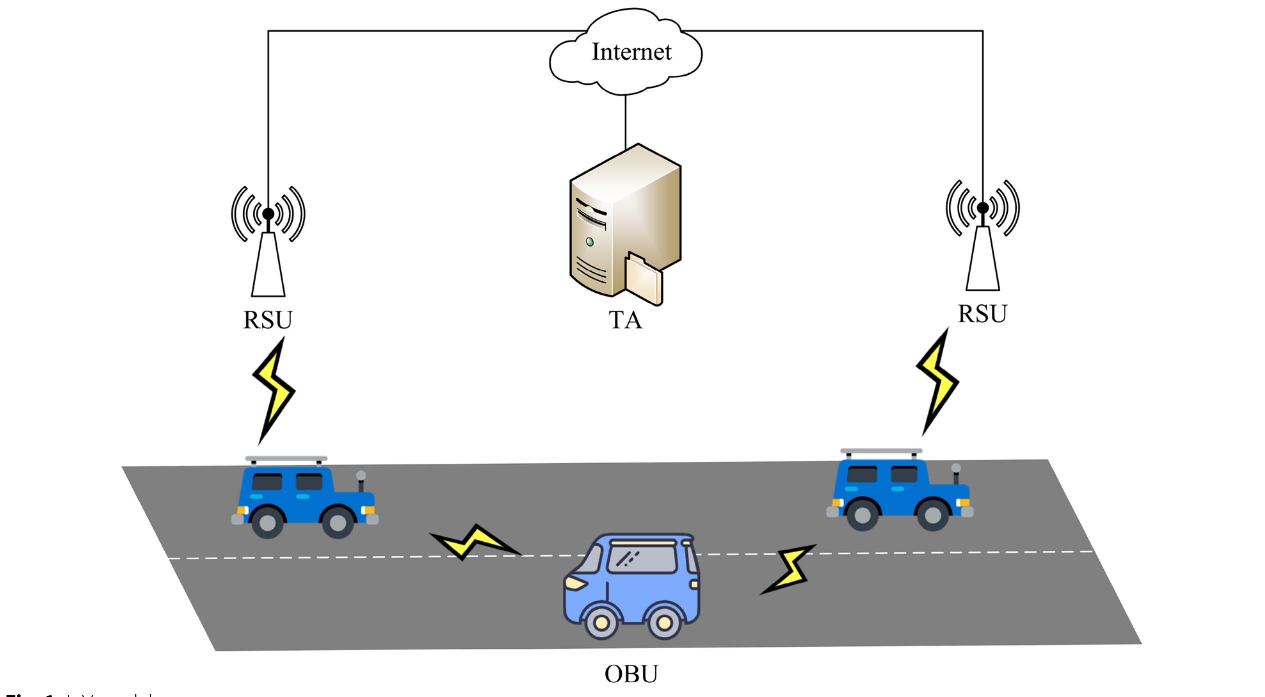

Fig. 1 IoV model

the distribution χ^m, for $\mathbf{s}' \in \mathbb{Z}_q^n$ and $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, the LWE probability distribution can be expressed as $A_{s',\chi^m} = (\mathbf{A}, \mathbf{C} = \mathbf{A}^\top \cdot \mathbf{s}' + \mathbf{e}')$.

There are two kinds of LWE problems. The *search* $-\,\mathrm{LWE}_{n,m,q,\chi}$ problem refers to the problem of discovering the secret vector s given $(\mathbf{A}, \mathbf{C})$, while the *decision* $-\,\mathrm{LWE}_{n,m,q,\chi}$ problem requires distinguishing the LWE sample from the uniform sample. The security of our scheme relies on the difficulty of the latter.

For $q \geq 2$, $\beta \geq \sqrt{n}\omega(\log n)$ and distribution χ, the $\mathrm{LWE}_{n,m,q,\chi}$ problem is as hard as $\mathrm{SIVP}_{\tilde{\mathcal{O}}(nq/\beta)}$.

Lemma 1 (TrapGen Algorithm Alwen and Peikert 2009) *Given positive integer n, $q = poly(n)$ and $m = \mathcal{O}(n \log q)$. This algorithm $TrapGen(1^n, 1^m, q)$ outputs $\mathbf{B} \in \mathbb{Z}_q^{n\times m}$ and its trapdoor $\mathbf{T_B}$ for $\Lambda_q^\perp(\mathbf{B})$ satisfying $\mathbf{B} \cdot \mathbf{T_B} = 0 \mod q$. The matrix $\mathbf{B}$ is uniformly distributed in $\mathbb{Z}_q^{n\times m}$ and $||\mathbf{T_B}|| \leq \mathcal{O}(\sqrt{n \log q})$.*

Lemma 2 (SampleD Algorithm Gentry et al. 2008) *Given a lattice $\Lambda^\perp(\mathbf{B})$ and its basis $\mathbf{T_B}$ where $\mathbf{B} \in \mathbb{Z}_q^{n\times m}$ and $\mathbf{T_B} \in \mathbb{Z}_q^{m\times m}$. Let $s \geq \omega(\sqrt{\log n})$ and $\mathbf{u} \in \mathbb{Z}_q^n$. The algorithm $SampleD(\mathbf{B}, \mathbf{T_B}, \mathbf{u}, s)$ which outputs $\mathbf{x} \in \Lambda^u(\mathbf{B})$ sampled from the distribution $D_{\mathbb{Z}^m,s}$ satisfying $\mathbf{B} \cdot \mathbf{x} = \mathbf{u} \mod q$.*

Lemma 3 (ExtBasis Algorithm Cash et al. 2010) *Given $\mathbf{B}' \in \mathbb{Z}^{n\times m'}$, a lattice $\Lambda^\perp(\mathbf{B})$ and its basis $\mathbf{T_B}$ where $\mathbf{B} \in \mathbb{Z}_q^{n\times m}$ and $\mathbf{T_B} \in \mathbb{Z}_q^{m\times m}$, there is a PPT algorithm $ExtBasis(\mathbf{T_B}, \mathbf{B}|\mathbf{B}')$ which generates the basis $\mathbf{T_{B|B'}}$ of $\Lambda^\perp(\mathbf{B}|\mathbf{B}')$ with $||\mathbf{T_B}|| = ||\mathbf{T_{B|B'}}||$ satisfying $(\mathbf{B}|\mathbf{B}') \cdot \mathbf{T_{B|B'}} = 0 \mod q$.*

Lemma 4 (RandBasis Algirithm Cash et al. 2010) *There is a PPT algorithm $RandBasis(\mathbf{B}, \mathbf{T_B}, \sigma)$ which takes a basis $\mathbf{T_B}$ of $\Lambda(\mathbf{B})$ a parameter $\sigma \geq ||\tilde{\mathbf{T}}_\mathbf{B}|| \cdot \omega\sqrt{\log n}$ as input, returns a new basis $\mathbf{T}'_\mathbf{B}$ of Λ with $||\mathbf{T}'_\mathbf{B}|| \leq \sigma \cdot \sqrt{m}$. If $\mathbf{T_B}$ and $\mathbf{T}'_\mathbf{B}$ are two different bases for the same Λ and $\sigma \geq max\{||\tilde{\mathbf{T}}_\mathbf{B}||, ||\tilde{\mathbf{T}}'_\mathbf{B}||\} \cdot \omega(\sqrt{\log n})$, the bases $\mathbf{T_B}$ and $\mathbf{T}'_\mathbf{B}$ are statistically close.*

Bonsai tree signature scheme and node select algorithm

Bonsai tree: Bonsai tree can also be called a hierarchy of trapdoor functions. In a lattice-based instantiation of Bonsai tree, the root node constructed by a lattice can generate a new lattice with lager dimension as the branch node of next level and its basis. The growth from root to branch includes undirected growth without trapdoor and controlled growth with trapdoor. The controlled growth also requires extending control and randomizing control. Extending control extends the basis $\mathbf{T}_{\mathbf{A}}$ of $\Lambda_q^{\perp}(\mathbf{A})$ to the basis $\mathbf{T}_{\mathbf{A}'}$ of $\Lambda_q^{\perp}(\mathbf{A}')$ with larger dimension without increasing the norm. Randomizing control ensures that $\mathbf{T}_{\mathbf{A}}$ and $\mathbf{T}_{\mathbf{A}'}$ are independent of each other.

Bonsai tree is widely used in signature schemes. The bonsai tree signature scheme (Cash et al. 2010) is given as follows.

Setup. Take security parameter λ as input, $n = \mathcal{O}(\lambda)$, $m = \mathcal{O}(n \log q)$ and integer $q = poly(n)$. Set l be message's length, $\tilde{m} = m(l+1)$ and $s = \mathcal{O}(\sqrt{n \log q}) \cdot \omega(\log n)$.

KeyGen. Run algorithm *TrapGen* to obtain $(\mathbf{A}_0, \mathbf{S}_0)$ with $||\mathbf{S}_0|| \leq \mathcal{O}(\sqrt{n \log q})$. Sample $\mathbf{A}_i^b \overset{\$}{\leftarrow} \mathbb{Z}_q^{n \times m}$ for $b \in \{0,1\}$ and $i \in [l]$. Output verification public key $vk = (\mathbf{A}_0, \mathbf{A}_i^b)$, secret key $sk = \mathbf{S}_0$.

Sign. Let message $\mu = (\mu_1, \ldots, \mu_l) \in \{0,1\}^l$, $\mathbf{A}_\mu = (\mathbf{A}_0 | \mathbf{A}_1^{\mu_1} | \ldots | \mathbf{A}_l^{\mu_l})$.

$$\boldsymbol{\sigma} \leftarrow SampleD(ExtBasis(\mathbf{S}_0, \mathbf{A}_\mu), 0, s).$$

The signer outputs signature $\boldsymbol{\sigma}$.

Verify. If $\boldsymbol{\sigma} \neq 0$, $||\boldsymbol{\sigma}|| \leq s\sqrt{\tilde{m}}$ and $\mathbf{A}_\mu \cdot \boldsymbol{\sigma} = 0 \mod q$, return true. Otherwise, return false.

Node select algorithm: For each user, there is a binary tree which represents the composition of the time period. Each leaf node represents the time period of each key. Assume that the maximum lifetime is partitioned into $T = 2^d$ discrete periods for $d \in \mathbb{Z}^+$. These time periods are represented using a binary tree which are associated with leaf node where time period $t \in \{0, 1, 2, \ldots, T-1\}$, and each node of the binary tree at depth k is associated with a binary number z of length $k-1$. Following (Boyen et al. 2006), we define the node's "second sibling at depth k" *sibling*(k, t) for $k \in [1, d+1]$ and $t \in [0, T-1]$ as follows.

From depth 1 to d+1, we sequentially number the nodes from 0 and increase by 1 with binary from left to right. For the target leaf node t, we can figure out the path nodes from the leaf node to the root node. For each path node from depth 1 to d, if it exits right neighbor, then set *sibling*(k, t) to its corresponding number, and set $sibling(k,t) = \perp$, otherwise. Finally, set $sibling(d+1, t) = Bin(t)$ and generate the node set $Nodes_{(t \to T-1)}$ which is composed of $\{sibling(k,t)\}$. Then we can generate all path nodes after the current time period by node set (Fig. 2).

Efficient lattice-based zero-knowledge arguments

The basic protocol

In an efficient zero-knowledge argument of knowledge(ZKAoK) (Yang et al. 2019), which has the quadratic constraints, it has standard soundness and soundness error 1/poly, which is much smaller than the soundness error 2/3 of stern-type protocol.

Define m, n, l and θ be positive integers, respectively, q be prime number, $\mathcal{C}$ be challenge space. Define matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, vector $\mathbf{x} \in \mathbb{Z}_q^n$, $\mathbf{y} \in \mathbb{Z}_q^m$, and $\mathcal{M}$ is represented as a collection of l 3-tuples, each containing three integers in the range $[1, n]$. The relation $\mathcal{R}$ as follows:

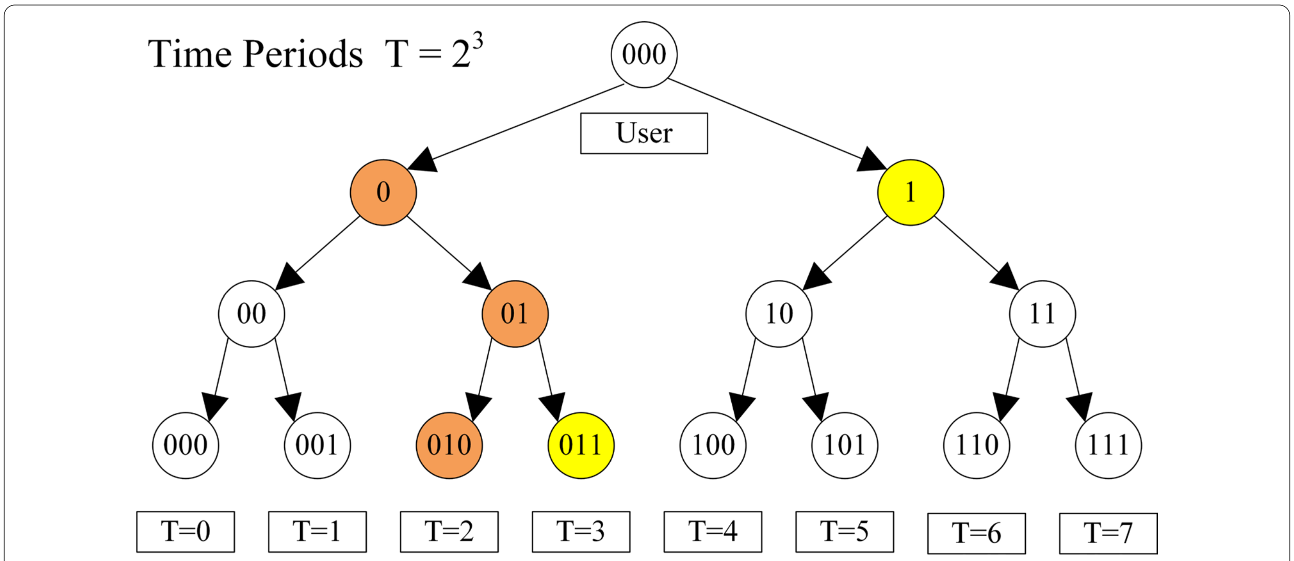

Fig. 2 For example, there is a tree with $2^3 = 8$ leaves representing 8 time periods. For each level in the path of the target node (010) which stands for time periods $T = 2$, we add its right sibling to the set $Nodes_{(2 \to 7)}$. Then we have $Nodes_{(2 \to 7)} = \{(1)^\top, \perp, (011)^\top, (010)^\top\}$

$$\mathcal{R} = \{(\mathbf{A}, \mathbf{y}, \mathcal{M}), (\mathbf{x}) : \mathbf{A} \cdot \mathbf{x} = \mathbf{y} \wedge \forall (h,i,j) \in \mathcal{M}, \mathbf{x}[h] = \mathbf{x}[i] \cdot \mathbf{x}[j]\}.$$

The following is a description of the basic ZKAoK protocol. Firstly, the verifier receives the vector $\mathbf{t} = \mathbf{A} \cdot \mathbf{r}$ sent by prover. Then the prover computes $\mathbf{z} = \alpha \cdot \mathbf{x} + \mathbf{r}$ to the verifier where $\alpha \in \mathcal{C}$ is sampled by the verifier. Finally, the verifier determines whether the condition $\mathbf{A} \cdot \mathbf{z} = \alpha \cdot \mathbf{y} + \mathbf{t}$ is met. As for proving quadratic constraints over the witnesses, it can be transform into prove that the quadratic polynomial is linear in α. Furthermore, the homomorphic commitment scheme is employed to hide information about witness.

The basic protocol can transform into non-interactive zero-knowledge arguments of knowledge(NIZKAoK) via Fiat-Shamir transform and reach negligible soundness error by repetition. The increase in completeness error caused by repeating protocol is avoided by rejecting sampling techniques. The specific construction of protocol and security proof follows directly from Yang et al. (2019) relying on the SIS and LWE assumptions, and we omit the details here.

Our zero-knowledge AoK of committed values

We construct an AoK of committed values for the commitment scheme. There is a ZKAoK that proves the knowledge of

$$\{\mathbf{w}_1, \mathbf{w}_2, \mathbf{s} \in \mathbb{Z}_q^m, \mathbf{e} \in \mathbb{Z}_q^n;\ ||\mathbf{w}_1||_\infty, ||\mathbf{w}_2||_\infty, ||\mathbf{s}||_\infty \leq \theta\}$$

that satisfies the following relation:

$$\mathbf{A}' \cdot \mathbf{w}_1 + \mathbf{A}'' \cdot \mathbf{w}_2 = \mathbf{c}_1 \mod q$$
$$\mathbf{B} \cdot \mathbf{s} + \mathbf{e} = \mathbf{c}_2 \mod q$$

where $\mathbf{A}'$,$\mathbf{A}''$,$\mathbf{B} \in \mathbb{Z}_q^{n\times m}$ and $\mathbf{c}_1, \mathbf{c}_2 \in \mathbb{Z}_q^n$.

We define $\mathbf{w}_1' = \mathbf{w}_1 + \boldsymbol{\theta}_1$, $\mathbf{w}_2' = \mathbf{w}_2 + \boldsymbol{\theta}_1$, $\mathbf{s}' = \mathbf{s} + \boldsymbol{\theta}_1$ and $\mathbf{e}' = \mathbf{e} + \boldsymbol{\theta}_2$ where $\boldsymbol{\theta}_1 = (\theta\ \theta\ ...\ \theta) \in \mathbb{Z}_q^m$ and $\boldsymbol{\theta}_2 = (\theta\ \theta\ ...\ \theta) \in \mathbb{Z}_q^n$. For vectors $\mathbf{w}_1'$, $\mathbf{w}_2'$, $\mathbf{s}'$ and $\mathbf{e}'$, we use vector decomposition techniques to decompose them into binary representations. Let $\mathbf{g} = (\lfloor(2\theta + 2^{1-1})/2^1\rfloor||...||\lfloor(2\theta + 2^{k-1})/2^k\rfloor)$ where $k = \lfloor \log 2\theta \rfloor + 1$, and define the gadget matrices $\mathbf{G}_1 = \mathbf{I}_m \bigotimes \mathbf{g}$ and $\mathbf{G}_2 = \mathbf{I}_n \bigotimes \mathbf{g}$.

Therefore, vectors $\mathbf{w}_1'$, $\mathbf{w}_2'$, $\mathbf{s}'$ and $\mathbf{e}'$ can be represented as binary vectors $\hat{\mathbf{w}}_1$, $\hat{\mathbf{w}}_2$, $\hat{\mathbf{s}}_1$ and $\hat{\mathbf{e}}_1$ which satisfy that $\mathbf{w}_1' = \mathbf{G}_1 \cdot \hat{\mathbf{w}}_1$, $\mathbf{w}_2' = \mathbf{G}_1 \cdot \hat{\mathbf{w}}_2$, $\mathbf{s}' = \mathbf{G}_1 \cdot \hat{\mathbf{s}}_1$ and $\mathbf{e}' = \mathbf{G}_2 \cdot \hat{\mathbf{e}}_1$. Finally, we set $\mathbf{A} = \begin{pmatrix} \mathbf{A}'\mathbf{G}_1 & \mathbf{A}''\mathbf{G}_1 & 0 & 0 \\ 0 & 0 & \mathbf{B}\mathbf{G}_1 & \mathbf{G}_2 \end{pmatrix}$, $\mathbf{x} = (\hat{\mathbf{w}}_1^\top\ \hat{\mathbf{w}}_2^\top\ \hat{\mathbf{s}}_1^\top\ \hat{\mathbf{e}}_1^\top)^\top$,$\mathbf{y} = \begin{pmatrix} \mathbf{c}_1 + (\mathbf{A}' + \mathbf{A}'') \cdot \boldsymbol{\theta}_1 \\ \mathbf{c}_2 + \mathbf{B} \cdot \boldsymbol{\theta}_1 + \boldsymbol{\theta}_2 \end{pmatrix}$ and $\mathcal{M} = \{(h,h,h)\}_{h\in[1,(3m+n)\cdot k]}$. The new form $(\mathbf{A}, \mathbf{x}, \mathbf{y}, \mathcal{M})$ is the same as original relation. Therefore, we can prove knowledge of secret committed values satisfying correlation equations.

Our zero-knowledge AoK of plaintext

Here, we introduce a ZKAoK of the plaintext for the encryption scheme from Yang et al. (2019). Let l_1, l_2 and L be positive integers. Define the relation $\mathcal{R}_{enc}$ as follows.

$$\begin{aligned}\mathcal{R}_{enc} = \{&(\mathbf{B}_1, \mathbf{B}_2, \mathbf{c}_1, \mathbf{c}_2), (\mathbf{r}, \mathbf{e}_1, \mathbf{e}_2, \mathbf{w}) \\ &\in (\mathbb{Z}_q^{l_1\times l_2} \times \mathbb{Z}_q^{L\times l_2} \times \mathbb{Z}_q^{l_1} \times \mathbb{Z}_q^{L}) \times (\mathbb{Z}_q^{l_2} \times \mathbb{Z}_q^{l_1} \times \mathbb{Z}_q^{L} \times \{0,1\}^L) : \\ &||\mathbf{r}||_\infty, ||\mathbf{e}_1||_\infty, ||\mathbf{e}_2||_\infty \leq \theta \wedge \mathbf{B}_1 \cdot \mathbf{r} + \mathbf{e}_1 = \mathbf{c}_1 \\ &\wedge \mathbf{B}_2 \cdot \mathbf{r} + \mathbf{e}_2 + \lfloor q/2 \rceil \cdot \mathbf{w} = \mathbf{c}_2\}.\end{aligned}$$

We convert relation $\mathcal{R}_{enc}$ to relation $\mathcal{R}$ by following these steps. First, we define vectors $\boldsymbol{\theta}_x = (\theta\ \theta\ ...\ \theta)^\top \in \mathbb{Z}_q^{l_2}$, $\boldsymbol{\theta}_y = (\theta\ \theta\ ...\ \theta)^\top \in \mathbb{Z}_q^{l_1}$, $\boldsymbol{\theta}_z = (\theta\ \theta\ ...\ \theta)^\top \in \mathbb{Z}_q^{L}$ and define $\mathbf{r}' = \mathbf{r} + \boldsymbol{\theta}_x$, $\mathbf{e}_1' = \mathbf{e}_1 + \boldsymbol{\theta}_y$ and $\mathbf{e}_2' = \mathbf{e}_1 + \boldsymbol{\theta}_z$. According to the binary decomposition method, for integer $a \in [0, 2\theta]$, there is a set of sequences $\theta_1, \theta_2, ..., \theta_p$ in which $\sum_{j=1}^{p} u_j\theta_j = a$, where $\mathbf{u} = (u_1\ u_2\ ...\ u_j) \in \{0,1\}^p, p = \lfloor \log 2\theta \rfloor + 1, \theta_1 = \lceil \theta/2 \rceil$, $\theta_2 = \lceil(\theta - \theta_1)/2\rceil, \theta_3 = \lceil(\theta - \theta_1 - \theta_2)/2\rceil, ..., \theta_p = 1$.

Next, we convert vectors $\mathbf{r}'$, $\mathbf{e}_1'$, $\mathbf{e}_2'$ to binary vectors $\hat{\mathbf{r}}$, $\hat{\mathbf{e}_1}$, $\hat{\mathbf{e}_2}$ by binary decomposition. Let $\mathbf{g} = (\theta_1||\theta_2||...||\theta_p)$ and define gadget matrices $\mathbf{G}_1 = \mathbf{I}_{l_2} \bigotimes \mathbf{g}$, $\mathbf{G}_2 = \mathbf{I}_{l_1} \bigotimes \mathbf{g}$, $\mathbf{G}_3 = \mathbf{I}_L \bigotimes \mathbf{g}$ which satisfy $\mathbf{G}_1 \cdot \hat{\mathbf{r}} = \mathbf{r}'$, $\mathbf{G}_2 \cdot \hat{\mathbf{e}_1} = \mathbf{e}_1'$, $\mathbf{G}_3 \cdot \hat{\mathbf{e}_2} = \mathbf{e}_2'$.

Finally, we set $\mathbf{A} = \begin{pmatrix} \mathbf{B}_1 \cdot \mathbf{G}_1 & \mathbf{G}_2 & 0 & 0 \\ \mathbf{B}_2 \cdot \mathbf{G}_1 & 0 & \mathbf{G}_3 & \lfloor q/2 \rceil \cdot \mathbf{I}_L \end{pmatrix}$, $\mathbf{x} = (\hat{\mathbf{r}}^\top\ \hat{\mathbf{e}}_1^\top\ \hat{\mathbf{e}}_2^\top\ \mathbf{w}^\top)^\top$, $\mathbf{y} = \begin{pmatrix} \mathbf{c}_1 + \mathbf{B}_1 \cdot \boldsymbol{\theta}_1 + \boldsymbol{\theta}_2 \\ \mathbf{c}_2 + \mathbf{B}_2 \cdot \boldsymbol{\theta}_1 + \boldsymbol{\theta}_3 \end{pmatrix}$ and $\mathcal{M} = \{(h,h,h)\}_{h\in[1,(l_1+l_2+L)\cdot k+L]}$. Similarly, we can prove knowledge of secret plaintext satisfying correlation equations.

Dynamic forward-secure group signature

Definition

A fully DFSGS scheme is composed of eight algorithms (*GSetup*, *GKgen*, ⟨*GUJoin*, *GIssue*⟩, *GSign*, *GRevoke*, *KeyUpdate*, *GVerify*, *GOpen*) described as follows:

GSetup(λ) $\rightarrow$ **pp**. On input λ, this algorithm outputs public parameter *pp*.

GKgen(**pp**) $\rightarrow$ ((msk,mpk),(tsk,tpk)). On input *pp*, group manager GM generates group manager's key pair (*msk*, *mpk*), and tracing manager TM generates tracing manager's tracing key pair (*tsk*, *tpk*). GM initializes registration table *reg*, revocation list *RL* and token list *TL*. We define the group public key $gpk = (pp, mpk, tpk)$.

$\langle$ GUJoin($\mathbf{token_{sk}}$,t) $\rightarrow$ (cert,$\mathbf{usk_t[i]}$),GIssue(gpk,msk,t) $\rightarrow$ reg $\rangle$. The interactive protocol is performed by the GM and user. A new user i first generates revocation $token_{sk}$ at time period t. If the protocol ends successfully, GM generates user's secret key $usk_t[i]$ and certificate for an identify *id* and revocation token $token_{i,t}$, and sends $usk_t[i]$ and group membership certificate $cert = (cert_{index}, cert_{token}, id, token_{i,t})$ to the group user. Finally, GM updates registration table *reg*.

GRevoke(gpk,$\mathbf{RL_t}$,$\mathbf{TL_t}$,t) $\rightarrow$ (($\mathbf{RL_t})_{\mathbf{new}}$,($\mathbf{TL_t})_{\mathbf{new}}$). GM is in charge of executing this algorithm. GM adds the revocation $token_{i,t}$ and $token_{sk}$ to the revocation list RL_t and token list TL_t, respectively, then it updates revocation list and token list to $(RL_t)_{new}$ and $(TL_t)_{new}$ at time period t.

KeyUpdate(gpk,$\mathbf{usk_t[i]}$,$\mathbf{token_{i,t}}$,$\mathbf{RL_t}$,$\mathbf{TL_t}$,t+1) $\rightarrow$ ($\mathbf{usk_{t+1}[i]}, \mathbf{token_{i,t+1}}, \mathbf{RL_{t+1}}, \mathbf{TL_{t+1}}$). The algorithm is operated by user and GM to update $usk_t[i]$, $token_{i,t}$, RL_t and TL_t from time period t to $t+1$.

GSign(gpk,$\mathbf{usk_t[i]}$,M,t) $\rightarrow \sum$. On input *gpk*, signing secret key $usk_t[i]$ and time period *t*, it returns $\sum$ as a signature of message *M*.

GVerify(gpk,$\mathbf{RL_t}$,M,$\sum$,t) $\rightarrow$ 0/1. This algorithm is run by verifier to determine whether the signer has been revoked and the signature's validity, and outputs 1 if the signature is legitimate and 0 otherwise.

GOpen(gpk,tsk,M,$\sum$) $\rightarrow$ id. TM is in charge of executing this algorithm. It outputs the signer's identity *id* if the signature $\sum$ is valid, and outputs $\perp$(false) otherwise.

Security requirements

The security of DFSGS scheme requires correctness, full anonymity and forward-secure traceability. Correctness includes verification correctness and opening correctness. Verification correctness demands the signature produced by a legal user should pass the verification algorithm while opening correctness requires that a valid signature should be traced by TM to reveal the real signer's identity. Full anonymity means that anyone who receives a signature cannot identify which signer generated the signature on challenged message, and adversary $\mathcal{A}$ could query the opening of any signatures. Forward-secure traceability requires that even if the adversary $\mathcal{A}$ could corrupt tracing manager's secret key as well as some of the users, $\mathcal{A}$ cannot generate a legal signature which could be traced back to a non-corrupted user, or a corrupted user but the signature had been produced before this user was corrupted.

- *AddU*(i): Add an honest user i into the list *HUL* whose signing secret keys are generated honestly.
- *CrptU*(i): Choose a new user i corrupted by adversary $\mathcal{A}$ and add i to the list *CUL* storing corrupted users.
- $SenToGM(i, M_{in})$: This interactive protocol is carried out collaboratively by the corrupted user i and the legitimate GM.
- $SenToUser(i, M_{in})$: This interactive protocol is carried out collaboratively by the corrupted GM and the legitimate user i.
- *AlterReg*(i, *val*): Alter the registration table information reg_i to a specific value *val* chosen by $\mathcal{A}$.
- *ReadReg*(i): $\mathcal{A}$ obtains the registration information reg_i of group.
- *RevealU*(i, t): Sent the user's secret key to $\mathcal{A}$ and add i to *BUL* which means that its signing secret key is sent to $\mathcal{A}$ at time t.
- *RevealRT*(i, t): Sent the user's revocation token to $\mathcal{A}$ and add the user i to *UTL* at time t which means that its revocation token is sent to $\mathcal{A}$.
- *Sign*(i, M, t): $\mathcal{A}$ receives a signature $\sum$ on message M from an honest user i at time t and adds it to list *SL*.
- $Chal_b(i_0, i_1, M, t)$: Return a signature for given message M signed by user i_b for $b \in \{0, 1\}$ at time t and adds the challenge signature to list *CL*. Moreover, both of i_0, i_1 are honest and active at time t. Specially, this oracle could only be invoked once.
- *Revoke*(i, t): $\mathcal{A}$ could specify any user and request to be revoked. Then, add revoked user's revocation $token_{i,t}$ to list *RL* at time t.
- *UpdateK*(*info*, t): Update *info* including all of the secret keys of user and revocation tokens from time $t-1$ to t.
- $Open(M, \sum, t)$: $\mathcal{A}$ can obtain the user's identity who generates a signature $\sum$ at time t, and the signature $\sum \notin CL$.
- *IsActive*(i, *reg*, t): This algorithm checks whether the user i is registered in the group at time t and returns either 0 (indicating illegal user) or 1 (indicating legal user).

Given security parameter λ and time period T, the following is our definition of security experiments.

Definition 3 (*Correctness*) A DFSGS scheme is correct if all PPT adversaries $\mathcal{A}$ have at most a negligible advantage winning the correctness game that $Adv^{Corr}_{DFSGS,\mathcal{A}} = Pr[Exp^{Corr}_{DFSGS,\mathcal{A}}(\lambda, T) = 1]$ in experiment $Exp^{Corr}_{DFSGS,\mathcal{A}}(\lambda, T)$.

Correctness: $Exp^{Corr}_{DFSGS,\mathcal{A}}(\lambda, T)$

$(gpk, msk, tsk, usk_t) \leftarrow GKgen(\lambda, T)$; $HUL \leftarrow \emptyset$.

$(i, M) \leftarrow \mathcal{A}^{AddU,ReadReg,Revoke}(gpk)$.

$usk_t[i] \leftarrow KeyUpdate(gpk, usk_{t-1}[i], token_{i,t}, RL_{t-1}, t)$.

If $i \notin HUL$ or $usk_t[i] = \perp$ or $IsActive(i, reg_i, t) = 0$, output 0.

$\sum \leftarrow GSign(gpk, usk_t[i], M, t)$.

Output 1 if $GVerify(gpk, RL_t, M, \sum, t) = 0$ or $GOpen(gpk, tsk, M, \sum) = j$ and $j \neq i$,

else output 0.

Definition 4 (*Full Anonymity*) A DFSGS scheme is anonymous if all PPT adversaries $\mathcal{A}$ have at most a negligible advantage winning the anonymity game that $Adv^{Anon}_{DFSGS,\mathcal{A}} = Pr[Exp^{Anon-1}_{DFSGS,\mathcal{A}}(\lambda, T) = 1] - Pr[Exp^{Anon-0}_{DFSGS,\mathcal{A}}(\lambda, T) = 1] = 1$ in experiment $Exp^{Anon-b}_{DFSGS,\mathcal{A}}(\lambda, T)$.

Full Anonymity: $Exp^{Anon-b}_{DFSGS,\mathcal{A}}(\lambda, T)$

$(gpk, msk, tsk, usk_t) \leftarrow GKgen(\lambda, T)$; $HUL, CUL, BUL, CL, SL, UTL \leftarrow \emptyset$.

$b^* \leftarrow \mathcal{A}^{AddU,CrptU,RevealU,SenToUser,Open,AlterReg,UpdateK,Chal_b}(gpk)$.

Output b^*.

Definition 5 (*Forward-secure Traceability*) A DFSGS scheme is forward-secure traceable if all PPT adversaries $\mathcal{A}$ have at most a negligible advantage winning the forward-secure traceability game that $Adv^{FS-Trace}_{DFSGS,\mathcal{A}} = Pr[Exp^{FS-Trace}_{DFSGS,\mathcal{A}}(\lambda, T) = 1]$.

Forward-secure Traceability: $Exp^{FS-Trace}_{DFSGS,\mathcal{A}}(\lambda, T)$

$(gpk, msk, tsk, usk_t) \leftarrow GKgen(\lambda, T)$; $HUL, CUL, BUL, SL, UTL \leftarrow \emptyset$.

$(t^*, M^*, \sum^*) \leftarrow \mathcal{A}^{AddU,CrptU,RevealU,SenToGM,Open,AlterReg,UpdateK,Revoke}(gpk, tsk)$.

If $GVerify(gpk, RL_{t^*}, M^*, \sum^*, t^*) = 0$ or $\sum^* \in SL$, then output 0.

$i^* \leftarrow GOpen(gpk, tsk, M^*, \sum^*)$.

If $i^* \notin BUL$ or $i^* \in BUL$ and $\mathcal{A}$ only queried $usk_t[i^*]$ for $t > t^*$, then output 1, else output 0.

Our dynamic forward-secure group signature from lattice

The core construction of our DFSGS scheme is to implement base extension by combining bonsai tree signature and node select algorithm to achieve key update to satisfy forward security. The revocation function is implemented by generating a revocation token and then using the verifier-local revocation method. Finally, ZKAoK is constructed according to the relevant conditions.

Let N an anticipated number of prospective users, e.g. $N = 2^L$ for $L \in \mathbb{Z}^+$, and T maximum time periods, e.g. $T = 2^d$ for $d \in \mathbb{Z}^+$. Our DFSGS scheme consists of several algorithms as shown below.

GSetup(λ). Take security parameter λ as input, select $n = \mathcal{O}(\lambda)$, $q = poly(n)$, $k = \lceil \log q \rceil$, $m = 2nk$ and $m' = 2(n + L)k$. Let $p = poly(\lambda)$, $\kappa = \lambda / \log p$ and collision resistant hash function $H_1 : \{0,1\}^* \rightarrow [-p, p]^\kappa$. Define $\mathbf{G} = \mathbf{I}_n \bigotimes (1\ 2\ ...\ 2^{k-1})$. For every vector $\mathbf{b} \in \mathbb{Z}_q^n$, it can be expressed as $\mathbf{b} = \mathbf{G} \cdot bin(\mathbf{b})$, where $bin(\mathbf{b}) \in \{0,1\}^{nk}$. Select Gaussian parameters $s_i = \mathcal{O}(\sqrt{nk \log q})^{i+1} \cdot \omega(\sqrt{\log n})^{i+1}$ to generate bases or sample vectors for $i \in \{0, 1, \ldots, d\}$ and integer bound $\beta = \lceil s_{d+1} \cdot \log n \rceil$. Finally, output public parameters

$$pp = \{\lambda, n, m, m', q, L, d, s_0, \ldots, s_d\}.$$

GKgen$_{\text{GM,TM}}$(pp). The algorithm initializes the keys of group manager GM and tracing manager TM. Subsequently, GM outputs the group public key.

$GKgen_{GM}(pp)$
:

1. Run algorithm *TrapGen* to get $\mathbf{A}_0 \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{S}_0 \in \mathbb{Z}_q^{m \times m}$. Define $mpk = \mathbf{A}_0$ and $msk = \mathbf{S}_0$.
2. Sample $\mathbf{e} \xleftarrow{\$} \mathbb{Z}_q^n$, $\mathbf{A}_1, \mathbf{A}_2, \mathbf{B} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$ and $\mathbf{A}_j^b \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$ for $b \in \{0,1\}$ and $j \in [d]$.

$GKgen_{TM}(pp)$:

1. Sample $\mathbf{D}_0 \xleftarrow{\$} \mathbb{Z}_q^{n \times m'}$ and for each $i \in \{1, 2\}$, sample $\mathbf{S}_i \xleftarrow{\$} \{0,1\}^{L \times n}$, $\mathbf{E}_i \xleftarrow{\$} \{0,1\}^{L \times m'}$.
2. Compute $\mathbf{D}_1 = \mathbf{S}_1 \cdot \mathbf{D}_0 + \mathbf{E}_1$, $\mathbf{D}_2 = \mathbf{S}_2 \cdot \mathbf{D}_0 + \mathbf{E}_2$, set $tsk = \mathbf{S}_1$, $tpk = (\mathbf{D}_0, \mathbf{D}_1, \mathbf{D}_2)$ and send tpk to GM.

Then, GM initializes the counter of registered users to be $c = 0$, the revocation list $RL = \emptyset$, the token list $TL = \emptyset$ and outputs

$$gpk = (pp, \mathbf{A}_0, \mathbf{A}_1, \mathbf{A}_2, \mathbf{A}_1^0, \mathbf{A}_1^1, ..., \mathbf{A}_d^0, \mathbf{A}_d^1, \mathbf{B}, \mathbf{D}_0, \mathbf{D}_1, \mathbf{D}_2, \mathbf{e}).$$

$\langle$ GUJoin($\mathbf{token_{sk}}$,t),GIssue(gpk,msk,t) $\rangle$. The user who has a personal key pair (*pusk*, *pupk*) registered in the PKI interact with GM as follows.

1. The user samples $token_{sk} = \mathbf{x}_i \in \mathbb{Z}_q^{(d+2)m}$. Then user generates $sig_i = Sig_{pusk}(\mathbf{x}_i)$ by normal digital signature, and send $(sig_i, \mathbf{x}_i)$ to GM.
2. GM receives the user's requisition to join the group and checks the legitimacy of sig_i with *pupk*. The value c is the counter of current users in the group, which is initially 0 and increases by 1 when a new user joins the group. If the signature is valid, GM sets $id := c \in [0, 2^L - 1]$ as the member identifier for the user. Then, GM generates the certificate for the index $cert_{index} = Sign(msk, id)$ and increases the counter

$c := c + 1$. The node set should be determined as follows. For $z \in Nodes_{(0\to T-1)}$, we set $usk_0[i][z] = \perp$ if $z = \perp$. Else, we use d_z represent the length of z with $d_z \leq d$, and define the matrix

$$\mathbf{A}_{i,z} = [\mathbf{A}_0|\mathbf{A}_1 + id\mathbf{A}_2|\mathbf{A}_1^{z[1]}|...|\mathbf{A}_{d_z}^{z[d_z]}] \in \mathbb{Z}_q^{n\times(d_z+2)m}.$$

If $d_z = d$, it generates a vector $\mathbf{v}_{i,z} \in \mathbb{Z}^{(d+2)m}$ by

$$\mathbf{v}_{i,z} \leftarrow SampleD(ExtBasis(\mathbf{S}_0, \mathbf{A}_{i,z}), e, s_d),$$

and set $usk_0[i][z] = \mathbf{v}_{i,z}$. If $1 \leq d_z < d$, it generates a matrix $\mathbf{S}_{i,z} \in \mathbb{Z}^{(d_z+2)m\times(d_z+2)m}$ by

$$\mathbf{S}_{i,z} \leftarrow RandBasis(ExtBasis(\mathbf{S}_0, \mathbf{A}_{i,z}), s_{d_z}),$$

and set $usk_0[i][z] = \mathbf{S}_{i,z}$. The user secret key is $usk_0[i] = \{usk_0[i][z], z \in Nodes_{(0\to T-1)}, id, \mathbf{x}_i\}$. Then, run *KeyUpdate* algorithm to get $usk_t[i] = \{usk_t[i][z], z \in Nodes_{(t\to T-1)}, id, \mathbf{x}_i\}$, and revocation token is $token_{i,t} = \mathbf{A}_{i,t} \cdot \mathbf{x}_i$ where $\mathbf{A}_{i,t}$ represents $\mathbf{A}_{i,z}$ when $d_z = d$ at period t. GM generates the certificate for the token $cert_{token} = Sign(msk, token_{i,t})$. Finally, GM sends the $cert_i = (cert_{index}, cert_{token}, id, token_{i,t})$ to the user and updates the registration table *reg*.

GRevoke(gpk,$\mathbf{RL_t}$,$\mathbf{TL_t}$,t). GM publishes *RL* at the beginning of each period, marking users who have been revoked from this period. To revoke a group member $User_i$, GM adds $token_{i,t}$ directly into the revocation list RL_t at period t where $t \in [0, T-1]$ and updates $RL_t = RL_t \cup token_{i,t}$ and $TL_t = TL_t \cup \mathbf{x}_i$. Then GM publishes the revocation information RL_t.

KeyUpdate(gpk,$\mathbf{usk_t[i]}$,$\mathbf{token_{i,t}}$,$\mathbf{RL_t}$,$\mathbf{TL_t}$,t+1). Parse the set $usk_t[i] = \{usk_t[i][z], z \in Nodes_{(t\to T-1)}, id, \mathbf{x}_i\}$ and determine the set $Nodes_{(t+1\to T-1)}$. If $z' \in Nodes_{(t+1\to T-1)} = \perp$, set $usk_{t+1}[i][z'] = \perp$. Else, there is a $z \in Nodes_{(t\to T-1)}$ as a prefix of $z' = z||h$. The following are two scenarios.

1. If $z' = z$ (i.e. h is empty), then $usk_{t+1}[i][z'] = usk_t[i][z]$.
2. If $z' = z||h$ (i.e. h is non-empty), it can delegate the basis in following two subcases. If $d_{z'} = d$, then run

$$\mathbf{v}_{i,z'} \leftarrow SampleD(ExtBasis(\mathbf{S}_{i,z}, \mathbf{A}_{i,z'}), \mathbf{e}, s_d),$$

and set $usk_{t+1}[i][z'] = \mathbf{v}_{i,z'}$. If $d_{z'} < d$, it computes a matrix $\mathbf{S}_{i,z'}$ by

$$\mathbf{S}_{i,z'} \leftarrow RandBasis(ExtBasis(\mathbf{S}_{i,z}, \mathbf{A}_{i,z'}), s_{d_z'}),$$

and set $usk_{t+1}[i][z'] = \mathbf{S}_{i,z'}$.

The updated key is $usk_{t+1}[i] = \{usk_{t+1}[i][z'], z' \in Nodes_{(t+1\to T-1)}, id, \mathbf{x}_i\}$. At the same time, the user needs update $token_{i,t}$ to $token_{i,t+1}$ and GM also needs update RL_t to RL_{t+1}. For $\mathbf{x}_i \in TL_t$, GM computes $token_{i,t+1} = \mathbf{A}_{i,z'} \cdot \mathbf{x}_i$ where $d_{z'} = d$, and publishes RL_{t+1}.

GSign(gpk,$\mathbf{usk_t[i]}$,M,t). Based on the node set $Nodes_{(t\to T-1)}$, we can find a z satisfying $z = bin(t)$ and $usk_t[i][z] = \mathbf{v}_{i,z}$. The user signs a message M using $usk_t[i] = \{\mathbf{v}_{i,z}, bin(id), \mathbf{x}_i\}$ as follows.

1. For each $j \in \{1,2\}$, sample $\mathbf{r}_j \xleftarrow{\$} \{0,1\}^{m'}$, $\mathbf{e}_0 \xleftarrow{\$} \chi^m$, $\mathbf{e}_{j,1} \xleftarrow{\$} \{0,1\}^n$, $\mathbf{e}_{j,2} \xleftarrow{\$} \{0,1\}^L$ and compute

$$\begin{cases} \mathbf{A}_{i,z} \cdot \mathbf{v}_{i,z} = \mathbf{e} \mod q \\ \mathbf{w} = \mathbf{B}^\top \cdot token_{i,t} + \mathbf{e}_0 \mod q \\ \mathbf{c}_{1,1} = \mathbf{D}_0 \cdot \mathbf{r}_1 + \mathbf{e}_{1,1} \\ \mathbf{c}_{1,2} = \mathbf{D}_1 \cdot \mathbf{r}_1 + \mathbf{e}_{1,2} + \lfloor q/2 \rceil \cdot bin(id) \\ \mathbf{c}_{2,1} = \mathbf{D}_0 \cdot \mathbf{r}_2 + \mathbf{e}_{2,1} \\ \mathbf{c}_{2,2} = \mathbf{D}_2 \cdot \mathbf{r}_2 + \mathbf{e}_{2,2} + \lfloor q/2 \rceil \cdot bin(id) \end{cases}$$

2. Generate a NIZKAoK $\prod_{gs}$ to demonstrate the possession of tuple

$$\xi = (\mathbf{v}_{i,z}, \mathbf{x}_i, \mathbf{r}_1, \mathbf{r}_2, \mathbf{e}_{1,1}, \mathbf{e}_{1,2}, \mathbf{e}_{2,1}, \mathbf{e}_{2,2}, id)$$

satisfying that: 1) $(\mathbf{c}_{1,1}, \mathbf{c}_{1,2})$ and $(\mathbf{c}_{2,1}, \mathbf{c}_{2,2})$ are two legitimate ciphertexts of identity id; 2) $\mathbf{A}_{i,z} \cdot \mathbf{v}_{i,z} = \mathbf{e} \mod q$ and $||\mathbf{v}_{i,z}||_\infty \leq \beta$; and 3) $\mathbf{w} = \mathbf{B}^\top \cdot \mathbf{A}_{i,t} \cdot \mathbf{x}_i + \mathbf{e}_0 \mod q$. The ZKAoK for the encryption scheme to handle statement 1) was mentioned in section "The basic protocol". The matrix $\mathbf{A}_{i,z}$ can be represented as $[\mathbf{A}_{id}|\mathbf{A}_z]$, so the form $\mathbf{A}_{i,z} \cdot \mathbf{v}_{i,z} = \mathbf{e}$ can be expressed as $\mathbf{A}_{id} \cdot \mathbf{v}_1 + \mathbf{A}_z \cdot \mathbf{v}_2 = \mathbf{e}$. To protect the anonymity of the user, the matrix $\mathbf{A}_{id}$ cannot be disclosed. Therefore, we convert $\mathbf{A}_{id} = [\mathbf{A}_0|\mathbf{A}_1 + id\mathbf{A}_2]$ to $\mathbf{A}' = [\mathbf{A}_0|\mathbf{A}_1|\mathbf{A}_2|2\mathbf{A}_2|...|2^{L-1}\mathbf{A}_2] = [\mathbf{A}_0|\mathbf{A}_1|\mathbf{g}_L \bigotimes \mathbf{A}_2]$ and $\mathbf{A}_{id} \cdot \mathbf{v}_1$ can be expressed as $[\mathbf{A}_0|\mathbf{A}_1|\mathbf{g}_L \bigotimes \mathbf{A}_2] \cdot (\mathbf{v}_{1,1}, \mathbf{v}_{1,2}, bin(id) \bigotimes \mathbf{v}_{1,2})$. The statements 2) and 3) were covered by ZKAok for committed values mentioned in section "Our zero-knowledge AoK of committed values". Then, we convert all of the conditions into the equation $\hat{\mathbf{A}} \cdot \hat{\mathbf{x}} = \hat{\mathbf{y}}$ where $\hat{\mathbf{A}}, \hat{\mathbf{y}}$ are public and $\hat{\mathbf{x}}$ is secret. Finally, the protocol is repeated $\kappa = \lambda/\log p$ times to ensure negligible soundness error and make it non-interactive via Fiat-Shamir transform as a triple $\prod_{gs} = ((\alpha_i)_{i=1}^\kappa, (RSP_i)_{i=1}^\kappa)$ where $(\alpha_i)_{i=1}^\kappa = H_1(M, \{C_{aux_i}\}_{i=1}^\kappa, \hat{\mathbf{A}}, \hat{\mathbf{y}}, \mathcal{M}, \mathbf{w}, \mathbf{c}_{1,1}, \mathbf{c}_{1,2}, \mathbf{c}_{2,1}, \mathbf{c}_{2,2}, t) \in [-p, p]$ and C_{aux_i} is computed according to the commitment scheme aCommit from Yang et al. (2019). Finally, output the signature $\sum = (\prod_{gs}, \mathbf{w}, \mathbf{c}_{1,1}, \mathbf{c}_{1,2}, \mathbf{c}_{2,1}, \mathbf{c}_{2,2})$.

GVerify(gpk,$\mathbf{RL_t}$,M,$\sum$,t). The following is how the algorithm works.

1. Run the verification phase of the NIZKAoK to check the proof $\prod_{gs}$. If any of the conditions fails, output 0.
2. Check the revocation list RL_t. For $token_{j,t} \in RL_t$, compute $\mathbf{w}' = \mathbf{w} - \mathbf{B}^\top \cdot token_{j,t}$ which is $\mathbf{w}' = \mathbf{B}^\top (token_{i,t} - token_{j,t}) + \mathbf{e}_0$. If $token_{i,t} = token_{j,t}$ and $||\mathbf{w}'||_\infty \le \beta$, it proves that the signer has been revoked. In this case, output 0.
3. Output 1.

GOpen(gpk,tsk,M,$\sum$). Check the signature's legitimacy before proceeding with the steps below.

1. Use $\mathbf{S_1}$ to decrypt id by computing $b = \lfloor \frac{c_{1,2} - S_1 \cdot c_{1,1}}{q/2} \rceil$.
2. Compute the identity $id = \sum_{l=1}^{L} 2^{l-1} \cdot b[l]$ and output the identity.

Security analysis

Correctness

For each $token_{j,t} \in RL_t$, compute $\mathbf{w}' = \mathbf{w} - \mathbf{B}^\top token_{j,t} = \mathbf{B}^\top (token_{i,t} - token_{j,t}) + \mathbf{e}_0$. If there is a revocation token such as $token_{i,t} = token_{j,t}$ and $||\mathbf{w}'||_\infty \le \beta$, it means that the verification fails, and the signature is rejected. Due to the completeness of the argument of knowledge, the valid signature $\sum$ is always accepted by algorithm *GVerify*. As for opening correctness, the algorithm *GOpen* computes $b = \mathbf{c}_{1,2} - \mathbf{S}_1 \cdot \mathbf{c}_{1,1} = \mathbf{E}_1 \cdot \mathbf{r}_1 + \mathbf{e}_{1,2} + \lfloor q/2 \rceil \cdot bin(id) - \mathbf{S}_1 \cdot \mathbf{e}_1$. For $l \in [1, L]$, it sets $b[l] = 1$ if $b[l]$ is closer to $\lfloor q/2 \rceil$ than to 0 and $b[l] = 0$ otherwise. Finally, it converts binary b to an integer *id*.

Full anonymity

Theorem 1 *In the RO model, our DFSGS scheme is fully anonymous under the LWE assumption.*

Proof Define the challenger and adversary role of $\mathcal{C}$ and $\mathcal{A}$, respectively. A sequence of indistinguishable games will be used to prove this theorem. In game i, let W_i denote the adversary's output.

Game 0 : We define the experiment $Exp^{Anon-0}_{DFSGS,\mathcal{A}}(\lambda, T)$ as original game. Challenger $\mathcal{C}$ obtains the group public key, member certificate, existing group user's secret key, tracing public key according to the scheme and sends them to adversary $\mathcal{A}$. $\mathcal{C}$ initializes the revocation list $RL = \varnothing$, registration query list *RU*, corruption user list *CL* and revocation token query list *UTL*. In the query stage, $\mathcal{A}$ can query for the signature of any message of any user, open the query of the signature on the corresponding message and update *RU*, *CL*, *UTL*. In the challenge phase, $\mathcal{A}$ sends message M^* along with two users $i_0, i_1 \in [N]$, $i_0, i_1 \notin CL \cup UTL$ and $token_{i_0}, token_{i_1} \notin RL$. $\mathcal{C}$ sends back a signature $\sum^* = (\prod^*_{gs}, \mathbf{w}^*, \mathbf{c}^*_{1,1}, \mathbf{c}^*_{1,2}, \mathbf{c}^*_{2,1}, \mathbf{c}^*_{2,2}) \leftarrow GSign$ $(gpk, usk_{t^*}[i_0], M^*, t^*)$. $\mathcal{A}$ can still perform signature query, secret key query, opening query and revocation token query about $i_b \neq \{i_0, i_1\}$. In the end, $\mathcal{A}$ returns $b^* = 1$ for the conjecture of i_b. We have $Pr[W_0 = 1] = Pr[Exp^{Anon-0}_{DFSGS\mathcal{A}}(\lambda, T) = 1]$.

Game 1 : This game is completely consistent with Game 0 aside from adding $\mathbf{S}_2$ to *tsk* instead of erasing it. This change makes no difference to Game 0 in $\mathcal{A}$'s view. So $Pr[W_1 = 1] = Pr[W_0 = 1]$.

Game 2 : This game is completely consistent with Game 1 aside from the open oracle opens signatures using the $\mathbf{S}_2$ instead of using real *tsk* $\mathbf{S}_1$. It is clear that $\mathcal{A}$'s perspective will remain unchanged from Game 1 until incident F_1 happens that $\mathcal{A}$ queries the opening of a signature $\sum = (\prod_{gs}, \mathbf{w}, \mathbf{c}_{1,1}, \mathbf{c}_{1,2}, \mathbf{c}_{2,1}, \mathbf{c}_{2,2})$ which encrypts distinct bit strings. Since F_1 breaks the soundness of the argument system $\prod_{gs}$, $|Pr[W_2 = 1] - Pr[W_1 = 1]| \le Pr[F_1] \le Adv^{sound}_{\prod_{gs}} = negl(\lambda)$.

Game 3 : This game follows Game 2 aside from $\mathcal{C}$ replaces the legitimate proof with a simulated proof without using the witness. Game 3 and Game 2 are statistically indistinguishable from each other from the perspective of $\mathcal{A}$ since the argument system is statistically zero-knowledge. For this reason, $Pr[W_3 = 1] \approx Pr[W_2 = 1]$.

Game 4 : In this game, we compute $(\mathbf{c}^*_{1,1}, \mathbf{c}^*_{1,2})$ by encrypting the binary representation of i_1 while $(\mathbf{c}^*_{2,1}, \mathbf{c}^*_{2,2})$ still encrypt i_0. The semantic security of our encryption scheme with respect to $(\mathbf{D}_0, \mathbf{D}_1)$ (which is implied by the LWE assumption) ensures that $|Pr[W_4 = 1] - Pr[W_3 = 1]| = negl(\lambda)$.

Game 5 : This game is completely consistent with Game 4 with one modification that we switch back to use $\mathbf{S}_1$ for open oracle. Obviously, the view of $\mathcal{A}$'s will remain unchanged unless incident F_2 that $\mathcal{A}$ queries the open oracle for $\sum = (\prod_{gs}, \mathbf{w}, \mathbf{c}_{1,1}, \mathbf{c}_{1,2}, \mathbf{c}_{2,1}, \mathbf{c}_{2,2})$

which encrypts distinct strings. By reason of F_2 violates the simulation soundness of the protocol, we have $|Pr[W_5 = 1] - Pr[W_4 = 1]| \leq Pr[F_2] \leq Adv^{sim}_{\prod_{gs}} = negl(\lambda)$.

Game 6 : This game is completely consistent with game 5 aside from we change $(\mathbf{c}^*_{2,1}, \mathbf{c}^*_{2,2})$ by encrypting the binary representation of i_1. By the semantic security of the encryption scheme for $(\mathbf{D}_0, \mathbf{D}_2)$, $\mathcal{A}$ is unaffected by this change. Since we are now using $\mathbf{S}_1$ for open oracle, changing $(\mathbf{c}^*_{2,1}, \mathbf{c}^*_{2,2})$ has no effect on $\mathcal{A}$'s view. Hence, $|Pr[W_6 = 1] - Pr[W_5 = 1]| = negl(\lambda)$.

Game 7 : This game is totally coequal with Game 6 aside from the $\mathcal{C}$ replaces the initial revocation token. We have $\mathbf{w} = \mathbf{B}^\top \cdot token_{i,t} + \mathbf{e}_0 \mod q$. $\mathcal{C}$ samples $\mathbf{v} \xleftarrow{\$} \mathbb{Z}_q^n$ uniformly and compute $\mathbf{w} = \mathbf{B}^\top \cdot \mathbf{v} + \mathbf{e}_0 \mod q$. Thus, $Pr[W_7 = 1] \approx Pr[W_6 = 1]$.

Game 8 : In this game, the challenge $\mathcal{C}$ samples $\mathbf{w}$ uniformly. Since the pair $(\mathbf{B}, \mathbf{w})$ is an $\mathrm{LWE}_{n,m,q,\chi}$ instance, $\mathcal{C}$ replaces $\mathbf{w}$ with uniformly sampled $\mathbf{w}' \xleftarrow{\$} \mathbb{Z}_q^m$, $Pr[W_8 = 1] \approx Pr[W_7 = 1]$.

Game 9 : This game is totally coequal with Game 8 aside from that replace the simulated proof with a real proof $\prod_{gs*}$ using the witnesses i.e. replace the simulated transcript by a real transcript. Due to the statistical zero-knowledge property of argument system $\prod_{gs*}$, the two transcripts are indistinguishable. Thus, we have $Pr[W_9 = 1] \approx Pr[W_8 = 1]$. In this instance, the view of the $\mathcal{A}$ is the same as the experiment $Exp^{Anon-1}_{DFSGS,\mathcal{A}}(\lambda)$. So $Pr[W_9 = 1] = Pr[Exp^{Anon-1}_{DFSGS,\mathcal{A}}(\lambda) = 1]$.

Finally, we have $|Pr[Exp^{Anon-1}_{DFSGS,\mathcal{A}}(\lambda) = 1] - Pr[Exp^{Anon-0}_{DFSGS,\mathcal{A}}(\lambda) = 1]| = negl(\lambda)$. Thereby, our scheme is proved anonymous by these games.

Forward-secure traceability

Theorem 2 *In the RO model, our DFSGS scheme is forward-secure traceable under the SIS assumption.*

Proof

Assume that adversary $\mathcal{A}$ could break the forward-secure traceability of our scheme with non-negligible probability, there is a adversary $\mathcal{B}$ can solve the $\mathrm{SIS}^{\infty}_{n,\bar{m},q,2\beta}$ problem with non-negligible probability as well. □

Given a matrix $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n\times\bar{m}}$, $\mathcal{B}$ is required to discover a non-zero vector $\bar{\mathbf{v}} \in \mathbb{Z}_q^{\bar{m}}$ satisfying that $\bar{\mathbf{A}} \cdot \bar{\mathbf{v}} = 0 \mod q$ and $||\bar{\mathbf{v}}||_\infty \leq 2\beta$. Simulating the view of the adversary $\mathcal{A}$ attacking the forward-secure traceability, $\mathcal{B}$ constructs an algorithm that outputs a valid $\bar{\mathbf{v}}$ satisfying $\bar{\mathbf{A}} \cdot \bar{\mathbf{v}} = 0 \mod q$ and $||\bar{\mathbf{v}}||_\infty \leq 2\beta$.

Setup: Define matrix $\bar{\mathbf{A}} = [\bar{\mathbf{A}}_0|\bar{\mathbf{A}}_i|\bar{\mathbf{A}}_1|...|\bar{\mathbf{A}}_d]$ for $\bar{\mathbf{A}}_j \in \mathbb{Z}_q^{n\times m}$, $j \in \{0, i, 1, 2, ..., d\}$. Set $t = 0$ and BUL to be empty, sample $\bar{\mathbf{z}} = (\mathbf{z}_0||\mathbf{z}_i||\mathbf{z}_1||...||\mathbf{z}_d) \in \mathbb{Z}^{\bar{m}}$ where $\mathbf{z}_j$ is sampled from $D_{\mathbb{Z}^m, s_{d+1}}$ and $||\bar{\mathbf{z}}||_\infty \leq \beta$, and compute $\mathbf{e} = \bar{\mathbf{A}} \cdot \bar{\mathbf{z}} \mod q$. Let $i^* \in [0, N-1]$ be the targeted user, $t^* \in [0, T-1]$ be the targeted forgery time and $z^* = bin(t^*)$. Define $\mathbf{A}_0$ to be $\bar{\mathbf{A}}_0$, $\mathbf{A}_1 + id\mathbf{A}_2$ to be $\bar{\mathbf{A}}_i$ and $\mathbf{A}_b^{z^*[b]}$ to be $\bar{\mathbf{A}}_b$ for $b \in [d]$. Generate $\mathbf{A}_b^{1-z^*[b]}$ via the algorithm $(\mathbf{A}_b^{1-z^*[b]}, \mathbf{S}_b) \leftarrow TrapGen(n, m, q)$ and the tracing manager key pair $(\mathbf{B}, \mathbf{S}) \leftarrow TrapGen(n, m, q)$. Finally, send group public key and tracking key to the adversary $\mathcal{A}$.

Join: For $i \in [N]$, if $i \neq i^*$, choose $\mathbf{x}_i \xleftarrow{\$} \mathbb{Z}_q^{(d+2)m}$ randomly and send $\mathbf{x}_i$, $sig_i = Sig(\mathbf{x}_i)$ to $\mathcal{B}$. Then $\mathcal{B}$ computes $\mathbf{A}_{i,t} = [\mathbf{A}_0|\mathbf{A}_1 + id\mathbf{A}_2|\mathbf{A}_1^{z[1]}|...|\mathbf{A}_d^{z[d]}]$ and revocation $token_{i,t} = \mathbf{A}_{i,t} \cdot \mathbf{x}_i$. Finally, $\mathcal{B}$ sends $cert_i = (cert_{index}, cert_{token}, id, token_{i,t})$ to user.

Queries: When $\mathcal{A}$ asks the random oracle H_1, $\mathcal{B}$ responds to a uniformly random string and records the information inquired. At time period t, $\mathcal{B}$ interact with $\mathcal{A}$ and replies with $\mathcal{A}$'s queries as follows.

- Secret key Queries: When the queried user's identity $i = i^*$, if $i^* \in BUL$ or $t \leq t^*$, $\mathcal{B}$ aborts. Besides, for $z \in Nodes_{(t\to T-1)}$, $\mathcal{B}$ generates $usk[i][z]$ via $SampleD(ExtBasis(\mathbf{S}_{i^*,z'}, \mathbf{A}_{i^*,z}), \mathbf{e}, s_d)$ and $RandBasis(ExtBasis(\mathbf{S}_{i^*,z'}, \mathbf{A}_{i^*,z}), s_{d_z})$. Next, $\mathcal{B}$ sends $usk_t[i]$ to $\mathcal{A}$ and adds i^* to BUL. When $\mathcal{A}$ queried user's identity $i \neq i^*$, if $i \in BUL$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ uses the same method to compute $usk_t[i]$. Finally, $\mathcal{B}$ sends $usk_t[i]$ to $\mathcal{A}$ and adds i to BUL.
- Signature Queries: $\mathcal{A}$ queries the random oracle for the signature of the message M. If $i \in BUL$ at time t, $\mathcal{B}$ aborts. Otherwise, if $i = i^*$, $\mathcal{B}$ utilizes simulated zero-knowledge proof $\prod^*$ with the help of oracle H_1 to generate a signature of M, and return the signature $\sum$ to $\mathcal{A}$. If $i \neq i^*$, $\mathcal{B}$ answers $\mathcal{A}$ with algorithm $GSign$.

Forgery: $\mathcal{A}$ forges a signature $\sum^*$ of message M^* at targeted time period t' satisfying that $GVerify = 1$ and making a signing query at M^* yields no result for $\prod^*$. If $t' \neq t^*$, $\mathcal{B}$ aborts. Assuming that $\mathcal{A}$ successfully forges the signature

Table 1 Comparison among lattice-based dynamic signature schemes

Schemes	*Gpk*	*Usk*	*GS*	Forward Secure	Dynamic
Libert et al. (2016)	$\tilde{\mathcal{O}}(\lambda^2 \cdot L)$	$\tilde{\mathcal{O}}(\lambda)$	$\tilde{\mathcal{O}}(\lambda \cdot L)$	No	Partially Dynamic
Ling et al. (2018)	$\tilde{\mathcal{O}}(\lambda^2 \cdot L)$	$\tilde{\mathcal{O}}(\lambda \cdot L)$	$\tilde{\mathcal{O}}(\lambda \cdot L)$	No	Partially Dynamic
Ling et al. (2019)	$\tilde{\mathcal{O}}(\lambda^2 + \lambda \cdot L)$	$\tilde{\mathcal{O}}(\lambda + L)$	$\tilde{\mathcal{O}}(\lambda \cdot L)$	No	Fully Dynamic
Kansal et al. (2020)	$\tilde{\mathcal{O}}(\lambda^2 \cdot L)$	$\tilde{\mathcal{O}}(\lambda)$	$\tilde{\mathcal{O}}(\lambda^3 \cdot L)$	Yes	Fully Dynamic
Ours	$\tilde{\mathcal{O}}(\lambda^2 \cdot d + \lambda \cdot L)$	$\tilde{\mathcal{O}}(\lambda \cdot d^3)$	$\tilde{\mathcal{O}}(\lambda^2 \cdot (L + d))$	Yes	Fully Dynamic

$\sum^* = ((\alpha_i^*)_{i=1}^{\kappa}, (RSP_i^*)_{i=1}^{\kappa}, \mathbf{w}, \mathbf{c}_{1,1}, \mathbf{c}_{1,2}, \mathbf{c}_{2,1}, \mathbf{c}_{2,2})$ with advantage ε. i' can be obtained by algorithm *GOpen*. If $i' = i^*$, $\mathcal{B}$ can use the forgery to handle the SIS problem in the following way.

For $(M^*, \{C_{aux}^*\}_{i=1}^{\kappa}, \mathbf{w}^*, \mathbf{c}_{1,1}^*, \mathbf{c}_{1,2}^*, \mathbf{c}_{2,1}^*, \mathbf{c}_{2,2}^*, t^*)$, $\mathcal{A}$ must have queried oracle H_1. Owing to the challenge space $CH : \{-p, \ldots, 0, \ldots, p\}$ and the quadratic constraint, the probability of guessing challenge value is $2/(2p+1)$. The probability of correctly guessing this value (i.e. $(2/(2p+1))^{\kappa}$) is negligible because of the choice of κ. Let Q_{H_1} be the upper limit of queries to oracle H_1 and $(M^*, \{C_{aux}^*\}_{i=1}^{\kappa}, \mathbf{w}^*, \mathbf{c}_{1,1}^*, \mathbf{c}_{1,2}^*, \mathbf{c}_{2,1}^*, \mathbf{c}_{2,2}^*, t^*)$ be the h-th oracle query. Let h represents the forking point that is being targeted. $\mathcal{B}$ replays $\mathcal{A}$ polynomial-number times. Among these queries, the first h-1 queries keep the input and the oracle H_1 unchanged, and the challenge values $\alpha_1, \alpha_2, ..., \alpha_{h-1}$ are the same. But starting from the h-th query, the challenge values $\alpha_h, \alpha_{h+1}, \ldots, \alpha_{Q_{H_1}}$ start to be different. The improved Forking Lemma (Brickell et al. 2000) guarantees that for $(M^*, \{C_{aux_i}^*\}_{i=1}^{\kappa}, \mathbf{w}^*, \mathbf{c}_{1,1}^*, \mathbf{c}_{1,2}^*, \mathbf{c}_{2,1}^*, \mathbf{c}_{2,2}^*, t^*)$, $\mathcal{B}$ can obtain $(\alpha_h^{-p}, \ldots, \alpha_h^{p}) \in \{-p, \ldots, p\}^{\kappa}$ with a probability greater than 1/2. There exits $(\alpha_{h,j}^{-p}, \ldots, \alpha_{h,j}^{p}) = \{-p, \ldots, p\}$ for some $j \in [\kappa]$ with a probability close to 1. From the corresponding response $(RSP_{h,j}^{-p}, \ldots, RSP_{h,j}^{p})$, $\mathcal{B}$ can extract the witness tuple $\xi^* = (\mathbf{v}_{i,z}, \mathbf{x}_i, \mathbf{r}_1, \mathbf{r}_2, \mathbf{e}_{1,1}, \mathbf{e}_{1,2}, \mathbf{e}_{2,1}, \mathbf{e}_{2,2}, id)$ such that $||\mathbf{v}_{i,z}||_\infty \le \beta, ||\mathbf{x}_i||_\infty \le \beta$ and

$$\begin{cases} \mathbf{A}_{i,z} \cdot \mathbf{v}_{i,z} = \mathbf{e} \mod q \\ \mathbf{w} = \mathbf{B}^\top \cdot token_{i,t} + \mathbf{e}_0 \mod q \\ \mathbf{c}_{1,1} = \mathbf{D}_0 \cdot \mathbf{r}_1 + \mathbf{e}_{1,1} \\ \mathbf{c}_{1,2} = \mathbf{D}_1 \cdot \mathbf{r}_1 + \mathbf{e}_{1,2} + \lfloor q/2 \rceil \cdot bin(id) \\ \mathbf{c}_{2,1} = \mathbf{D}_0 \cdot \mathbf{r}_2 + \mathbf{e}_{2,1} \\ \mathbf{c}_{2,2} = \mathbf{D}_2 \cdot \mathbf{r}_2 + \mathbf{e}_{2,2} + \lfloor q/2 \rceil \cdot bin(id). \end{cases}$$

When correctly guessing i^* and t^*, it means $id = id^*$ and $z = z^*$. In this case, we have $\bar{\mathbf{A}} \cdot \mathbf{v}_{i,z} = \bar{\mathbf{A}} \cdot \bar{\mathbf{z}} = \mathbf{e} \mod q$ where $\bar{\mathbf{A}} = \mathbf{A}_{i,z}$. Because $\mathcal{A}$ has never queried the secret key at all or user secret key at time before t^*, $\bar{\mathbf{z}}$ is unknown to $\mathcal{A}$. Moreover, from the perspective of $\mathcal{A}$, $\bar{\mathbf{z}}$ is from the distribution $D_{\mathbb{Z}^m, s_{d+1}}$. At this time, there is a high probability that $\mathbf{v}_{i,z} \neq \bar{\mathbf{z}}$. Let $\bar{\mathbf{v}} = \mathbf{v}_{i,z} - \bar{\mathbf{z}}$ and $||\bar{\mathbf{v}}||_\infty \le 2\beta$, so $\bar{\mathbf{v}}$ is a non-zero solution of $\bar{\mathbf{A}} \cdot \bar{\mathbf{v}} = 0 \mod q$. Due to the difficulty of the SIS problem, the advantage of $\mathcal{A}$'s successful forgery of a signature is negligible and hence the scheme is forward-secure traceable.

Efficiency analysis

In Table 1, we show the comparison between some related GS schemes based on lattice in term of the length of *Gpk*, the length of *Usk*, the length of signature *GS*, forward security and support of full dynamics. Among them, $N = 2^L$ is the quantity of group members, $T = 2^d$ is the max quantity of time periods. Define $t = \omega(\log \lambda)$ for other schemes and $t' = \lambda / \log p$ for our scheme which represent the number of interactions between the prover and verifier in zero knowledge where p is polynomial in λ.

From Table 1, our scheme has the forward-security while achieving full dynamics. Moreover, Compared with Kansal et al. (2020), the length of *Gpk* and *GS* has been improved, and the length of *Usk* has no connection with the quantity of group members. Since our scheme combines an efficient ZKAoK with soundness error 1/poly, for the same negligible soundness error, the number t' of repeating the protocol is much smaller than t. At the same time, the verification efficiency of the protocol will also be significantly improved.

Conclusion

In this paper, we constructed a lattice-based DFSGS scheme which is provably secure under the RO model. Compared with the existing schemes, our DFSGS scheme allows members to join and to be revoked at any time and achieves forward security, and the length of signature have been improved by constructing an efficient zero-knowledge proof. In our next work, we will explore to design a more efficient dynamic forward-secure group signature scheme based on lattice such as the length of group public key and signature have no connection with the quantity of group members and DFSGS scheme without NIZK in the standard model. Besides, it would bring a further improvement of signature size if we modify our scheme to work over ideal lattice or NTRU lattice and apply the zero-knowledge argument as Esgin et al. (2020), Lyubashevsky et al. (2020), Attema et al. (2020). In addition, we will consider whether there is another approach to design a fully dynamic forward-secure group signature from lattice in our further research.

Acknowledgements
Not applicable.

Authors' contributions
The first author constructed the scheme with careful security proofs and wrote the manuscript. The second author reviewed the manuscript and checked the validity of the scheme and the security proofs. He also proofread the manuscript and corrected the grammar mistakes. The third author joined the discussion of the work and designed the whole figures and tables of the manuscript. All authors read and approved the final manuscript.

Funding
This work is supported by the Major Program of Guangdong Basic and Applied Research (2019B030302008), National Natural Science Foundation of China (61872152), and Science and Technology Program of Guangzhou (201902010081).

Availability of data and materials
Not applicable.

Declarations

Competing interests
The authors declare that they have no competing interests.

Author details
[1]College of Mathematics and Informatics, South China Agricultural University, 483 Wushan Road, Guangzhou 510642, China. [2]Guangzhou Key Laboratory of Intelligent Agriculture, 483 Wushan Road, Guangzhou 510642, China.

Received: 18 January 2022 Accepted: 24 March 2022
Published online: 02 October 2022

References
Ajtai M (1996) Generating hard instances of lattice problems. In: Proceedings of the twenty-eighth annual ACM symposium on theory of computing, pp 99–108
Alwen J, Peikert C (2009) Generating shorter bases for hard random lattices. In: 26th International symposium on theoretical aspects of computer science STACS 2009. IBFI Schloss Dagstuhl, pp 75–86
Ateniese G, Camenisch J, Joye M, Tsudik G (2000) A practical and provably secure coalition-resistant group signature scheme. In: Annual international cryptology conference. Springer, pp 255–270
Attema T, Lyubashevsky V, Seiler G (2020) Practical product proofs for lattice commitments. In: Annual international cryptology conference. Springer, pp 470–499
Bellare M, Micciancio D, Warinschi B (2003) Foundations of group signatures: formal definitions, simplified requirements, and a construction based on general assumptions. In: International conference on the theory and applications of cryptographic techniques. Springer, Berlin, pp 614–629
Bellare M, Shi H, Zhang C(2005) Foundations of group signatures: the case of dynamic groups. In: Cryptographers' Track at the RSA conference. Springer, pp 136–153
Boneh D, Boyen X, Shacham H (2004) Short group signatures. In: Annual international cryptology conference. Springer, pp 41–55
Boneh D, Shacham H (2004) Group signatures with verifier-local revocation. In: Proceedings of the 11th ACM conference on computer and communications security, pp 168–177
Boyen X, Shacham H, Shen E, Waters B (2006) Forward-secure signatures with untrusted update. In: Proceedings of the 13th ACM conference on computer and communications security, pp 191–200
Boyen X, Waters B (2007) Full-domain subgroup hiding and constant-size group signatures. In: International workshop on public key cryptography. Springer, pp 1–15
Boyen X, Waters B(2006) Compact group signatures without random oracles. In: Annual international conference on the theory and applications of cryptographic techniques. Springer, pp 427–444
Brickell E, Pointcheval D, Vaudenay S, Yung M (2000) Design validations for discrete logarithm based signature schemes. In: International workshop on public key cryptography. Springer, pp 276–292
Camenisch J, Lysyanskaya A (2002) Dynamic accumulators and application to efficient revocation of anonymous credentials. In: Annual international cryptology conference. Springer, pp 61–76
Cash D, Hofheinz D, Kiltz E, Peikert C (2010) Bonsai trees, or how to delegate a lattice basis. In: Annual international conference on the theory and applications of cryptographic techniques. Springer, pp 523–552
Chaum D, Van Heyst E (1991) Group signatures. In: Workshop on the theory and application of of cryptographic techniques. Springer, Berlin, pp 257–265
Esgin MF, Nguyen NK, Seiler G (2020) Practical exact proofs from lattices: New techniques to exploit fully-splitting rings. In: International conference on the theory and application of cryptology and information security. Springer, pp 259–288
Gentry C, Peikert C, Vaikuntanathan V (2008) Trapdoors for hard lattices and new cryptographic constructions. In: Proceedings of the fortieth annual ACM symposium on theory of computing, pp 197–206
Gordon SD, Katz J, Vaikuntanathan V (2010) A group signature scheme from lattice assumptions. In: International conference on the theory and application of cryptology and information security. Springer, pp 395–412
Groth J (2007) Fully anonymous group signatures without random oracles. In: International conference on the theory and application of cryptology and information security. Springer, pp 164–180
Groth J(2006) Simulation-sound nizk proofs for a practical language and constant size group signatures. In: International conference on the theory and application of cryptology and information security. Springer, pp 444–459
Kansal M, Dutta R, Mukhopadhyay S (2020) Group signature from lattices preserving forward security in dynamic setting. Adv Math Commun 14(4):535
Kiayias A, Yung M (2006) Secure scalable group signature with dynamic joins and separable authorities. Int J Secur Netw 1(1–2):24–45
Kiayias A, Tsiounis Y, Yung M (2004) Traceable signatures. In: International conference on the theory and applications of cryptographic techniques. Springer, pp 571–589
Laguillaumie F, Langlois A, Libert B, Stehlé, D(2013) Lattice-based group signatures with logarithmic signature size. In: International conference on the theory and application of cryptology and information security. Springer, pp 41–61
Libert B, Ling S, Mouhartem F, Nguyen K, Wang H(2016) Signature schemes with efficient protocols and dynamic group signatures from lattice assumptions. In: International conference on the theory and application of cryptology and information security. Springer, pp 373–403
Libert B, Peters T, Yung M(2012) Group signatures with almost-for-free revocation. In: Annual cryptology conference. Springer, pp 571–589
Libert B, Peters T, Yung M(2012) Scalable group signatures with revocation. In: Annual international conference on the theory and applications of cryptographic techniques. Springer, pp 609–627
Ling S, Nguyen K, Roux-Langlois A, Wang H (2018) A lattice-based group signature scheme with verifier-local revocation. Theor Comput Sci 730:1–20
Ling S, Nguyen K, Wang H, Xu Y (2019) Lattice-based group signatures: achieving full dynamicity (and deniability) with ease. Theor Comput Sci 783:71–94
Ling S, Nguyen K, Wang H(2015) Group signatures from lattices: simpler, tighter, shorter, ring-based. In: IACR international workshop on public key cryptography. Springer, pp 427–449
Ling S, Nguyen K, Wang H, Xu Y(2017) Lattice-based group signatures: achieving full dynamicity with ease. In: International conference on applied cryptography and network security. Springer, pp 293–312

Ling S, Nguyen K, Wang H, Xu Y(2019) Forward-secure group signatures from lattices. In: International conference on post-quantum cryptography. Springer, pp 44–64

Lyubashevsky V, Nguyen NK, Seiler G (2020) Practical lattice-based zero-knowledge proofs for integer relations. In: Proceedings of the 2020 ACM SIGSAC conference on computer and communications security, pp 1051–1070

Micciancio D, Peikert C(2013) Hardness of sis and lwe with small parameters. In: Annual cryptology conference. Springer, pp 21–39

Nguyen L (2005) Accumulators from bilinear pairings and applications to id-based ring signatures and group membership revocation. In: Topics in cryptology-CT-RSA 2005, pp 275–292

Nguyen PQ, Zhang J, Zhang Z (2015) Simpler efficient group signatures from lattices. In: IACR international workshop on public key cryptography. Springer, pp 401–426

Peikert C (2015) A decade of lattice cryptography. Cryptology ePrint Archive

Regev O (2009) On lattices, learning with errors, random linear codes, and cryptography. J ACM 56(6):1–40

Shor PW (1994) Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings 35th annual symposium on foundations of computer science, pp 124–134 . IEEE

Song DX (2001) Practical forward secure group signature schemes. In: Proceedings of the 8th ACM conference on computer and communications security, pp 225–234

Yang R, Au MH, Zhang Z, Xu Q, Yu Z, Whyte W (2019) Efficient lattice-based zero-knowledge arguments with standard soundness: construction and applications. In: Annual international cryptology conference. Springer, pp 147–175

Publisher's Note