Lattice-Based Group Signatures: Achieving Full Dynamicity with Ease

San Ling, Khoa Nguyen, Huaxiong Wang, Yanhong Xu

Division of Mathematical Sciences,
School of Physical and Mathematical Sciences,
Nanyang Technological University, Singapore.
{lingsan,khoantt,hxwang,xu0014ng}@ntu.edu.sg

Abstract. Lattice-based group signature is an active research topic in recent years. Since the pioneering work by Gordon, Katz and Vaikuntanathan (Asiacrypt 2010), eight other schemes have been proposed, providing various improvements in terms of security, efficiency and functionality. However, most of the existing constructions work only in the static setting where the group population is fixed at the setup phase. The only two exceptions are the schemes by Langlois et al. (PKC 2014) that handles user revocations (but new users cannot join), and by Libert et al. (Asiacrypt 2016) which addresses the orthogonal problem of dynamic user enrollments (but users cannot be revoked).

In this work, we provide the first lattice-based group signature that offers full dynamicity (i.e., users have the flexibility in joining and leaving the group), and thus, resolve a prominent open problem posed by previous works. Moreover, we achieve this non-trivial feat in a relatively simple manner. Starting with Libert et al.'s fully static construction (Eurocrypt 2016) - which is arguably the most efficient lattice-based group signature to date, we introduce simple-but-insightful tweaks that allow to upgrade it directly into the fully dynamic setting. More startlingly, our scheme even produces slightly shorter signatures than the former. The scheme satisfies the strong security requirements of Bootle et al.'s model (ACNS 2016), under the Short Integer Solution (SIS) and the Learning With Errors (LWE) assumptions.

1 Introduction

Group signature, introduced by Chaum and van Heyst [15], is a fundamental anonymity primitive which allows members of a group to sign messages on behalf of the whole group. Yet, users are kept accountable for the signatures they issue since a tracing authority can identify them should the need arise. There have been numerous works on group signatures in the last quarter-century.

Ateniese et al. [2] proposed the first scalable instantiation meeting the security properties that can be intuitively expected from the primitive, although clean security notions were not available yet at that time. Bellare et al. [4] filled this gap by providing strong security notions for static groups, in which the group population is fixed at the setup phase. Subsequently, Kiayias and Yung [24] and

Bellare et al. [5] established the models capturing the partially dynamic setting, where users are able to join the group at any time, but once they have done so, they cannot leave the group. Sakai et al. [45] strengthened these models by suggesting the notion of opening soundness, guaranteeing that a valid signature only traces to one user. Efficient schemes satisfying these models have been proposed in the random oracle model [41,17] and in the standard model [21,33].

One essential functionality of group signatures is the support for membership revocation. Enabling this feature in an efficient manner is quite challenging, since one has to ensure that revoked users are no longer able to sign messages and the workloads of other parties (managers, non-revoked users, verifiers) do not significantly increase in the meantime. Several different approaches have been suggested [10,12,47,40,6] to address this problem, and notable pairing-based constructions supporting both dynamic joining and efficient revocation were given in [38,32,31]. Very recently, Bootle et al. [7] pointed out a few shortcomings of previous models, and put forward stringent security notions for fully dynamic group signatures. They also demonstrated a construction satisfying these notions based on the decisional Diffie-Hellman (DDH) assumption, following a generic transformation from a secure accountable ring signature scheme [8].

For the time being, existing schemes offering full dynamicity all rely on number-theoretic assumptions which are vulnerable to quantum attacks. To avoid putting all eggs in one basket, it is thus encouraging to consider instantiations based on alternative, post-quantum foundations, e.g., lattice assumptions. In view of this, let us now look at the topic of lattice-based group signatures.

Lattice-based group signatures. Lattice-based cryptography has been an exciting research area since the seminal works of Regev [43] and Gentry et al. [19]. Along with other primitives, lattice-based group signature has received noticeable attention in recent years. The first scheme was introduced by Gordon, Katz and Vaikuntanathan [20] whose solution produced signature size linear in the number of group users N. Camenisch et al. [13] then extended [20] to achieve anonymity in the strongest sense. Later, Laguillaumie *et al.* [25] put forward the first scheme with the signature size logarithmic in N, at the cost of relatively large parameters. Simpler and more efficient solutions with $\mathcal{O}(\log N)$ signature size were subsequently given by Nguyen et al. [42] and Ling et al. [35]. Libert et al. [29] obtained substantial efficiency improvements via a construction based on Merkle trees which eliminates the need for GPV trapdoors [19]. More recently, a scheme supporting message-dependent opening (MDO) feature [44] was proposed in [30]. All the schemes mentioned above are designed for static groups.

The only two known lattice-based group signatures that have certain dynamic features were proposed by Langlois et al. [26] and Libert et al. [27]. The former is a scheme with verifier-local revocation (VLR) [6], which means that only the verifiers need to download the up-to-date group information. The latter addresses the orthogonal problem of dynamic user enrollments (but users cannot be revoked). To achieve those partially dynamic functionalities, both of

the proposals have to incorporate relatively complicated mechanisms[1] and both heavily rely on lattice trapdoors.

The discussed above situation is somewhat unsatisfactory, given that the full dynamicity feature is highly desirable in most applications of group signatures (e.g., protecting the privacy of commuters in public transportation), and it has been achieved based on number-theoretic assumptions. This motivates us to work on fully dynamic group signatures from lattices. Furthermore, considering that the journey to partial dynamicity in previous works [26,27] was shown not easy, we ask ourselves an inspiring question: Can we achieve full dynamicity with ease? At the end of the day, it is good to solve an open research question, but it would be even better and more exciting to do this in a simple way. To make it possible, we will likely need some new and insightful ideas.

Our Results and Techniques. We introduce the first fully dynamic group signature from lattices. The scheme satisfies the strong security requirements put forward by Bootle et al. [7], under the Short Integer Solution (SIS) and the Learning With Errors (LWE) assumptions. As in all previous lattice-based group signatures, our scheme is analyzed in the random oracle model.

For a security parameter parameter λ and maximum expected number of group users N, our scheme features signature size $\widetilde{\mathcal{O}}(\lambda \cdot \log N)$ and group public key size $\widetilde{\mathcal{O}}(\lambda^2 + \lambda \cdot \log N)$. The user's secret key has bit-size $\widetilde{\mathcal{O}}(\lambda) + \log N$. At each epoch when the group information is updated, the verifiers only need to download an extra $\widetilde{\mathcal{O}}(\lambda)$ bits in order to perform verification of signatures[2], while each active signer only has to download $\widetilde{\mathcal{O}}(\lambda \cdot \log N)$ bits. In Table 1, we give a detailed comparison of our scheme with known lattice-based group signatures, in terms of efficiency and functionality. The full dynamicity feature is achieved with a very reasonable cost and without having to rely on lattice trapdoors. Somewhat surprisingly, our scheme even produces shorter signatures than the scheme from [29] - which is arguably the most efficient lattice-based group signature known to date. Furthermore, these results are obtained in a relatively simple manner, thanks to three main ideas/techniques discussed below.

Our starting point is the scheme [29], which works in the static setting. Instead of relying on trapdoor-based ordinary signatures as in prior works, the LLNW scheme employs on a SIS-based Merkle tree accumulator. For a group of $N = 2^\ell$ users, the manager chooses uniformly random vectors $\mathbf{x}_0, \ldots, \mathbf{x}_{N-1}$; hashes them to $\mathbf{p}_0, \ldots, \mathbf{p}_{N-1}$, respectively; builds a tree on top of these hash values; and publishes the tree root $\mathbf{u}$. The signing key of user i consists of $\mathbf{x}_i$

[1] Langlois et al. considered users' "tokens" as functions of Bonsai signatures [14] and associated them with a sophisticated revocation technique, while Libert et al. used a variant of Boyen's signature [9] to sign users' public keys. Both underlying signature schemes require long keys and lattice trapdoors.

[2] We remark that in the DDH-based instantiation from [7] which relies on the accountable ring signature from [8], the verifiers have to periodically download public keys of active signers. Our scheme overcomes this issue, thanks to the use of an updatable accumulator constructed in Section 3.

Scheme	Sig. size	Group PK size	Signer's SK size	Trap-door?	Model	Extra info per epoch
GKV [20]	$\widetilde{\mathcal{O}}(\lambda^2 \cdot N)$	$\widetilde{\mathcal{O}}(\lambda^2 \cdot N)$	$\widetilde{\mathcal{O}}(\lambda^2)$	yes	static	NA
CNR [13]	$\widetilde{\mathcal{O}}(\lambda^2 \cdot N)$	$\widetilde{\mathcal{O}}(\lambda^2)$	$\widetilde{\mathcal{O}}(\lambda^2)$	yes	static	NA
LLLS [25]	$\widetilde{\mathcal{O}}(\lambda \cdot \ell)$	$\mathcal{O}(\lambda^2 \cdot \ell)$	$\widetilde{\mathcal{O}}(\lambda^2)$	yes	static	NA
LLNW [26]	$\widetilde{\mathcal{O}}(\lambda \cdot \ell)$	$\widetilde{\mathcal{O}}(\lambda^2 \cdot \ell)$	$\widetilde{\mathcal{O}}(\lambda \cdot \ell)$	yes	VLR	Sign: **no** / Ver: $\widetilde{\mathcal{O}}(\lambda) \cdot R$
NZZ [42]	$\widetilde{\mathcal{O}}(\lambda + \ell^2)$	$\widetilde{\mathcal{O}}(\lambda^2 \cdot \ell^2)$	$\widetilde{\mathcal{O}}(\lambda^2)$	yes	static	NA
LNW [35]	$\widetilde{\mathcal{O}}(\lambda \cdot \ell)$	$\widetilde{\mathcal{O}}(\lambda^2 \cdot \ell)$	$\widetilde{\mathcal{O}}(\lambda)$	yes	static	NA
LLNW [29]	$\widetilde{\mathcal{O}}(\lambda \cdot \ell)$	$\widetilde{\mathcal{O}}(\lambda^2 + \lambda \cdot \ell)$	$\widetilde{\mathcal{O}}(\lambda \cdot \ell)$	**FREE**	static	NA
LLM+ [27]	$\widetilde{\mathcal{O}}(\lambda \cdot \ell)$	$\widetilde{\mathcal{O}}(\lambda^2 \cdot \ell)$	$\widetilde{\mathcal{O}}(\lambda)$	yes	partially dynamic	NA
LMN [30]	$\widetilde{\mathcal{O}}(\lambda \cdot \ell)$	$\widetilde{\mathcal{O}}(\lambda^2 \cdot \ell)$	$\widetilde{\mathcal{O}}(\lambda)$	yes	MDO	NA
Ours	$\widetilde{\mathcal{O}}(\lambda \cdot \ell)$	$\widetilde{\mathcal{O}}(\lambda^2 + \lambda \cdot \ell)$	$\widetilde{\mathcal{O}}(\lambda) + \ell$	**FREE**	fully dynamic	Sign: $\widetilde{\mathcal{O}}(\lambda \cdot \ell)$ / Ver: $\widetilde{\mathcal{O}}(\lambda)$

Table 1. Comparison of known lattice-based group signatures, in terms of efficiency and functionality. The comparison is done based on two governing parameters: security parameter λ and the maximum expected number of group users $N = 2^\ell$. As for the scheme from [26], R denotes the number of revoked users at the epoch in question.

and the witness for the fact that $\mathbf{p}_i$ was accumulated in $\mathbf{u}$. When issuing signatures, the user proves knowledge of a valid pair $(\mathbf{x}_i, \mathbf{p}_i)$ and of the tree path from $\mathbf{p}_i$ to $\mathbf{u}$. The user also has to encrypt the binary representation $\mathsf{bin}(i)$ of his identity i, and prove that the ciphertext is well-formed. The encryption layer is also lattice-trapdoor-free, since it utilizes the Naor-Yung double-encryption paradigm [39] with Regev's LWE-based encryption scheme. To upgrade the LLNW scheme directly into a fully dynamic group signature, we now let the user compute the pair $(\mathbf{x}_i, \mathbf{p}_i)$ on his own (for enabling non-frameability), and we employ the following three ideas/techniques.

First, we add a dynamic ingredient into the static Merkle tree accumulator from [29]. To this end, we equip it with an efficient updating algorithm with complexity $\mathcal{O}(\log N)$: to change an accumulated value, we simply update the values at the corresponding leaf and along its path to the root.

Second, we create a simple rule to handle user enrollment and revocation efficiently (i.e., without resetting the whole tree). Specifically, we use the updating algorithm to set up the system so that: (i)- If a user has not joined the group or has been revoked, the value at the leaf associated with him is set as $\mathbf{0}$; (ii)- When a user joins the group, that value is set as his public key $\mathbf{p}_i$. Our setup guarantees that only active users (i.e., who has joined and has not been revoked at the given epoch) have their *non-zero* public keys accumulated into the updated root. This rule effectively separates active users who can sign from

those who cannot: when signing messages, the user proceeds as in the LLNW scheme, and is asked to additionally prove in zero-knowledge that $\mathbf{p}_i \neq \mathbf{0}$. In other words, the seemingly big gap between being fully static and being fully dynamic has been reduced to a small difference!

Third, the arising question now is how to additionally prove the inequality $\mathbf{p}_i \neq \mathbf{0}$ in the framework of the Stern-like [46] protocol from [29]. One would naturally expect that this extra job could be done without losing too much in terms of efficiency. Here, the surprising and somewhat unexpected fact is that we can actually do it while *gaining* efficiency, thanks to the following simple idea. Recall that, in [29], to prove knowledge of $\mathbf{p}_i \in \{0,1\}^{nk}$, an extension technique from [34] is employed, in which $\mathbf{p}_i$ is extended into a vector of dimension $2nk$. We note that, the authors of [34] also suggested a slightly modified version of their technique, that allows to simultaneously prove that $\mathbf{p}_i \in \{0,1\}^{nk}$ and $\mathbf{p}_i$ is non-zero while working only with dimension $2nk - 1$. This intriguing tweak enables us to obtain a zero-knowledge protocol with slightly lower communication cost, and hence, group signatures with slightly smaller size than in [29].

To summarize, we solve a prominent open question in the field of lattice-based group signatures. Moreover, our solution is simple and comparatively efficient. Our results, while not yielding a truly practical scheme, would certainly help to bring the field one step closer to practice.

Organization. In Section 2, we recall some background on fully dynamic group signatures and lattice-based cryptography. Section 3 develops an updatable Merkle tree accumulator. Our main scheme is constructed and analyzed in Section 4.

2 Preliminaries

2.1 Fully Dynamic Group Signatures

We recall the definition and security notions of fully dynamic group signatures (FDGS) presented by Bootle et al. [7]. A FDGS scheme involves the following entities: a group manager GM that determines who can join the group, a tracing manager TM who can open signatures, and a set of users who are potential group members. Users can join/leave the group at the discretion of GM. We assume GM will publish some information info_τ regularly associated with a distinct index τ (referred as epoch hereafter). Without loss of generality, assume there is one-to-one correspondence between information and the associated epoch. The information describes changes of the group, e.g., current group members or members that are excluded from the group. We assume the published group information is authentic. By comparing current group information with previous one, it allows any party to identify revoked users at current epoch. For simplicity assume $\tau_1 < \tau_2$ if info_{τ_1} is published before info_{τ_2}, i.e., the epoch preserves the order in which the corresponding group information was published. In existing models, the keys of authorities are supposed to be generated honestly; while in [7], Bootle et al. consider a stronger model where the keys of authorities can be maliciously generated by the adversary.

Syntax of fully dynamic group signatures. A FDGS scheme is a tuple of following polynomial-time algorithms.

- $\mathsf{GSetup}(\lambda) \to \mathsf{pp}$. On input security parameter λ, this algorithm generates public parameters pp.
- $\langle \mathsf{GKgen}_{\mathsf{GM}}(\mathsf{pp}), \mathsf{GKgen}_{\mathsf{TM}}(\mathsf{pp}) \rangle$. This is an interactive protocol between the group manager GM and the tracing manager TM. If it completes successfully, algorithm $\mathsf{GKgen}_{\mathsf{GM}}$ outputs a manager key pair $(\mathsf{mpk}, \mathsf{msk})$. Meanwhile, GM initializes the group information info and the registration table **reg**. The algorithm $\mathsf{GKgen}_{\mathsf{TM}}$ outputs a tracing key pair $(\mathsf{tpk}, \mathsf{tsk})$. Set group public key $\mathsf{gpk} = (\mathsf{pp}, \mathsf{mpk}, \mathsf{tpk})$.
- $\mathsf{UKgen}(\mathsf{pp}) \to (\mathsf{upk}, \mathsf{usk})$. Given public parameters pp, this algorithm generates a user key pair $(\mathsf{upk}, \mathsf{usk})$.
- $\langle \mathsf{Join}(\mathsf{info}_{\tau_{\mathrm{current}}}, \mathsf{gpk}, \mathsf{upk}, \mathsf{usk}); \mathsf{Issue}(\mathsf{info}_{\tau_{\mathrm{current}}}, \mathsf{msk}, \mathsf{upk}) \rangle$. This is an interactive algorithm run by a user and GM. If it completes successfully, this user becomes a group member with an identifier uid and the Join algorithm stores secret group signing key $\mathsf{gsk}[\mathsf{uid}]$ while Issue algorithm stores registration information in the table **reg** with index uid.
- $\mathsf{GUpdate}(\mathsf{gpk}, \mathsf{msk}, \mathsf{info}_{\tau_{\mathrm{current}}}, \mathcal{S}, \mathbf{reg}) \to \mathsf{info}_{\tau_{\mathrm{new}}}$. This is an algorithm run by GM to update group information while advancing the epoch. Given gpk,msk, $\mathsf{info}_{\tau_{\mathrm{current}}}$, registration table **reg**, a set $\mathcal{S}$ of active users to be removed from the group, GM computes new group information $\mathsf{info}_{\tau_{\mathrm{new}}}$ and may update the table **reg**. If there is no change to the group, GM outputs $\perp$.
- $\mathsf{Sign}(\mathsf{gpk}, \mathsf{gsk}[\mathsf{uid}], \mathsf{info}_{\tau}, M) \to \Sigma$. This algorithm outputs a group signature Σ on message M by user uid. It outputs $\perp$ if this user is inactive at epoch τ.
- $\mathsf{Verify}(\mathsf{gpk}, \mathsf{info}_{\tau}, M, \Sigma) \to 0/1$. This algorithm checks the validity of the signature Σ on message M at epoch τ.
- $\mathsf{Trace}(\mathsf{gpk}, \mathsf{tsk}, \mathsf{info}_{\tau}, \mathbf{reg}, M, \Sigma) \to (\mathsf{uid}, \Pi_{\mathsf{trace}})$. This is an algorithm run by TM. Given the inputs, TM returns an identity uid of a group member who signed the message and a proof indicating this tracing result or $\perp$ if it fails to trace to a group member.
- $\mathsf{Judge}(\mathsf{gpk}, \mathsf{uid}, \mathsf{info}_{\tau}, \Pi_{\mathsf{trace}}, M, \Sigma) \to 0/1$. This algorithm checks the validity of Π_{trace} outputted by the Trace algorithm.

Correctness and security of fully dynamic group signatures. As put forward by Bootle et al. [7], a FDGS scheme must satisfy *correctness*, *anonymity*, *non-frameability*, *traceability* and *tracing soundness*.

Correctness demands that a signature generated by an honest and active user is always accepted by algorithm Verify, and that algorithm Trace can always identify that user, as well as produces a proof accepted by algorithm Judge.

Anonymity requires that it is infeasible for any PPT adversary to distinguish signatures generated by two active users of its choice at the chosen epoch, even if it can corrupt any user, can choose the key of GM, and is given access to the Trace oracle.

Non-Frameability makes sure that the adversary cannot generate a valid signature that traces to an honest user even if it can corrupt all other users, and can choose keys of both managers.

Traceability ensures that the adversary cannot produce a valid signature that cannot be traced to an active user at the chosen epoch, even if it can corrupt any user and can choose the key of TM.

Tracing Soundness requires that it is infeasible to produce a valid signature that traces to two different users, even if all group users and both managers are fully controlled by the adversary.

Formal definitions of correctness and security requirements are deferred to Appendix B.

2.2 Background on Lattices

We recall the average-case lattice problems SIS and LWE, together with their hardness results.

Definition 1 ([1,19]). The $\mathsf{SIS}^{\infty}_{n,m,q,\beta}$ problem is as follows: Given uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, find a non-zero vector $\mathbf{x} \in \mathbb{Z}^m$ such that $\|\mathbf{x}\|_\infty \leq \beta$ and $\mathbf{A} \cdot \mathbf{x} = \mathbf{0} \bmod q$.

If $m, \beta = \mathsf{poly}(n)$, and $q > \beta \cdot \widetilde{\mathcal{O}}(\sqrt{n})$, then the $\mathsf{SIS}^{\infty}_{n,m,q,\beta}$ problem is at least as hard as worst-case lattice problem SIVP_γ for some $\gamma = \beta \cdot \widetilde{\mathcal{O}}(\sqrt{nm})$ (see, e.g., [19,37]). Specifically, when $\beta = 1$, $q = \widetilde{\mathcal{O}}(n)$, $m = 2n\lceil \log q \rceil$, the $\mathsf{SIS}^{\infty}_{n,m,q,1}$ problem is at least as hard as SIVP_γ with $\gamma = \widetilde{\mathcal{O}}(n)$.

In the last decade, numerous SIS-based cryptographic primitives have been proposed. In this work, we will extensively employ 2 such constructions:

- Our group signature scheme is based on the Merkle tree accumulator from [29], which is built upon a specific family of collision-resistant hash functions.
- Our zero-knowledge argument systems use the statistically hiding and computationally binding string commitment scheme from [23].

For appropriate setting of parameters, the security of the above two constructions can be based on the worst-case hardness of $\mathsf{SIVP}_{\widetilde{\mathcal{O}}(n)}$.

In the group signature in Section 4, we will employ the multi-bit version of Regev's encryption scheme [43], presented in [22]. The scheme is based on the hardness of the LWE problem.

Definition 2 ([43]). Let $n, m \geq 1$, $q \geq 2$, and let χ be a probability distribution on $\mathbb{Z}$. For $\mathbf{s} \in \mathbb{Z}_q^n$, let $\mathcal{A}_{\mathbf{s},\chi}$ be the distribution obtained by sampling $\mathbf{a} \xleftarrow{\$} \mathbb{Z}_q^n$ and $e \hookleftarrow \chi$, and outputting $(\mathbf{a}, \mathbf{s}^\top \cdot \mathbf{a} + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$. The $\mathsf{LWE}_{n,q,\chi}$ problem asks to distinguish m samples chosen according to $\mathcal{A}_{\mathbf{s},\chi}$ (for $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$) and m samples chosen according to the uniform distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$.

If q is a prime power, χ is the discrete Gaussian distribution $D_{\mathbb{Z},\alpha q}$, where $\alpha q \geq 2\sqrt{n}$, then $\mathsf{LWE}_{n,q,\chi}$ is as least as hard as $\mathsf{SIVP}_{\widetilde{\mathcal{O}}(n/\alpha)}$ (see [43,36,37]).

2.3 Stern-like Protocols for Lattice-Based Relations

The zero-knowledge (ZK) argument systems appearing in this paper operate in the framework of Stern's protocol [46]. Let us now recall some background. This protocol was originally proposed in the context of code-based cryptography, and was later adapted into the lattice setting by Kawachi et al. [23]. Subsequently, it was empowered by Ling et al. [34] to handle the matrix-vector relations associated with the SIS and LWE problems, and further developed to design several lattice-based schemes: group signatures [26,35,29,30,27], policy-based signatures [16] and group encryption [28].

Stern-like protocols are quite useful in the context of lattice-based privacy-preserving systems, when one typically works with modular linear equations of the form $\sum_i \mathbf{M}_i \cdot \mathbf{x}_i = \mathbf{v} \bmod q$ - where $\{\mathbf{M}_i\}_i$, $\mathbf{v}$ are public, and one wants to prove in ZK that secret vectors $\{\mathbf{x}_i\}_i$ satisfy certain constraints, e.g., they have small norms and/or have coordinates arranged in a special way. The high-level ideas can be summarized as follows. If the given constraints are invariant under certain type of permutations of coordinates, then one readily uses uniformly random permutations to prove those constraints. Otherwise, one performs some pre-processings with $\{\mathbf{x}_i\}_i$ to reduce to the former case. Meanwhile, to prove that the modular linear equation holds, one makes use of a standard masking technique.

The basic protocol consists of 3 moves: commitment, challenge, response. If the statistically hiding and computationally binding string commitment scheme from [23] is employed in the first move, then one obtains a statistical zero-knowledge argument of knowledge (ZKAoK) with perfect completeness, constant soundness error 2/3, and communication cost $\mathcal{O}(|w| \cdot \log q)$, where $|w|$ denotes the total bit-size of the secret vectors. In many applications, the protocol is repeated $\kappa = \omega(\log \lambda)$ times, for security parameter λ, to achieve negligible soundness error, and then made non-interactive via the Fiat-Shamir heuristic [18]. In the random oracle model, this results in a non-interactive zero-knowledge argument of knowledge (NIZKAoK) with bit-size $\mathcal{O}(|w| \cdot \log q) \cdot \omega(\log \lambda)$.

3 Updatable Lattice-Based Merkle Hash Trees

We first recall the lattice-based Merkle-tree accumulator from [29], and then, we equip it with a simple updating algorithm which allows to change an accumulated value in time logarithmic in the size of the accumulated set. This updatable hash tree will serve as the building block of our construction in Section 4.

3.1 Cryptographic Accumulators

An *accumulator scheme* is a tuple of polynomial-time algorithms defined below.

$\mathsf{TSetup}(\lambda)$ On input security parameter λ, output the public parameter pp.

$\mathsf{TAcc}_{\mathsf{pp}}$ On input a set $R = \{\mathbf{d}_0, \ldots, \mathbf{d}_{N-1}\}$ of N data values, output an accumulator value $\mathbf{u}$.

$\mathsf{TWitness}_{\mathsf{pp}}$ On input a data set R and a value $\mathbf{d}$, output $\perp$ if $\mathbf{d} \notin R$; otherwise output a witness w for the fact that $\mathbf{d}$ is accumulated in $\mathsf{TAcc}(R)$. (Typically, the size of w should be short (*e.g.*, constant or logarithmic in N) to be useful.)

$\mathsf{TVerify}_{\mathsf{pp}}$ On input accumulator value $\mathbf{u}$ and a value-witness pair $(\mathbf{d}, w)$, output 1 (which indicates that $(\mathbf{d}, w)$ is valid for the accumulator $\mathbf{u}$) or 0.

An accumulator scheme is called correct if for all $\mathsf{pp} \leftarrow \mathsf{TSetup}(\lambda)$, we have $\mathsf{TVerify}_{\mathsf{pp}}\big(\mathsf{TAcc}_{\mathsf{pp}}(R), \mathbf{d}, \mathsf{TWitness}_{\mathsf{pp}}(R, \mathbf{d})\big) = 1$ for all $\mathbf{d} \in R$.

A natural security requirement for accumulators, as considered in [3,12,29], says that it is infeasible to prove that a value $\mathbf{d}^*$ was accumulated in a value $\mathbf{u}$ if it was not. This property is formalized as follows.

Definition 3 ([29]). A*n accumulator scheme* $(\mathsf{TSetup}, \mathsf{TAcc}, \mathsf{TWitness}, \mathsf{TVerify})$ *is called secure if for all PPT adversaries* $\mathcal{A}$*:*

$$\Pr\big[\mathsf{pp} \leftarrow \mathsf{TSetup}(\lambda); (R, \mathbf{d}^*, w^*) \leftarrow \mathcal{A}(\mathsf{pp}) : \\ \mathbf{d}^* \notin R \wedge \mathsf{TVerify}_{\mathsf{pp}}(\mathsf{TAcc}_{\mathsf{pp}}(R), \mathbf{d}^*, w^*) = 1\big] = \mathsf{negl}(\lambda).$$

3.2 The LLNW Merkle-tree Accumulator

NOTATIONS. Hereunder we will use the notation $x \xleftarrow{\$} S$ to indicate that x is chosen uniformly at random from finite set S. For bit $b \in \{0,1\}$, we let $\bar{b} = 1 - b$.

The Merkle-tree accumulator scheme from [29] works with parameters $n = \mathcal{O}(\lambda)$, $q = \widetilde{\mathcal{O}}(n^{1.5})$, $k = \lceil \log_2 q \rceil$, and $m = 2nk$. The set $\mathbb{Z}_q$ is identified by $\{0, \ldots, q-1\}$. Define the "powers-of-2" matrix

$$\mathbf{G} = \begin{bmatrix} 1\ 2\ 4\ \ldots\ 2^{k-1} & & \\ & \cdots & \\ & & 1\ 2\ 4\ \ldots\ 2^{k-1} \end{bmatrix} \in \mathbb{Z}_q^{n \times nk}.$$

Note that for every $\mathbf{v} \in \mathbb{Z}_q^n$, we have $\mathbf{v} = \mathbf{G} \cdot \mathsf{bin}(\mathbf{v})$, where $\mathsf{bin}(\mathbf{v}) \in \{0,1\}^{nk}$ denotes the binary representation of $\mathbf{v}$. The scheme is built upon the following family of SIS-based collision-resistant hash functions.

Definition 4. *The function family* $\mathcal{H}$ *mapping* $\{0,1\}^{nk} \times \{0,1\}^{nk}$ *to* $\{0,1\}^{nk}$ *is defined as* $\mathcal{H} = \{h_{\mathbf{A}} \mid \mathbf{A} \in \mathbb{Z}_q^{n \times m}\}$*, where for* $\mathbf{A} = [\mathbf{A}_0 | \mathbf{A}_1]$ *with* $\mathbf{A}_0, \mathbf{A}_1 \in \mathbb{Z}_q^{n \times nk}$*, and for any* $(\mathbf{u}_0, \mathbf{u}_1) \in \{0,1\}^{nk} \times \{0,1\}^{nk}$*, we have:*

$$h_{\mathbf{A}}(\mathbf{u}_0, \mathbf{u}_1) = \mathsf{bin}\big(\mathbf{A}_0 \cdot \mathbf{u}_0 + \mathbf{A}_1 \cdot \mathbf{u}_1 \bmod q\big) \in \{0,1\}^{nk}.$$

Note that $h_{\mathbf{A}}(\mathbf{u}_0, \mathbf{u}_1) = \mathbf{u} \Leftrightarrow \mathbf{A}_0 \cdot \mathbf{u}_0 + \mathbf{A}_1 \cdot \mathbf{u}_1 = \mathbf{G} \cdot \mathbf{u} \bmod q$.

A Merkle tree with $N = 2^\ell$ leaves, where ℓ is a positive integer, then can be constructed based on the function family $\mathcal{H}$ as follows.

$\mathsf{TSetup}(\lambda)$. Sample $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$, and output $\mathsf{pp} = \mathbf{A}$.

$\mathsf{TAcc}_{\mathbf{A}}(R = \{\mathbf{d}_0 \in \{0,1\}^{nk}, \ldots, \mathbf{d}_{N-1} \in \{0,1\}^{nk}\})$. For every $j \in [0, N-1]$, let $\mathsf{bin}(j) = (j_1, \ldots, j_\ell) \in \{0,1\}^\ell$ be the binary representation of j, and let $\mathbf{d}_j = \mathbf{u}_{j_1,\ldots,j_\ell}$. Form the tree of depth $\ell = \log N$ based on the N leaves $\mathbf{u}_{0,0,\ldots,0}, \ldots, \mathbf{u}_{1,1,\ldots,1}$ as follows:

1. At depth $i \in [\ell]$, the node $\mathbf{u}_{b_1,\ldots,b_i} \in \{0,1\}^{nk}$, for all $(b_1, \ldots, b_i) \in \{0,1\}^i$, is defined as $h_{\mathbf{A}}(\mathbf{u}_{b_1,\ldots,b_i,0}, \mathbf{u}_{b_1,\ldots,b_i,1})$.
2. At depth 0: The root $\mathbf{u} \in \{0,1\}^{nk}$ is defined as $h_{\mathbf{A}}(\mathbf{u}_0, \mathbf{u}_1)$.

The algorithm outputs the accumulator value $\mathbf{u}$.

$\mathsf{TWitness}_{\mathbf{A}}(R, \mathbf{d})$. If $\mathbf{d} \notin R$, return $\perp$. Otherwise, $\mathbf{d} = \mathbf{d}_j$ for some $j \in [0, N-1]$ with binary representation $(j_1, \ldots, j_\ell)$. Output the witness w defined as:

$$w = \big((j_1, \ldots, j_\ell), (\mathbf{u}_{j_1,\ldots,j_{\ell-1},\bar{j_\ell}}, \ldots, \mathbf{u}_{j_1,\bar{j_2}}, \mathbf{u}_{\bar{j_1}})\big) \in \{0,1\}^\ell \times \big(\{0,1\}^{nk}\big)^\ell,$$

for $\mathbf{u}_{j_1,\ldots,j_{\ell-1},\bar{j_\ell}}, \ldots, \mathbf{u}_{j_1,\bar{j_2}}, \mathbf{u}_{\bar{j_1}}$ computed by algorithm $\mathsf{TAcc}_{\mathbf{A}}(R)$.

$\mathsf{TVerify}_{\mathbf{A}}\big(\mathbf{u}, \mathbf{d}, w\big)$. Let the given witness w be of the form:

$$w = \big((j_1, \ldots, j_\ell), (\mathbf{w}_\ell, \ldots, \mathbf{w}_1)\big) \in \{0,1\}^\ell \times \big(\{0,1\}^{nk}\big)^\ell.$$

The algorithm recursively computes the path $\mathbf{v}_\ell, \mathbf{v}_{\ell-1}, \ldots, \mathbf{v}_1, \mathbf{v}_0 \in \{0,1\}^{nk}$ as follows: $\mathbf{v}_\ell = \mathbf{d}$ and

$$\forall i \in \{\ell - 1, \ldots, 1, 0\} : \mathbf{v}_i = \begin{cases} h_{\mathbf{A}}(\mathbf{v}_{i+1}, \mathbf{w}_{i+1}), \text{ if } j_{i+1} = 0; \\ h_{\mathbf{A}}(\mathbf{w}_{i+1}, \mathbf{v}_{i+1}), \text{ if } j_{i+1} = 1. \end{cases} \tag{1}$$

Then it returns 1 if $\mathbf{v}_0 = \mathbf{u}$. Otherwise, it returns 0.

The following lemma states the correctness and security of the above Merkle tree accumulator.

Lemma 1 ([29]). *The given accumulator scheme is correct and is secure in the sense of Definition 3, assuming the hardness of the* $\mathsf{SIS}^\infty_{n,m,q,1}$ *problem.*

3.3 An Efficient Updating Algorithm

Unlike the static group signature scheme from [29], our fully dynamic construction of Section 4 requires to regularly edit the accumulated values without having to reconstruct the whole tree. To this end, we equip the Merkle tree accumulator from [29] with a simple, yet efficient, updating algorithm: to change the value at a given leaf, we simply modify all values in the path from that leaf to the root. The algorithm, which takes as input a bit string $\mathsf{bin}(j) = (j_1, j_2, \ldots, j_\ell)$ and a value $\mathbf{d}^* \in \{0,1\}^{nk}$, is formally described below.

Given the tree in Section 3.2, algorithm $\mathsf{TUpdate}_{\mathbf{A}}((j_1, j_2, \ldots, j_\ell), \mathbf{d}^*)$ performs the following steps:

1. Let $\mathbf{d}_j$ be the current value at the leaf of position determined by $\mathsf{bin}(j)$, and let $((j_1, \ldots, j_\ell), (\mathbf{w}_{j,\ell}, \ldots, \mathbf{w}_{j,1}))$ be its associated witness.

2. Set $\mathbf{v}_\ell := \mathbf{d}^*$ and recursively compute the path $\mathbf{v}_\ell, \mathbf{v}_{\ell-1}, \ldots, \mathbf{v}_1, \mathbf{v}_0 \in \{0,1\}^{nk}$ as in (1).
3. Set $\mathbf{u} := \mathbf{v}_0$; $\mathbf{u}_{j_1} := \mathbf{v}_1; \ldots; \mathbf{u}_{j_1,j_2,\ldots,j_{\ell-1}} := \mathbf{v}_{\ell-1}$; $\mathbf{u}_{j_1,j_2,\ldots,j_\ell} := \mathbf{v}_\ell = \mathbf{d}^*$.

It can be seen that the provided algorithm runs in time $\mathcal{O}(\ell) = \mathcal{O}(\log N)$. In **Fig. 1**, we give an illustrative example of a tree with $2^3 = 8$ leaves.

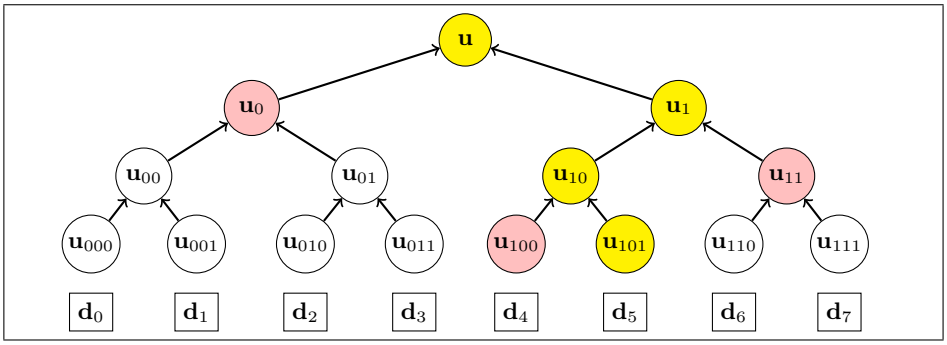

Fig. 1: A Merkle tree with $2^3 = 8$ leaves, which accumulates the data blocks $\mathbf{d}_0, \ldots, \mathbf{d}_7$ into the value $\mathbf{u}$ at the root. The bit string (101) and the pink nodes form a witness to the fact that $\mathbf{d}_5$ is accumulated in $\mathbf{u}$. If we replace $\mathbf{d}_5$ by a new value $\mathbf{d}^*$, we only need to update the yellow nodes.

4 Our Fully Dynamic Group Signatures from Lattices

In this section, we construct our lattice-based fully dynamic group signature and prove its security in Bootle et al.'s model [7]. We start with the LLNW scheme [29], which works in the static setting.

While other constructions of lattice-based group signatures employ trapdoor-based ordinary signature schemes (e.g., [9,14]) to certify users, the LLNW scheme relies on a SIS-based Merkle tree accumulator which we recalled in Section 3.2. The GM, who manages a group of $N = 2^\ell$ users, chooses uniformly random vectors $\mathbf{x}_0, \ldots, \mathbf{x}_{N-1} \in \{0,1\}^m$; hashes them to $\mathbf{p}_0, \ldots, \mathbf{p}_{N-1} \in \{0,1\}^{nk}$, respectively; builds a tree on top of these hash values; and lets the tree root $\mathbf{u} \in \{0,1\}^{nk}$ be part of the group public key. The signing key of user i consists of $\mathbf{x}_i$ and the witness for the fact that $\mathbf{p}_i$ was accumulated in $\mathbf{u}$. When generating group signatures, the user proves knowledge of a valid pair $(\mathbf{x}_i, \mathbf{p}_i)$ and of the tree path from $\mathbf{p}_i$ to $\mathbf{u}$. The user also has to encrypt the binary representation $\mathsf{bin}(i)$ of his identity i, and prove that the ciphertext is well-formed. The encryption layer utilizes the Naor-Yung double-encryption paradigm [39] with Regev's LWE-based cryptosystem, and thus, it is also lattice-trapdoor-free.

To upgrade the LLNW scheme directly into a fully dynamic group signature, some tweaks and new ideas are needed. First, to enable the non-frameability

feature, we let the user compute the pair $(\mathbf{x}_i, \mathbf{p}_i)$ on his own. The second problem we have to consider is that Merkle hash trees seem to be a static primitive. To this end, we equip the accumulator with an efficient updating algorithm (see Section 3.3). Now, the challenging question is how to handle user enrollment and revocation in a simple manner (i.e., without having to reset the whole tree). To tackle these issues, we associate each of the N potential users with a leaf in the tree, and then use the updating algorithm to set up the system so that:

1. If a user has not joined the group or has been revoked, the value at the leaf associated with him is set as $\mathbf{0}$;
2. When a user joins the group, that value is set as his public key $\mathbf{p}_i$.

Our setup guarantees that only active users (i.e., who has joined and has not been revoked at the given epoch) have their *non-zero* public keys accumulated into the updated root. This effectively gives us a method to separate active users who can sign from those who cannot: when signing messages, the user proceeds as in the LLNW scheme, and is asked to additionally prove in ZK that $\mathbf{p}_i \neq \mathbf{0}$.

At this point, the arising question is how to prove the inequality $\mathbf{p}_i \neq \mathbf{0}$ in the framework of the Stern-like [46] protocol from [29]. One would naturally hope that this extra job could be done without losing too much in terms of efficiency. Here, the surprising and somewhat unexpected fact is that we can actually do it while *gaining* efficiency, thanks to a technique originally proposed in [34].

To begin with, let B_t^L denote the set of all vectors in $\{0,1\}^L$ having Hamming weight exactly t. In Stern-like protocols (see Section 2.3), a common technique for proving in ZK the possession of $\mathbf{p} \in \{0,1\}^{nk}$ consists of appending nk "dummy" entries to it to obtain $\mathbf{p}^* \in \mathsf{B}_{nk}^{2nk}$, and demonstrating to the verifier that a random permutation of $\mathbf{p}^*$ belongs to the "target set" B_{nk}^{2nk}. This suffices to convince the verifier that the original vector $\mathbf{p}$ belongs to $\{0,1\}^{nk}$, while the latter cannot learn any additional information about $\mathbf{p}$, thanks to the randomness of the permutation. This extending-then-permuting technique was first proposed in [34], and was extensively used in the underlying protocol of the LLNW scheme. Now, to address our question, we will employ a modified version of this technique, which was also initially suggested in [34]. Let us think of another "target set", so that it is possible to extend $\mathbf{p} \in \{0,1\}^{nk}$ to an element of that set if and only if $\mathbf{p}$ is non-zero. That set is B_{nk}^{2nk-1}. Indeed, the extended vector $\mathbf{p}^*$ belongs to B_{nk}^{2nk-1} if and only if the original vector has Hamming weight at least $nk - (nk-1) = 1$, which means that it cannot be a zero-vector. When combining with the permuting step, this modification allows us to additionally prove the given inequality while working with smaller dimension. As a result, our fully dynamic scheme produces slightly shorter signatures than the original static scheme.

Finally, we remark that the fully dynamic setting requires a proof of correct opening, which boils down to proving correct decryption for Regev's encryption scheme. It involves modular linear equations with bounded-norm secrets, and can be easily handled using Stern-like techniques from [34,27].

4.1 Description of the Scheme

Our scheme is described as follows.

$\mathsf{GSetup}(\lambda)$**.** On input security parameter λ, this algorithm specifies the following:

- An expected number of potential users $N = 2^\ell = \mathsf{poly}(\lambda)$.
- Dimension $n = \mathcal{O}(\lambda)$, prime modulus $q = \widetilde{\mathcal{O}}(n^{1.5})$, and $k = \lceil \log_2 q \rceil$. These parameters implicitly determine the "powers-of-2" matrix $\mathbf{G} \in \mathbb{Z}_q^{n \times nk}$, as defined in Section 3.
- Matrix dimensions $m = 2nk$ for the hashing layer, and $m_E = 2(n+\ell)k$ for the encryption layer.
- An integer $\beta = \sqrt{n} \cdot \omega(\log n)$, and a β-bounded noise distribution χ.
- A hash function $\mathcal{H}_{\mathsf{FS}} : \{0,1\}^* \to \{1,2,3\}^\kappa$, where $\kappa = \omega(\log \lambda)$, to be modelled as a random oracle in the Fiat-Shamir transformations [18].
- Let $\mathsf{COM} : \{0,1\}^* \times \{0,1\}^m \to \mathbb{Z}_q^n$ be the string commitment scheme from [23], to be used in our zero-knowledge argument systems.
- Uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$.

The algorithm outputs public parameters

$$\mathsf{pp} = \{\lambda, N, n, q, k, m, m_E, \ell, \beta, \chi, \kappa, \mathcal{H}_{\mathsf{FS}}, \mathsf{COM}, \mathbf{A}\}.$$

$\langle \mathsf{GKgen}_{\mathsf{GM}}(\mathsf{pp}), \mathsf{GKgen}_{\mathsf{TM}}(\mathsf{pp}) \rangle$**.** The group manager GM and the tracing manager TM initialize their keys and the public group information as follows.

- $\mathsf{GKgen}_{\mathsf{GM}}(\mathsf{pp})$. This algorithm samples $\mathsf{msk} \xleftarrow{\$} \{0,1\}^m$ and computes $\mathsf{mpk} = \mathbf{A} \cdot \mathsf{msk} \bmod q$, and outputs $(\mathsf{mpk}, \mathsf{msk})$. Here, we consider mpk as an identifier of the group managed by GM who has mpk as his public key. Furthermore, as in [7, Sec. 3.3, full version], we assume that the group information board is visible to everyone, but can only be edited by a party knowing msk.
- $\mathsf{GKgen}_{\mathsf{TM}}(\mathsf{pp})$. This algorithm initializes the Naor-Yung double-encryption mechanism with the ℓ-bit version Regev encryption scheme. It first chooses $\mathbf{B} \xleftarrow{\$} \mathbb{Z}_q^{n \times m_E}$. For each $i \in \{1,2\}$, it samples $\mathbf{S}_i \xleftarrow{\$} \chi^{n \times \ell}$, $\mathbf{E}_i \hookleftarrow \chi^{\ell \times m_E}$, and computes $\mathbf{P}_i = \mathbf{S}_i^\top \cdot \mathbf{B} + \mathbf{E}_i \in \mathbb{Z}_q^{\ell \times m_E}$. Then, TM sets $\mathsf{tsk} = (\mathbf{S}_1, \mathbf{E}_1)$, and $\mathsf{tpk} = (\mathbf{B}, \mathbf{P}_1, \mathbf{P}_2)$.
- TM sends tpk to GM who initializes the following:
  - Table $\mathbf{reg} := (\mathbf{reg}[0][1], \mathbf{reg}[0][2], \ldots, \mathbf{reg}[N-1][1], \mathbf{reg}[N-1][2])$, where for each $i \in [0, N-1]$: $\mathbf{reg}[i][1] = \mathbf{0}^{nk}$ and $\mathbf{reg}[i][2] = 0$. Looking ahead, $\mathbf{reg}[i][1]$ will be used to record the public key of a registered user, while $\mathbf{reg}[i][2]$ stores the epoch at which the user joins.
  - The Merkle tree $\mathcal{T}$ built on top of $\mathbf{reg}[0][1], \ldots, \mathbf{reg}[N-1][1]$. (Note that $\mathcal{T}$ is an all-zero tree at this stage, but it will be modified when a new user joins the group, or when GM computes the updated group information.)

- Counter of registered users $c := 0$.

Then, GM outputs $\mathsf{gpk} = (\mathsf{pp}, \mathsf{mpk}, \mathsf{tpk})$ and announces the initial group information $\mathsf{info} = \emptyset$. He keeps $\mathcal{T}$ and c for himself.

$\mathsf{UKgen}(\mathsf{pp})$**.** Each potential group user samples $\mathsf{usk} = \mathbf{x} \xleftarrow{\$} \{0,1\}^m$, and computes $\mathsf{upk} = \mathbf{p} = \mathrm{bin}(\mathbf{A} \cdot \mathbf{x}) \bmod q \in \{0,1\}^{nk}$.

Without loss of generality, we assume that every honestly generated upk is a non-zero vector. (Otherwise, the user would either pick $\mathbf{x} = \mathbf{0}$ or accidentally find a solution to the $\mathsf{SIS}^{\infty}_{n,m,q,1}$ problem associated with matrix $\mathbf{A}$ - both happen with negligible probability.)

$\langle \mathsf{Join}, \mathsf{Issue} \rangle$**.** If the user with key pair $(\mathsf{upk}, \mathsf{usk}) = (\mathbf{p}, \mathbf{x})$ requests to join the group at epoch τ, he sends $\mathbf{p}$ to GM. If the latter accepts the request, then the two parties proceed as follows.

1. GM issues a member identifier for the user as $\mathsf{uid} = \mathsf{bin}(c) \in \{0,1\}^{\ell}$. The user then sets his long-term signing key as $\mathsf{gsk}[c] = (\mathsf{bin}(c), \mathbf{p}, \mathbf{x})$.
2. GM performs the following updates:
   - Update $\mathcal{T}$ by running algorithm $\mathsf{TUpdate}_{\mathbf{A}}(\mathsf{bin}(c), \mathbf{p})$.
   - Register the user to table $\mathbf{reg}$ as $\mathbf{reg}[c][1] := \mathbf{p}$; $\mathbf{reg}[c][2] := \tau$.
   - Increase the counter $c := c + 1$.

$\mathsf{GUpdate}(\mathsf{gpk}, \mathsf{msk}, \mathsf{info}_{\tau_{\mathrm{current}}}, \mathcal{S}, \mathbf{reg})$**.** This algorithm is run by GM to update the group information while also advancing the epoch. It works as follows.

1. Let the set $\mathcal{S}$ contain the public keys of registered users to be revoked. If $\mathcal{S} = \emptyset$, then go to Step 2.
   Otherwise, $\mathcal{S} = \{\mathbf{reg}[i_1][1], \ldots, \mathbf{reg}[i_r][1]\}$, for some $r \in [1, N]$ and some $i_1, \ldots, i_r \in [0, N\text{–}1]$. Then, for all $t \in [r]$, GM runs $\mathsf{TUpdate}_{\mathbf{A}}(\mathsf{bin}(i_t), \mathbf{0}^{nk})$ to update the tree $\mathcal{T}$.
2. At this point, by construction, each of the zero leaves in the tree $\mathcal{T}$ corresponds to either a revoked user or a potential user who has not yet registered. In other words, only active users who are allowed to sign in the new epoch τ_{new} have their non-zero public keys, denoted by $\{\mathbf{p}_j\}_j$, accumulated in the root $\mathbf{u}_{\tau_{\mathrm{new}}}$ of the updated tree.
   For each j, let $w_j \in \{0,1\}^{\ell} \times (\{0,1\}^{nk})^{\ell}$ be the witness for the fact that $\mathbf{p}_j$ is accumulated in $\mathbf{u}_{\tau_{\mathrm{new}}}$. Then GM publishes the group information of the new epoch as:

$$\mathsf{info}_{\tau_{\mathrm{new}}} = \big(\mathbf{u}_{\tau_{\mathrm{new}}}, \{w_j\}_j\big).$$

We remark that the info_{τ} outputted at each epoch by GM is technically not part of the verification key. Indeed, as we will describe below, in order to verify signatures bound to epoch τ, the verifiers only need to download the first component $\mathbf{u}_{\tau}$ of size $\widetilde{\mathcal{O}}(\lambda)$ bits. Meanwhile, each active signer only has to download the respective witness of size $\widetilde{\mathcal{O}}(\lambda) \cdot \ell$.

$\mathsf{Sign}(\mathsf{gpk}, \mathsf{gsk}[j], \mathsf{info}_{\tau}, M)$**.** To sign message M using the group information info_{τ} at epoch τ, the user possessing $\mathsf{gsk}[j] = (\mathsf{bin}(j), \mathbf{p}, \mathbf{x})$ first checks if info_{τ} includes a witness containing $\mathsf{bin}(j)$. If this is not the case, return $\perp$. Otherwise, the user downloads $\mathbf{u}_{\tau}$ and the witness of the form $\big(\mathsf{bin}(j), (\mathbf{w}_{\ell}, \ldots, \mathbf{w}_1)\big)$ from info_{τ}, and proceeds as follows.

1. Encrypt vector $\mathsf{bin}(j) \in \{0,1\}^{\ell}$ twice using Regev's encryption scheme. Namely, for each $i \in \{1,2\}$, sample $\mathbf{r}_i \xleftarrow{\$} \{0,1\}^{m_E}$ and compute

$$\begin{aligned}\mathbf{c}_i &= (\mathbf{c}_{i,1}, \mathbf{c}_{i,2}) \\ &= \Big(\mathbf{B} \cdot \mathbf{r}_i \bmod q,\ \mathbf{P}_i \cdot \mathbf{r}_i + \lceil \frac{q}{2} \rfloor \cdot \mathsf{bin}(j) \bmod q\Big) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^{\ell}.\end{aligned}$$

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

$$\zeta = (\mathbf{x}, \mathbf{p}, \mathsf{bin}(j), \mathbf{w}_{\ell}, \ldots, \mathbf{w}_1, \mathbf{r}_1, \mathbf{r}_2) \tag{2}$$

such that:
(i) $\mathsf{TVerify}_{\mathbf{A}}\big(\mathbf{u}_{\tau}, \mathbf{p}, \big(\mathsf{bin}(j), (\mathbf{w}_{\ell}, \ldots, \mathbf{w}_1)\big)\big) = 1$ and $\mathbf{A} \cdot \mathbf{x} = \mathbf{G} \cdot \mathbf{p} \bmod q$;
(ii) $\mathbf{c}_1$ and $\mathbf{c}_2$ are both correct encryptions of $\mathsf{bin}(j)$ with randomness $\mathbf{r}_1$ and $\mathbf{r}_2$, respectively;
(iii) $\mathbf{p} \neq \mathbf{0}^{nk}$.
Note that statements (i) and (ii) were covered by the LLNW protocol [29]. Meanwhile, statement (iii) is handled using the technique described at the beginning of this Section. We thus obtain a Stern-like interactive zero-knowledge argument system which is a slight modification of the one from [29]. The detailed description of the protocol is presented in Appendix A.
The protocol is repeated $\kappa = \omega(\log \lambda)$ times to achieve negligible soundness error and made non-interactive via the Fiat-Shamir heuristic as a triple $\Pi_{\mathsf{gs}} = (\{\mathrm{CMT}_i\}_{i=1}^{\kappa}, \mathrm{CH}, \{\mathrm{RSP}\}_{i=1}^{\kappa})$, where

$$\mathrm{CH} = \mathcal{H}_{\mathsf{FS}}\big(M, (\{\mathrm{CMT}_i\}_{i=1}^{\kappa}, \mathbf{A}, \mathbf{u}_{\tau}, \mathbf{B}, \mathbf{P}_1, \mathbf{P}_2, \mathbf{c}_1, \mathbf{c}_2\big) \in \{1,2,3\}^{\kappa}.$$

3. Output the group signature

$$\Sigma = (\Pi_{\mathsf{gs}}, \mathbf{c}_1, \mathbf{c}_2). \tag{3}$$

$\mathsf{Verify}(\mathsf{gpk}, \mathsf{info}_{\tau}, M, \Sigma)$**.** This algorithm proceeds as follows:

1. Download $\mathbf{u}_{\tau} \in \{0,1\}^{nk}$ from info_{τ}.
2. Parse Σ as $\Sigma = \big(\{\mathrm{CMT}_i\}_{i=1}^{\kappa}, (Ch_1, \ldots, Ch_{\kappa}), \{\mathrm{RSP}\}_{i=1}^{\kappa}, \mathbf{c}_1, \mathbf{c}_2\big)$.
If $(Ch_1, \ldots, Ch_{\kappa}) \neq \mathcal{H}_{\mathsf{FS}}\big(M, (\{\mathrm{CMT}_i\}_{i=1}^{\kappa}, \mathbf{A}, \mathbf{u}_{\tau}, \mathbf{B}, \mathbf{P}_1, \mathbf{P}_2, \mathbf{c}_1, \mathbf{c}_2\big)$, then return 0.
3. For each $i = 1$ to κ, run the verification phase of the protocol in Appendix A to check the validity of RSP_i with respect to CMT_i and Ch_i. If any of the conditions does not hold, then return 0.
4. Return 1.

$\mathsf{Trace}(\mathsf{gpk}, \mathsf{tsk}, \mathsf{info}_{\tau}, \mathbf{reg}, M, \Sigma)$**.** This algorithm parses tsk as $(\mathbf{S}_1, \mathbf{E}_1)$, parses Σ as in (3), and performs the following steps.

1. Use $\mathbf{S}_1$ to decrypt $\mathbf{c}_1 = (\mathbf{c}_{1,1}, \mathbf{c}_{1,2})$ to obtain a string $\mathbf{b}' \in \{0,1\}^{\ell}$ (i.e., by computing $\big\lfloor (\mathbf{c}_{1,2} - \mathbf{S}_1^{\top} \cdot \mathbf{c}_{1,1})/(q/2) \big\rceil$.
2. If info_{τ} does not include a witness containing $\mathbf{b}'$, then return $\perp$.

3. Let $j' \in [0, N-1]$ be the integer having binary representation b'. If the record $\mathbf{reg}[j'][1]$ in table $\mathbf{reg}$ is $\mathbf{0}^{nk}$, then return $\perp$.
4. Generate a $\mathsf{NIZKAoK}$ Π_{trace} to demonstrate the possession of $\mathbf{S}_1 \in \mathbb{Z}^{n\times\ell}$, $\mathbf{E}_1 \in \mathbb{Z}^{\ell\times m_E}$, and $\mathbf{y} \in \mathbb{Z}^{\ell}$, such that:

$$\begin{cases} \|\mathbf{S}_1\|_\infty \leq \beta;\ \|\mathbf{E}_1\|_\infty \leq \beta;\ \|\mathbf{y}\|_\infty \leq \lceil q/5 \rceil; \\ \mathbf{S}_1^\top \cdot \mathbf{B} + \mathbf{E}_1 = \mathbf{P}_1 \bmod q; \\ \mathbf{c}_{1,2} - \mathbf{S}_1^\top \cdot \mathbf{c}_{1,1} = \mathbf{y} + \lfloor q/2 \rfloor \cdot \mathbf{b}' \bmod q. \end{cases} \tag{4}$$

   As the statement involves modular linear equations with bounded-norm secrets, we can obtain a statistical zero-knowledge argument by employing the Stern-like interactive protocol from [27]. The protocol is repeated $\kappa = \omega(\log \lambda)$ times to achieve negligible soundness error and made non-interactive via the Fiat-Shamir heuristic as a triple $\Pi_{\mathsf{trace}} = (\{\mathrm{CMT}_i\}_{i=1}^{\kappa}, \mathrm{CH}, \{\mathrm{RSP}\}_{i=1}^{\kappa})$, where

$$\mathrm{CH} = \mathcal{H}_{\mathsf{FS}}\big((\{\mathrm{CMT}_i\}_{i=1}^{\kappa}, \mathsf{gpk}, \mathsf{info}_\tau, M, \Sigma, \mathbf{b}'\big) \in \{1,2,3\}^{\kappa}. \tag{5}$$

5. Set $\mathsf{uid} = \mathbf{b}'$ and output $(\mathsf{uid}, \Pi_{\mathsf{trace}})$.

$\mathsf{Judge}(\mathsf{gpk}, \mathsf{uid}, \mathsf{info}_\tau, \Pi_{\mathsf{trace}}, M, \Sigma)$**.** This algorithm consists of verifying the argument Π_{trace} w.r.t. common input $(\mathsf{gpk}, \mathsf{info}_\tau, M, \Sigma, \mathsf{uid})$, in a similar manner as in algorithm Verify.
If Π_{trace} does not verify, return 0. Otherwise, return 1.

4.2 Analysis of the Scheme

Efficiency. We first analyze the efficiency of the scheme described in Section 4.1, with respect to security parameter λ and parameter $\ell = \log N$.

- The public key gpk contains several matrices, and has bit-size $\widetilde{\mathcal{O}}(\lambda^2 + \lambda \cdot \ell)$.
- For each $j \in [0, N-1]$, the signing key $\mathsf{gsk}[j]$ has bit-size $\ell + nk + m = \widetilde{\mathcal{O}}(\lambda) + \ell$.
- At each epoch, the signature verifiers downloads $nk = \widetilde{\mathcal{O}}(\lambda)$ bits, while each active signer downloads $\widetilde{\mathcal{O}}(\lambda \cdot \ell)$ bits.
- The size of signature Σ is dominated by that of the Stern-like $\mathsf{NIZKAoK}$ Π_{gs}, which is $\mathcal{O}(|\zeta| \cdot \log q) \cdot \omega(\log \lambda)$, where $|\zeta|$ denotes the bit-size of the witness-tuple ζ in (2). Overall, Σ has bit-size $\widetilde{\mathcal{O}}(\lambda \cdot \ell)$.
- The Stern-like $\mathsf{NIZKAoK}$ Π_{trace} has bit-size $\widetilde{\mathcal{O}}(\ell^2 + \lambda \cdot \ell)$.

Correctness. We now demonstrate that the scheme is correct with overwhelming probability, based on the perfect completeness of Stern-like protocols, and the correctness of Regev's encryption scheme.

First, note that a signature $\Sigma = (\Pi_{\mathsf{gs}}, \mathbf{c}_1, \mathbf{c}_2)$ generated by an active and honest user j is always accepted by algorithm Verify. Indeed, such a user can always compute a tuple $\zeta = (\mathbf{x}, \mathbf{p}, \mathsf{bin}(j), \mathbf{w}_\ell, \ldots, \mathbf{w}_1, \mathbf{r}_1, \mathbf{r}_2)$ satisfying conditions (i),(ii)

and (iii) in the Sign algorithm. The completeness of the underlying argument system then guarantees that Σ is always accepted by algorithm Verify.

Next, we show that algorithm Trace outputs $\mathsf{bin}(j)$ with overwhelming probability, and produces a proof Π_{trace} accepted by algorithm Judge. Observe that, the decryption algorithm essentially computes

$$\mathbf{e} = \mathbf{c}_{1,2} - \mathbf{S}_1^T \mathbf{c}_{1,1} = \mathbf{E}_1 \cdot \mathbf{r}_1 + \lfloor q/2 \rceil \cdot \mathsf{bin}(j) \mod q,$$

and sets the j-th bit of $\mathbf{b}'$ to be 0 if j-th entry of $\mathbf{e}$ is closer to 0 than to $\lfloor q/2 \rceil$ and 1 otherwise. Note that our parameters are set so that $\|\mathbf{E}_1 \cdot \mathbf{r}_1\|_\infty < q/5$, for $\mathbf{E}_1 \hookleftarrow \chi^{\ell \times m_E}$ and $\mathbf{r}_1 \xleftarrow{\$} \{0,1\}^{m_E}$. This ensures that $\mathbf{b}' = \mathsf{bin}(j)$ with overwhelming probability.

Further, as the user is active, info_τ must contain $w = (\mathsf{bin}(j), \mathbf{w}_\ell, \ldots, \mathbf{w}_1)$ and $\mathbf{reg}[j][1]$ in table $\mathbf{reg}$ is not $\mathbf{0}^{nk}$. Therefore, algorithm Trace will move to the 4-th step, where it can always obtain the tuple $(\mathbf{S}_1, \mathbf{E}_1, \mathbf{y})$ satisfying the conditions (4). By the completeness of the argument system, Π_{trace} will be accepted by the algorithm Judge.

Security. In Theorem 1, we prove that our scheme satisfies the security requirements of the Bootle et al.'s model [7].

Theorem 1. *Assume that the Stern-like argument systems used in Section 4.1 are simulation-sound. Then, in the random oracle model, the given fully dynamic group signature satisfies the anonymity, traceability, non-frameability and tracing soundness requirements under the* $\mathsf{LWE}_{n,q,\chi}$ *and* $\mathsf{SIS}^{\infty}_{n,m,q,1}$ *assumptions.*

In the random oracle model, the proof of Theorem 1 relies on the following facts:

1. The Stern-like zero-knowledge argument systems being used are simulation-sound;
2. The underlying encryption scheme, which is obtained from Regev cryptosystem [43] via the Naor-Yung transformation [39], is IND-CCA2 secure;
3. The Merkle tree accumulator we employ is secure in the sense of Definition 3;
4. For a properly generated key-pair $(\mathbf{x}, \mathbf{p})$, it is infeasible to find $\mathbf{x}' \in \{0,1\}^m$ such that $\mathbf{x}' \neq \mathbf{x}$ and $\mathsf{bin}(\mathbf{A} \cdot \mathbf{x}' \bmod q) = \mathbf{p}$.

Details of the proof of Theorem 1 are deferred to Appendix C.

Acknowledgements. The authors would like to thank Benoît Libert and the anonymous reviewers for helpful comments and discussions. The research was supported by Research Grant TL-9014101684-01 and the Singapore Ministry of Education under Research Grant MOE2013-T2-1-041.

References

1. M. Ajtai. Generating Hard Instances of Lattice Problems (Extended Abstract). In *STOC 1996*, pages 99–108. ACM, 1996.

2. G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A Practical and Provably Secure Coalition-Resistant Group Signature Scheme. In *CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 255–270. Springer, 2000.
3. N. Baric and B. Pfitzmann. Collision-Free Accumulators and Fail-Stop Signature Schemes Without Trees. In *EUROCRYPT 1997*, volume 1233 of *Lecture Notes in Computer Science*, pages 480–494. Springer, 1997.
4. M. Bellare, D. Micciancio, and B. Warinschi. Foundations of Group Signatures: Formal Definitions, Simplified Requirements, and a Construction Based on General Assumptions. In *EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 614–629. Springer, 2003.
5. M. Bellare, H. Shi, and C. Zhang. Foundations of Group Signatures: The Case of Dynamic Groups. In *CT-RSA 2005*, volume 3376 of *Lecture Notes in Computer Science*, pages 136–153. Springer, 2005.
6. D. Boneh and H. Shacham. Group Signatures with Verifier-local Revocation. In *ACM CCCS 2004*, pages 168–177. ACM, 2004.
7. J. Bootle, A. Cerulli, P. Chaidos, E. Ghadafi, and J. Groth. Foundations of Fully Dynamic Group Signatures. In *ACNS 2016*, volume 9696 of *Lecture Notes in Computer Science*, pages 117–136. Springer, 2016. Full version: `https://eprint.iacr.org/2016/368.pdf`.
8. J. Bootle, A. Cerulli, P. Chaidos, E. Ghadafi, J. Groth, and C. Petit. Short Accountable Ring Signatures Based on DDH. In *ESORICS 2015*, volume 9326 of *Lecture Notes in Computer Science*, pages 243–265. Springer, 2015.
9. X. Boyen. Lattice Mixing and Vanishing Trapdoors: A Framework for Fully Secure Short Signatures and More. In *PKC 2010*, volume 6056 of *Lecture Notes in Computer Science*, pages 499–517. Springer, 2010.
10. E. Bresson and J. Stern. Efficient Revocation in Group Signatures. In *PKC 2001*, volume 1992 of *Lecture Notes in Computer Science*, pages 190–206. Springer, 2001.
11. E. Brickell, D. Pointcheval, S. Vaudenay, and M. Yung. Design Validations for Discrete Logarithm Based Signature Schemes. In *PKC 2000*, pages 276–292. Springer, 2000.
12. J. Camenisch and A. Lysyanskaya. Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials. In *CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 61–76. Springer, 2002.
13. J. Camenisch, G. Neven, and M. Rückert. Fully Anonymous Attribute Tokens from Lattices. In *SCN 2012*, volume 7485 of *Lecture Notes in Computer Science*, pages 57–75. Springer, 2012.
14. D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert. Bonsai Trees, or How to Delegate a Lattice Basis. In *EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 523–552. Springer, 2010.
15. D. Chaum and E. van Heyst. Group Signatures. In *EUROCRYPT 1991*, volume 547 of *Lecture Notes in Computer Science*, pages 257–265. Springer, 1991.
16. S. Cheng, K. Nguyen, and H. Wang. Policy-Based Signature Scheme from Lattices. *Des. Codes Cryptography*, 81(1):43–74, 2016.
17. C. Delerablée and D. Pointcheval. Dynamic Fully Anonymous Short Group Signatures. In *VIETCRYPT 2006*, volume 4341 of *Lecture Notes in Computer Science*, pages 193–210. Springer, 2006.
18. A. Fiat and A. Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *CRYPTO 1986*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1987.
19. C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for Hard Lattices and New Cryptographic Constructions. In *STOC 2008*, pages 197–206. ACM, 2008.

20. S. D. Gordon, J. Katz, and V. Vaikuntanathan. A Group Signature Scheme from Lattice Assumptions. In *ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 395–412. Springer, 2010.
21. J. Groth. Fully Anonymous Group Signatures without Random Oracles. In *ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 164–180. Springer, 2007.
22. A. Kawachi, K. Tanaka, and K. Xagawa. Multi-bit Cryptosystems Based on Lattice Problems. In *PKC 2007*, volume 4450 of *Lecture Notes in Computer Science*, pages 315–329. Springer, 2007.
23. A. Kawachi, K. Tanaka, and K. Xagawa. Concurrently Secure Identification Schemes Based on the Worst-Case Hardness of Lattice Problems. In *ASIACRYPT 2008*, volume 5350 of *Lecture Notes in Computer Science*, pages 372–389. Springer, 2008.
24. A. Kiayias and M. Yung. Secure Scalable Group Signature with Dynamic Joins and Separable authorities. *Int. Journal of Security and Networks*, 1(1):24–45, 2006.
25. F. Laguillaumie, A. Langlois, B. Libert, and D. Stehlé. Lattice-Based Group Signatures with Logarithmic Signature Size. In *ASIACRYPT 2013*, volume 8270 of *Lecture Notes in Computer Science*, pages 41–61. Springer, 2013.
26. A. Langlois, S. Ling, K. Nguyen, and H. Wang. Lattice-Based Group Signature Scheme with Verifier-Local Revocation. In *PKC 2014*, volume 8383 of *Lecture Notes in Computer Science*, pages 345–361. Springer, 2014.
27. B. Libert, S. Ling, F. Mouhartem, K. Nguyen, and H. Wang. Signature Schemes with Efficient Protocols and Dynamic Group Signatures from Lattice Assumptions. In *ASIACRYPT 2016*, volume 10032 of *Lecture Notes in Computer Science*, pages 373–403. Springer, 2016.
28. B. Libert, S. Ling, F. Mouhartem, K. Nguyen, and H. Wang. Zero-Knowledge Arguments for Matrix-Vector Relations and Lattice-Based Group Encryption. In *ASIACRYPT 2016*, volume 10032 of *Lecture Notes in Computer Science*, pages 101–131, 2016.
29. B. Libert, S. Ling, K. Nguyen, and H. Wang. Zero-Knowledge Arguments for Lattice-Based Accumulators: Logarithmic-Size Ring Signatures and Group Signatures Without Trapdoors. In *EUROCRYPT 2016*, volume 9666 of *Lecture Notes in Computer Science*, pages 1–31. Springer, 2016.
30. B. Libert, F. Mouhartem, and K. Nguyen. A Lattice-Based Group Signature Scheme with Message-Dependent Opening. In *ACNS 2016*, volume 9696 of *Lecture Notes in Computer Science*, pages 137–155. Springer, 2016.
31. B. Libert, T. Peters, and M. Yung. Group Signatures with Almost-for-Free Revocation. In *CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 571–589. Springer, 2012.
32. B. Libert, T. Peters, and M. Yung. Scalable Group Signatures with Revocation. In *EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 609–627. Springer, 2012.
33. B. Libert, T. Peters, and M. Yung. Short Group Signatures via Structure-Preserving Signatures: Standard Model Security from Simple Assumptions. In *CRYPTO 2015*, volume 9216 of *Lecture Notes in Computer Science*. Springer, 2015.
34. S. Ling, K. Nguyen, D. Stehlé, and H. Wang. Improved Zero-Knowledge Proofs of Knowledge for the ISIS Problem, and Applications. In *PKC 2013*, volume 7778, pages 107–124. Springer, 2013.

35. S. Ling, K. Nguyen, and H. Wang. Group Signatures from Lattices: Simpler, Tighter, Shorter, Ring-Based. In *PKC 2015*, volume 9020 of *Lecture Notes in Computer Science*, pages 427–449. Springer, 2015.
36. D. Micciancio and P. Mol. Pseudorandom Knapsacks and the Sample Complexity of LWE Search-to-Decision Reductions. In *CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 465–484. Springer, 2011.
37. D. Micciancio and C. Peikert. Hardness of SIS and LWE with Small Parameters. In *CRYPTO 2013*, volume 8042 of *Lecture Notes in Computer Science*, pages 21–39. Springer, 2013.
38. T. Nakanishi, H. Fujii, Y. Hira, and N. Funabiki. Revocable Group Signature Schemes with Constant Costs for Signing and Verifying. In *PKC 2009*, volume 5443 of *Lecture Notes in Computer Science*, pages 463–480. Springer, 2009.
39. M. Naor and M. Yung. Public-Key Cryptosystems Provably Secure against Chosen Ciphertext Attacks. In *STOC 1990*, pages 427–437. ACM, 1990.
40. L. Nguyen. Accumulators from Bilinear Pairings and Applications. In *CT-RSA 2005*, volume 3376 of *Lecture Notes in Computer Science*, pages 275–292. Springer, 2005.
41. L. Nguyen and R. Safavi-Naini. Efficient and Provably Secure Trapdoor-Free Group Signature Schemes from Bilinear Pairings. In *ASIACRYPT 2004*, volume 3329 of *Lecture Notes in Computer Science*, pages 372–386. Springer, 2004.
42. P. Q. Nguyen, J. Zhang, and Z. Zhang. Simpler Efficient Group Signatures from Lattices. In *PKC 2015*, volume 9020 of *Lecture Notes in Computer Science*, pages 401–426. Springer, 2015.
43. O. Regev. On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. In *STOC 2005*, pages 84–93. ACM, 2005.
44. Y. Sakai, K. Emura, G. Hanaoka, Y. Kawai, T. Matsuda, and K. Omote. Group Signatures with Message-Dependent Opening. In *Pairing 2012*, volume 7708 of *Lecture Notes in Computer Science*, pages 270–294. Springer, 2012.
45. Y. Sakai, J. Schuldt, K. Emura, G. Hanaoka, and K. Ohta. On the Security of Dynamic Group Signatures: Preventing Signature Hijacking. In *PKC 2012*, volume 7293 of *Lecture Notes in Computer Science*, pages 715–732. Springer, 2012.
46. J. Stern. A New Paradigm for Public Key Identification. *IEEE Transactions on Information Theory*, 42(6):1757–1768, 1996.
47. G. Tsudik and S. Xu. Accumulating Composites and Improved Group Signing. In *ASIACRYPT 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 269–286. Springer, 2003.

A Details of the Main Zero-Knowledge Argument System

Our protocol is a modification of the one from [29], in which we additionally prove that $\mathbf{p} \neq \mathbf{0}^{nk}$ using the technique discussed at the beginning of Section 4. We will employ a new strategy for Stern-like protocols, suggested in [27], which consists of unifying all the equations to be proved into just one equation of the form $\mathbf{M} \cdot \mathbf{z} = \mathbf{u} \bmod q$, for some public matrix $\mathbf{M}$ and vector $\mathbf{u}$ over $\mathbb{Z}_q$. This strategy will help us to deliver a simpler presentation than in [29].

Let us first examine the equations associated with the execution of algorithm $\mathsf{TVerify}_{\mathbf{A}}\big(\mathbf{u}_\tau, \mathbf{p}, ((j_1, \ldots, j_\ell), (\mathbf{w}_\ell, \ldots, \mathbf{w}_1))\big)$. This algorithm computes the tree

path $\mathbf{v}_\ell, \mathbf{v}_{\ell-1}, \ldots, \mathbf{v}_1, \mathbf{v}_0 \in \{0,1\}^{nk}$ as follows: $\mathbf{v}_\ell = \mathbf{p}$ and

$$\forall i \in \{\ell-1, \ldots, 1, 0\} : \mathbf{v}_i = \begin{cases} h_{\mathbf{A}}(\mathbf{v}_{i+1}, \mathbf{w}_{i+1}), \text{ if } j_{i+1} = 0; \\ h_{\mathbf{A}}(\mathbf{w}_{i+1}, \mathbf{v}_{i+1}), \text{ if } j_{i+1} = 1. \end{cases} \tag{6}$$

The algorithm outputs 1 if $\mathbf{v}_0 = \mathbf{u}_\tau$. Observe that relation (6) can be equivalently rewritten as: $\forall i \in \{\ell-1, \ldots, 1, 0\}$,

$$\begin{aligned} & \bar{j}_{i+1} \cdot h_{\mathbf{A}}(\mathbf{v}_{i+1}, \mathbf{w}_{i+1}) + j_{i+1} \cdot h_{\mathbf{A}}(\mathbf{w}_{i+1}, \mathbf{v}_{i+1}) = \mathbf{v}_i \\ \Leftrightarrow\ & \bar{j}_{i+1} \cdot \big(\mathbf{A}_0 \cdot \mathbf{v}_{i+1} + \mathbf{A}_1 \cdot \mathbf{w}_{i+1}\big) + j_{i+1} \cdot \big(\mathbf{A}_0 \cdot \mathbf{w}_{i+1} + \mathbf{A}_1 \cdot \mathbf{v}_{i+1}\big) = \mathbf{G} \cdot \mathbf{v}_i \bmod q \\ \Leftrightarrow\ & \mathbf{A} \cdot \begin{pmatrix} \bar{j}_{i+1} \cdot \mathbf{v}_{i+1} \\ j_{i+1} \cdot \mathbf{v}_{i+1} \end{pmatrix} + \mathbf{A} \cdot \begin{pmatrix} j_{i+1} \cdot \mathbf{w}_{i+1} \\ \bar{j}_{i+1} \cdot \mathbf{w}_{i+1} \end{pmatrix} = \mathbf{G} \cdot \mathbf{v}_i \bmod q \\ \Leftrightarrow\ & \mathbf{A} \cdot \mathsf{ext}(j_{i+1}, \mathbf{v}_{i+1}) + \mathbf{A} \cdot \mathsf{ext}(\bar{j}_{i+1}, \mathbf{w}_{i+1}) = \mathbf{G} \cdot \mathbf{v}_i \bmod q. \end{aligned}$$

In the above, we use the notation $\mathsf{ext}(b, \mathbf{v})$, for bit b and vector $\mathbf{v}$, to denote the vector $\begin{pmatrix} \bar{b} \cdot \mathbf{v} \\ b \cdot \mathbf{v} \end{pmatrix}$. Later on, we will also use the notation $\mathsf{ext}_2(b)$, for bit b, to denote the 2-dimensional vector $(\bar{b}, b)^\top$.

Given the above discussion, our protocol can be summarized as follows.

Public input: $\mathbf{A}$, $\mathbf{G}$, $\mathbf{u}_\tau$, $\mathbf{B}$, $\mathbf{P}_1$, $\mathbf{P}_2$, $\mathbf{c}_1 = (\mathbf{c}_{1,1}, \mathbf{c}_{1,2})$, $\mathbf{c}_2 = (\mathbf{c}_{2,1}, \mathbf{c}_{2,2})$.

Prover's goal: Proving knowledge of secret $\mathbf{x} \in \{0,1\}^m$, $\mathbf{0} \neq \mathbf{p} \in \{0,1\}^{nk}$, $j_1, \ldots, j_\ell \in \{0,1\}$, $\mathbf{v}_1, \ldots, \mathbf{v}_{\ell-1}, \mathbf{w}_1, \ldots, \mathbf{w}_\ell \in \{0,1\}^{nk}$, $\mathbf{r}_1, \mathbf{r}_2 \in \{0,1\}^{m_E}$, such that the following equations hold:

$$\begin{cases} \mathbf{A} \cdot \mathsf{ext}(j_1, \mathbf{v}_1) + \mathbf{A} \cdot \mathsf{ext}(\bar{j}_1, \mathbf{w}_1) = \mathbf{G} \cdot \mathbf{u}_\tau \bmod q; \\ \mathbf{A} \cdot \mathsf{ext}(j_2, \mathbf{v}_2) + \mathbf{A} \cdot \mathsf{ext}(\bar{j}_2, \mathbf{w}_2) - \mathbf{G} \cdot \mathbf{v}_1 = \mathbf{0} \bmod q; \\ \qquad \ldots \quad \ldots \quad \ldots \\ \mathbf{A} \cdot \mathsf{ext}(j_\ell, \mathbf{p}) + \mathbf{A} \cdot \mathsf{ext}(\bar{j}_\ell, \mathbf{w}_\ell) - \mathbf{G} \cdot \mathbf{v}_{\ell-1} = \mathbf{0} \bmod q; \\ \mathbf{A} \cdot \mathbf{x} - \mathbf{G} \cdot \mathbf{p} = \mathbf{0} \bmod q; \\ \mathbf{B} \cdot \mathbf{r}_b = \mathbf{c}_{b,1} \bmod q, \text{ for } b \in \{1,2\}; \\ \mathbf{P}_b \cdot \mathbf{r}_b + \left\lfloor \frac{q}{2} \right\rceil \cdot (j_1, \ldots, j_\ell)^\top = \mathbf{c}_{b,2} \bmod q, \text{ for } b \in \{1,2\}. \end{cases} \tag{7}$$

Now, we employ extending-then-permuting techniques for Stern-like protocols [46,34,29,27] to handle the secret objects, as follows. For any positive integer i, we let $\mathcal{S}_i$ denote the set of all permutations of i elements.

- To prove that $\mathbf{p} \in \{0,1\}^{nk}$ and $\mathbf{p} \neq 0^{nk}$, we append $nk-1$ "dummy" entries to $\mathbf{p}$ to get $\mathbf{p}^* \in \mathsf{B}_{nk}^{2nk-1}$. Note that, for any $\pi_p \in \mathcal{S}_{2nk-1}$, we have:

$$\mathbf{p}^* \in \mathsf{B}_{nk}^{2nk-1} \Leftrightarrow \pi_p(\mathbf{p}^*) \in \mathsf{B}_{nk}^{2nk-1}.$$

- To prove that $\mathbf{x} \in \{0,1\}^m$, we append m "dummy" entries to $\mathbf{x}$ to get $\mathbf{x}^* \in \mathsf{B}_m^{2m}$. Note that, for any $\pi_x \in \mathcal{S}_{2m}$, we have: $\mathbf{x}^* \in \mathsf{B}_m^{2m} \Leftrightarrow \pi_x(\mathbf{x}^*) \in \mathsf{B}_m^{2m}$.

– Similarly, we extend $\mathbf{v}_1, \ldots, \mathbf{v}_{\ell-1}, \mathbf{w}_1, \ldots, \mathbf{w}_\ell$ to $\mathbf{v}_1^*, \ldots, \mathbf{v}_{\ell-1}^*, \mathbf{w}_1^*, \ldots, \mathbf{w}_\ell^* \in \mathsf{B}_{nk}^{2nk}$, respectively. We also extend $\mathbf{r}_1, \mathbf{r}_2$ to $\mathbf{r}_1^*, \mathbf{r}_2^* \in \mathsf{B}_{m_E}^{2m_E}$, respectively.
– For each $i = 1, \ldots, \ell$, to prove that the bit $j_i \in \{0,1\}$ is involved in both encryption layer and Merkle tree layer, we extend it to $\mathbf{j}_i = \mathsf{ext}_2(j_i)$. Then we use the following permuting technique.
  For bit $b \in \{0,1\}$, we let T_b be the permutation that transforms vector $\mathbf{t} = (t_0, t_1)^\top \in \mathbb{Z}^2$ into vector $T_b(\mathbf{t}) = (t_b, t_{\bar{b}})$. Note that, for any $b_i \in \{0,1\}$, we have: $\mathbf{j}_i = \mathsf{ext}_2(j_i) \Leftrightarrow T_{b_i}(\mathbf{j}_i) = \mathsf{ext}_2(j_i \oplus b_i)$, where $\oplus$ denotes the addition modulo 2. Then, to prove knowledge of $j_i \in \{0,1\}$, we instead prove knowledge of $\mathbf{j}_i = \mathsf{ext}_2(j_i)$, by sampling $b_i \xleftarrow{\$} \{0,1\}$ and showing the verifier that the right-hand side of the equivalence holds. Here, the crucial point is that b_i acts as a "one-time pad" that perfectly hides j_i. Furthermore, to prove that j_i appears in a different layer of the system, we will set up a similar equivalence in that layer, and use the same b_i at both places.
– Now, we have to handle the extended vectors appearing in the context after the above extensions. They are $\widehat{\mathbf{v}}_1 = \mathsf{ext}(j_1, \mathbf{v}_1^*), \ldots, \widehat{\mathbf{v}}_{\ell-1} = \mathsf{ext}(j_{\ell-1}, \mathbf{v}_{\ell-1}^*) \in \{0,1\}^{4nk}$ and $\widehat{\mathbf{w}}_1 = \mathsf{ext}(\bar{j}_1, \mathbf{w}_1^*), \ldots, \widehat{\mathbf{w}}_\ell = \mathsf{ext}(\bar{j}_\ell, \mathbf{w}_\ell^*) \in \{0,1\}^{4nk}$, as well as $\widehat{\mathbf{p}} = \mathsf{ext}(j_\ell, \mathbf{p}^*) \in \{0,1\}^{4nk-2}$. To prove that these vector are well-formed, we will use the following permuting techniques.
  Let $r \in \{2nk, 2nk-1\}$. For $b \in \{0,1\}$ and $\pi \in \mathcal{S}_r$, we define the permutation $F_{b,\pi}$ that transforms $\mathbf{t} = \begin{pmatrix} \mathbf{t}_0 \\ \mathbf{t}_1 \end{pmatrix} \in \mathbb{Z}^{2r}$ consisting of 2 blocks of size r into $F_{b,\pi}(\mathbf{t}) = \begin{pmatrix} \pi(\mathbf{t}_b) \\ \pi(\mathbf{t}_{\bar{b}}) \end{pmatrix}$. Namely, $F_{b,\pi}$ first rearranges the blocks of $\mathbf{t}$ according to b (it keeps the arrangement of blocks if $b = 0$, or swaps them if $b = 1$), then it permutes each block according to π. Note that, we have the following, for all $b_1, \ldots, b_\ell \in \{0,1\}$, $\phi_{v,1}, \ldots, \phi_{v,\ell-1}, \phi_{w,1}, \ldots, \phi_{w,\ell} \in \mathcal{S}_{2nk}$, and $\pi_p \in \mathcal{S}_{2nk-1}$:

$$\begin{cases} \forall i \in [\ell-1] : \widehat{\mathbf{v}}_i = \mathsf{ext}(j_i, \mathbf{v}_i^*) \iff F_{b_i,\phi_{v,i}}(\widehat{\mathbf{v}}_i) = \mathsf{ext}(j_i \oplus b_i, \phi_{v,i}(\mathbf{v}_i^*)); \\ \forall i \in [\ell] : \widehat{\mathbf{w}}_i = \mathsf{ext}(\bar{j}_i, \mathbf{w}_i^*) \iff F_{b_i,\phi_{w,i}}(\widehat{\mathbf{w}}_i) = \mathsf{ext}(\overline{j_i \oplus b_i}, \phi_{w,i}(\mathbf{w}_i^*)); \\ \widehat{\mathbf{p}} = \mathsf{ext}(j_\ell, \mathbf{p}^*) \iff F_{b_\ell,\pi_p}(\widehat{\mathbf{p}}) = \mathsf{ext}(j_\ell \oplus b_\ell, \pi_p(\mathbf{p}^*)). \end{cases}$$

  Combining these permuting techniques with the ones described earlier, we can prove that the considered extended vectors are well-formed, with respect to the secret $j_1, \ldots, j_\ell, \mathbf{v}_1^*, \ldots, \mathbf{v}_{\ell-1}^*, \mathbf{w}_1^*, \ldots, \mathbf{w}_\ell^*, \mathbf{p}^*$.

Given the above transformations, we now unify all the secret objects into vector $\mathbf{z} \in \{0,1\}^D$, where $D = 10nk\ell + 2m + 4m_E + 2\ell - 3$, of the following form (we abuse the notation of transposition):

$$\big(\mathbf{v}_1^* \,\|\, \widehat{\mathbf{v}}_1 \,\|\, \widehat{\mathbf{w}}_1 \,\|\, \ldots \,\|\, \mathbf{v}_{\ell-1}^* \,\|\, \widehat{\mathbf{v}}_{\ell-1} \,\|\, \widehat{\mathbf{w}}_{\ell-1} \,\|\, \mathbf{p}^* \,\|\, \widehat{\mathbf{p}} \,\|\, \widehat{\mathbf{w}}_\ell \,\|\, \mathbf{x}^* \,\|\, \mathbf{r}_1^* \,\|\, \mathbf{r}_2^* \,\|\, \mathbf{j}_1 \,\|\, \ldots \,\|\, \mathbf{j}_\ell\big). \quad (8)$$

Then, we observe that, the equations in (7) can be equivalently combined into one equation $\mathbf{M} \cdot \mathbf{z} = \mathbf{u} \bmod q$, where matrix $\mathbf{M}$ and vector $\mathbf{u}$ are built from the public input.

Next, to apply the permuting techniques we have discussed above, let us define VALID as the set of all vectors in $\{0,1\}^D$, that have the form (8), where $\mathbf{x}^* \in \mathsf{B}_m^{2m}$, $\mathbf{r}_1^*, \mathbf{r}_2^* \in \mathsf{B}_{m_E}^{2m_E}$, $\mathbf{p}^* \in \mathsf{B}_{nk}^{2nk-1}$, $\mathbf{v}_1^*, \ldots, \mathbf{v}_{\ell-1}^* \in \mathsf{B}_{nk}^{2nk}$, and there exist $j_1, \ldots, j_\ell \in \{0,1\}$, $\mathbf{w}_1^*, \ldots, \mathbf{w}_\ell^* \in \mathsf{B}_{nk}^{2nk}$, such that:

$$\begin{cases} \widehat{\mathbf{p}} = \mathsf{ext}(j_\ell, \mathbf{p}^*), \quad \forall i \in [\ell] : \mathbf{j}_i = \mathsf{ext}_2(j_i), \\ \forall i \in [\ell-1] : \widehat{\mathbf{v}}_i = \mathsf{ext}(j_i, \mathbf{v}_i^*), \quad \forall i \in [\ell] : \widehat{\mathbf{w}}_i = \mathsf{ext}(\bar{j}_i, \mathbf{w}_i^*). \end{cases}$$

It can be seen that our vector $\mathbf{z}$ belongs to this tailored set VALID. To prove that $\mathbf{z} \in \mathsf{VALID}$ using random permutations, let us determine how to permute the coordinates of $\mathbf{z}$. To this end, we first define the set:

$$\overline{\mathcal{S}} = \{0,1\}^\ell \times \mathcal{S}_{2m} \times \mathcal{S}_{2nk-1} \times (\mathcal{S}_{2m_E})^2 \times (\mathcal{S}_{2nk})^{2\ell-1}.$$

Then, $\forall \eta = \big((b_1, \ldots, b_\ell), \pi_x, \pi_p, (\pi_{r,1}, \pi_{r,2}), (\phi_{v,1}, \ldots, \phi_{v,\ell-1}, \phi_{w,1}, \ldots, \phi_{w,\ell})\big) \in \overline{\mathcal{S}}$, we let Γ_η be the permutation that, when applying to vector $\mathbf{t} \in \mathbb{Z}^D$ whose blocks are as in (8), it transforms those blocks as follows:

- For all $i \in [\ell-1]$: $\quad \mathbf{v}_i^* \mapsto \phi_{v,i}(\mathbf{v}_i^*); \quad \widehat{\mathbf{v}}_i \mapsto F_{b_i,\phi_{v,i}}(\widehat{\mathbf{v}}_i)$.
- For all $i \in [\ell]$: $\quad \widehat{\mathbf{w}}_i \mapsto F_{b_i,\phi_{w,i}}(\widehat{\mathbf{w}}_i); \quad \mathbf{j}_i \mapsto T_{b_i}(\mathbf{j}_i)$.
- $\mathbf{p}^* \mapsto \pi_p(\mathbf{p}^*); \quad \widehat{\mathbf{b}} \mapsto F_{b_\ell,\pi_p}(\widehat{\mathbf{p}})$.
- $\mathbf{x}^* \mapsto \pi_x(\mathbf{x}^*); \quad \mathbf{r}_1^* \mapsto \pi_{r,1}(\mathbf{r}_1^*); \quad \mathbf{r}_2^* \mapsto \pi_{r,2}(\mathbf{r}_2^*)$.

It now can be checked that we have the desired equivalence: For all $\eta \in \overline{\mathcal{S}}$,

$$\mathbf{z} \in \mathsf{VALID} \iff \Gamma_\eta(\mathbf{z}) \in \mathsf{VALID}.$$

At this point, we can now run a simple Stern-like protocol to prove knowledge of $\mathbf{z} \in \mathsf{VALID}$ such that $\mathbf{M} \cdot \mathbf{z} = \mathbf{u} \bmod q$. The common input is the pair $(\mathbf{M}, \mathbf{u})$, while the prover's secret input is $\mathbf{z}$. The interaction between prover $\mathcal{P}$ and verifier $\mathcal{V}$ is described in **Fig. 2**. The protocol employs the string commitment scheme COM from [23] that is statistically hiding and computationally binding if the $\mathsf{SIVP}_{\widetilde{\mathcal{O}}(n)}$ problem is hard.

Theorem 2. *Assume that the* $\mathsf{SIVP}_{\widetilde{\mathcal{O}}(n)}$ *problem is hard. Then the protocol in* **Fig. 2** *is a statistical* ZKAoK *with perfect completeness, soundness error* $2/3$*, and communication cost* $\widetilde{\mathcal{O}}(D \log q)$*. Namely:*

- *There exists a polynomial-time simulator that, on input* $(\mathbf{M}, \mathbf{u})$*, outputs an accepted transcript statistically close to that produced by the real prover.*
- *There exists a polynomial-time knowledge extractor that, on input a commitment* CMT *and* 3 *valid responses* $(\mathrm{RSP}_1, \mathrm{RSP}_2, \mathrm{RSP}_3)$ *to all* 3 *possible values of the challenge* Ch*, outputs* $\mathbf{z}' \in \mathsf{VALID}$ *such that* $\mathbf{M} \cdot \mathbf{z}' = \mathbf{u} \bmod q$.

Given vector $\mathbf{z}'$ outputted by the knowledge extractor, we can compute a tuple ζ' satisfying the conditions described at the beginning of this subsection, simply by "backtracking" the transformation steps we have done on the way. The protocol has communication cost $\widetilde{\mathcal{O}}((10nk\ell + 2m + 4m_E + 2\ell - 3) \log q) = \widetilde{\mathcal{O}}(\lambda \cdot \ell)$ bits.

The proof of Theorem 2 employs standard simulation and extraction techniques for Stern-like protocols [23,34,35,27].

1. **Commitment:** Prover samples $\mathbf{r}_z \xleftarrow{\$} \mathbb{Z}_q^D$, $\eta \xleftarrow{\$} \overline{\mathcal{S}}$ and randomness ρ_1, ρ_2, ρ_3 for COM. Then he sends $\mathrm{CMT} = \big(C_1, C_2, C_3\big)$ to the verifier, where

$$C_1 = \mathsf{COM}(\eta, \mathbf{M} \cdot \mathbf{r}_z; \rho_1), \;\; C_2 = \mathsf{COM}(\mathit{\Gamma}_\eta(\mathbf{r}_z); \rho_2), \;\; C_3 = \mathsf{COM}(\mathit{\Gamma}_\eta(\mathbf{z} + \mathbf{r}_z); \rho_3).$$

2. **Challenge:** The verifier sends a challenge $Ch \xleftarrow{\$} \{1, 2, 3\}$ to the prover.
3. **Response:** Depending on Ch, the prover sends RSP computed as follows:
   - $Ch = 1$: Let $\mathbf{t}_z = \mathit{\Gamma}_\eta(\mathbf{z})$, $\mathbf{t}_r = \mathit{\Gamma}_\eta(\mathbf{r}_z)$, and $\mathrm{RSP} = (\mathbf{t}_z, \mathbf{t}_r, \rho_2, \rho_3)$.
   - $Ch = 2$: Let $\eta_2 = \eta$, $\mathbf{z}_2 = \mathbf{z} + \mathbf{r}_z$, and $\mathrm{RSP} = (\eta_2, \mathbf{z}_2, \rho_1, \rho_3)$.
   - $Ch = 3$: Let $\eta_3 = \eta$, $\mathbf{z}_3 = \mathbf{r}_z$, and $\mathrm{RSP} = (\eta_3, \mathbf{z}_3, \rho_1, \rho_2)$.

Verification: Receiving RSP, the verifier proceeds as follows:

- $Ch = 1$: Check that $\mathbf{t}_z \in \mathsf{VALID}$ and $C_2 = \mathsf{COM}(\mathbf{t}_r; \rho_2)$, $C_3 = \mathsf{COM}(\mathbf{t}_z + \mathbf{t}_r; \rho_3)$.
- $Ch = 2$: Check that $C_1 = \mathsf{COM}(\eta_2, \mathbf{M} \cdot \mathbf{z}_2 - \mathbf{u}; \rho_1)$, $C_3 = \mathsf{COM}(\mathit{\Gamma}_{\eta_2}(\mathbf{z}_2); \rho_3)$.
- $Ch = 3$: Check that $C_1 = \mathsf{COM}(\eta_3, \mathbf{M} \cdot \mathbf{z}_3; \rho_1)$, $\;C_2 = \mathsf{COM}(\mathit{\Gamma}_{\eta_3}(\mathbf{z}_3); \rho_2)$.

In each case, the verifier outputs 1 if and only if all the conditions hold.

Fig. 2: Our Stern-like zero-knowledge argument of knowledge.

Proof. It can be checked that the protocol has perfect completeness: If an honest prover follows the protocol, then he always gets accepted by the verifier. It is also easy to see that the communication cost is bounded by $\widetilde{\mathcal{O}}(D \log q)$.

Zero-Knowledge Property. We construct a PPT simulator SIM interacting with a (possibly dishonest) verifier $\widehat{\mathcal{V}}$, such that, given only the public input, SIM outputs with probability negligibly close to $2/3$ a simulated transcript that is statistically close to the one produced by the honest prover in the real interaction.

The simulator first chooses a random $\overline{Ch} \in \{1, 2, 3\}$ as a prediction of the challenge value that $\widehat{\mathcal{V}}$ will *not* choose.

Case $\overline{Ch} = 1$: Using basic linear algebra over $\mathbb{Z}_q$, SIM computes a vector $\mathbf{z}' \in \mathbb{Z}_q^D$ such that $\mathbf{M} \cdot \mathbf{z}' = \mathbf{u} \bmod q$. Next, it samples $\mathbf{r}_z \xleftarrow{\$} \mathbb{Z}_q^D$, $\eta \xleftarrow{\$} \overline{\mathcal{S}}$, and randomness ρ_1, ρ_2, ρ_3 for COM. Then, it sends $\mathrm{CMT} = \big(C_1', C_2', C_3'\big)$ to $\widehat{\mathcal{V}}$, where

$$C_1' = \mathsf{COM}(\eta, \mathbf{M} \cdot \mathbf{r}_z; \rho_1),$$
$$C_2' = \mathsf{COM}(\mathit{\Gamma}_\eta(\mathbf{r}_z); \rho_2), \quad C_3' = \mathsf{COM}(\mathit{\Gamma}_\eta(\mathbf{z}' + \mathbf{r}_z); \rho_3).$$

Receiving a challenge Ch from $\widehat{\mathcal{V}}$, the simulator responds as follows:

- If $Ch = 1$: Output $\perp$ and abort.
- If $Ch = 2$: Send $\mathrm{RSP} = \big(\eta, \mathbf{z}' + \mathbf{r}_z, \rho_1, \rho_3\big)$.
- If $Ch = 3$: Send $\mathrm{RSP} = \big(\eta, \mathbf{r}_z, \rho_1, \rho_2\big)$.

Case $\overline{Ch} = 2$: SIM samples $\mathbf{z}' \xleftarrow{\$} \mathsf{VALID}$, $\mathbf{r}_z \xleftarrow{\$} \mathbb{Z}_q^D$, $\eta \xleftarrow{\$} \overline{\mathcal{S}}$, and randomness ρ_1, ρ_2, ρ_3 for COM. Then it sends $\mathrm{CMT} = \big(C'_1, C'_2, C'_3\big)$ to $\widehat{\mathcal{V}}$, where

$$C'_1 = \mathsf{COM}(\eta, \mathbf{M} \cdot \mathbf{r}_z; \rho_1),$$
$$C'_2 = \mathsf{COM}(\Gamma_\eta(\mathbf{r}_z); \rho_2), \quad C'_3 = \mathsf{COM}(\Gamma_\eta(\mathbf{z}' + \mathbf{r}_z); \rho_3).$$

Receiving a challenge Ch from $\widehat{\mathcal{V}}$, the simulator responds as follows:

- If $Ch = 1$: Send $\mathrm{RSP} = \big(\Gamma_\eta(\mathbf{z}'), \Gamma_\eta(\mathbf{r}_z), \rho_2, \rho_3\big)$.
- If $Ch = 2$: Output $\perp$ and abort.
- If $Ch = 3$: Send $\mathrm{RSP} = \big(\eta, \mathbf{r}_z, \rho_1, \rho_2\big)$.

Case $\overline{Ch} = 3$: SIM samples $\mathbf{z}' \xleftarrow{\$} \mathsf{VALID}$, $\mathbf{r}_z \xleftarrow{\$} \mathbb{Z}_q^D$, $\eta \xleftarrow{\$} \overline{\mathcal{S}}$, and randomness ρ_1, ρ_2, ρ_3 for COM. Then it sends $\mathrm{CMT} = \big(C'_1, C'_2, C'_3\big)$ to $\widehat{\mathcal{V}}$, where $C'_2 = \mathsf{COM}(\Gamma_\eta(\mathbf{r}_z); \rho_2)$, $C'_3 = \mathsf{COM}(\Gamma_\eta(\mathbf{z}' + \mathbf{r}_z); \rho_3)$ as in the previous two cases, while

$$C'_1 = \mathsf{COM}(\eta, \mathbf{M} \cdot (\mathbf{z}' + \mathbf{r}_z) - \mathbf{u}; \rho_1).$$

Receiving a challenge Ch from $\widehat{\mathcal{V}}$, it responds as follows:

- If $Ch = 1$: Send RSP computed as in the case $(\overline{Ch} = 2, Ch = 1)$.
- If $Ch = 2$: Send RSP computed as in the case $(\overline{Ch} = 1, Ch = 2)$.
- If $Ch = 3$: Output $\perp$ and abort.

We observe that, in every case, since COM is statistically hiding, the distribution of the commitment CMT and the distribution of the challenge Ch from $\widehat{\mathcal{V}}$ are statistically close to those in the real interaction. Hence, the probability that the simulator outputs $\perp$ is negligibly close to $1/3$. Moreover, one can check that whenever the simulator does not halt, it will provide an accepted transcript, the distribution of which is statistically close to that of the prover in the real interaction. In other words, we have constructed a simulator that can successfully impersonate the honest prover with probability negligibly close to $2/3$.

Argument of Knowledge. Suppose that $\mathrm{RSP}_1 = (\mathbf{t}_z, \mathbf{t}_r, \rho_2, \rho_3)$, $\mathrm{RSP}_2 = (\eta_2, \mathbf{z}_2, \rho_1, \rho_3)$, $\mathrm{RSP}_3 = (\eta_3, \mathbf{z}_3, \rho_1, \rho_2)$ are 3 valid responses to the same commitment $\mathrm{CMT} = (C_1, C_2, C_3)$, with respect to all 3 possible values of the challenge. The validity of these responses implies that:

$$\begin{cases} \mathbf{t}_z \in \mathsf{VALID};\ C_1 = \mathsf{COM}(\eta_2, \mathbf{M} \cdot \mathbf{z}_2 - \mathbf{u}; \rho_1) = \mathsf{COM}(\eta_3, \mathbf{M} \cdot \mathbf{z}_3; \rho_1); \\ C_2 = \mathsf{COM}(\mathbf{t}_r; \rho_2) = \mathsf{COM}(\Gamma_{\eta_3}(\mathbf{z}_3); \rho_2); \\ C_3 = \mathsf{COM}(\mathbf{t}_z + \mathbf{t}_r; \rho_3) = \mathsf{COM}(\Gamma_{\eta_2}(\mathbf{z}_2); \rho_3). \end{cases}$$

Since COM is computationally binding, we can deduce that:

$$\begin{cases} \mathbf{t}_z \in \mathsf{VALID};\ \eta_2 = \eta_3;\ \mathbf{t}_r = \Gamma_{\eta_3}(\mathbf{z}_3);\ \mathbf{t}_z + \mathbf{t}_r = \Gamma_{\eta_2}(\mathbf{z}_2) \bmod q; \\ \mathbf{M} \cdot \mathbf{z}_2 - \mathbf{u} = \mathbf{M} \cdot \mathbf{z}_3 \bmod q. \end{cases}$$

Since $\mathbf{t}_z \in \mathsf{VALID}$, if we let $\mathbf{z}' = [\Gamma_{\eta_2}]^{-1}(\mathbf{t}_z)$, then $\mathbf{z}' \in \mathsf{VALID}$. Furthermore, we have

$$\Gamma_{\eta_2}(\mathbf{z}') + \Gamma_{\eta_2}(\mathbf{z}_3) = \Gamma_{\phi_2}(\mathbf{z}_2) \bmod q,$$

which implies that $\mathbf{z}' + \mathbf{z}_3 = \mathbf{z}_2 \bmod q$, and that $\mathbf{M} \cdot \mathbf{z}' + \mathbf{M} \cdot \mathbf{z}_3 = \mathbf{M} \cdot \mathbf{z}_2 \bmod q$. As a result, we have $\mathbf{M} \cdot \mathbf{z}' = \mathbf{u} \bmod q$. This concludes the proof.

□

B Correctness and Security Requirements of Fully Dynamic Group Signatures

A fully dynamic group signature must satisfy *correctness* and security requirements: *anonymity*, *non-frameability*, *traceability* and *tracing soundness* [7].

We will define these requirements using a set of experiments, in which the adversary has access to a set of oracles. The oracles use the following global variables: HUL is a list of users whose keys are generated honestly. BUL is a list of users whose secret signing keys are revealed to the adversary; CUL is a list of users whose public keys are chosen by the adversary; SL is a set of signatures that are generated by the oracle Sign; CL is a set of signatures that are generated by the oracle Chal_b .

- $\mathsf{AddU}(\mathsf{uid})$. This oracle adds an honest user uid to the group at current epoch. It returns $\mathsf{upk}[\mathsf{uid}]$ and adds uid to the list HUL.
- $\mathsf{CrptU}(\mathsf{uid}, \mathsf{pk})$. This oracle allows the adversary to create a new user by choosing the key $\mathsf{upk}[\mathsf{uid}]$ as pk. It then adds uid to the list CUL. It returns $\perp$ if $\mathsf{uid} \in \mathsf{HUL} \cup \mathsf{CUL}$. This actually prepares for calling the oracle SndToGM.
- $\mathsf{SndToGM}(\mathsf{uid}, M_{\text{in}})$. This oracle engages in the $\langle \mathsf{Join}, \mathsf{Issue} \rangle$ protocol between the adversary, who corrupts the user uid in the list CUL, and the honest Issue-executing group manager.
- $\mathsf{SndToU}(\mathsf{uid}, M_{\text{in}})$. This oracle engages in the $\langle \mathsf{Join}, \mathsf{Issue} \rangle$ protocol between the adversary, who corrupts the group manager, and the honest Join−executing user uid in the list HUL.
- $\mathsf{RReg}(\mathsf{uid})$. This oracle returns registration information $\mathbf{reg}[\mathsf{uid}]$ of user uid.
- $\mathsf{MReg}(\mathsf{uid}, \rho)$. This oracle modifies $\mathbf{reg}[\mathsf{uid}]$ to any ρ chosen by the adversary.
- $\mathsf{RevealU}(\mathsf{uid})$. This oracle returns secret signing key $\mathsf{gsk}[\mathsf{uid}]$ to the adversary, and adds this user to the list BUL.
- $\mathsf{Sign}(\mathsf{uid}, M, \tau)$. This oracle returns a signature on the message M by the user uid at epoch τ, and then add the signature to the list SL.
- $\mathsf{Chal}_b(\mathsf{info}_\tau, \mathsf{uid}_0, \mathsf{uid}_1, M)$. This oracle receives inputs from the adversary, returns a signature on M by the user uid_b at epoch τ, and then adds the signature to the list CL. It is required that the two challenged users are active at epoch τ and this oracle can only be called once.
- $\mathsf{Trace}(M, \Sigma, \mathsf{Info}_\tau)$. This oracle returns the signer of the signature together with a proof, with respect to the epoch τ. We require that the signature is not in the list CL.

$\mathsf{GUpdate}(\mathcal{S})$. This oracle allows the adversary to update the group at current epoch τ_{current}. It is required that $\mathcal{S}$ is a set of active users at current epoch.

We also need the following polynomial-time algorithm in security experiments to ease composition.

$\mathsf{IsActive}(\mathsf{info}_\tau, \mathbf{reg}, \mathsf{uid}) \to 0/1$: It outputs 1 if this user is active at epoch τ and 0 otherwise.

We refer the readers to [7] for detailed descriptions of the above oracles.

Definition 5. *For any security parameter λ and any PPT adversary $\mathcal{A}$, we define correctness and security experiments in* **Fig**. *3.*

For correctness, non-frameability, traceability and tracing soundness, the advantage of the adversary is defined as the probability of outputting 1 *in the corresponding experiment. For anonymity, the advantage is defined as the absolute difference of probability of outputting* 1 *between experiment* $\mathbf{Exp}^{\mathsf{anon}-1}_{\mathsf{FDGS},\mathcal{A}}$ *and experiment* $\mathbf{Exp}^{\mathsf{anon}-0}_{\mathsf{FDGS},\mathcal{A}}$.

A fully dynamic group signature scheme is said to be correct and secure (i.e., anonymous, non-frameable, traceable and tracing sound) if the advantages of the adversary in all considered experiments are negligible in λ.

C Proof of Theorem 1

The proof of Theorem 1 is established by Lemmas 2-5 given below.

Lemma 2. *Assume that the* $\mathsf{LWE}_{n,q,\chi}$ *problem is hard. Then the given* FDGS *scheme provides anonymity in the random oracle model.*

Proof. Let $\mathcal{A}, \mathcal{B}$ be PPT algorithms, act as the adversary and challenger in the games respectively. We construct a sequence of indistinguishable games, in which the first game is the experiment $\mathbf{Exp}^{\mathsf{anon}-0}_{\mathsf{FDGS},\mathcal{A}}(\lambda)$ and the last one is the experiment $\mathbf{Exp}^{\mathsf{anon}-1}_{\mathsf{FDGS},\mathcal{A}}(\lambda)$. We will prove the lemma by demonstrating that any two consecutive games are indistinguishable. For each i, we denote by W_i the output of the adversary in game i.

Specifically, we consider the following games.

Game 0: This is exactly the experiment $\mathbf{Exp}^{\mathsf{anon}-0}_{\mathsf{FDGS},\mathcal{A}}(\lambda)$.

Game 1: This game is the same as Game 0 with only one modification: add $(\mathbf{S}_2, \mathbf{E}_2)$ to tsk. This will make no difference to the view of the adversary. So $\Pr[W_1 = 1] = \Pr[\mathbf{Exp}^{\mathsf{anon}-0}_{\mathsf{FDGS},\mathcal{A}}(\lambda) = 1]$.

Game 2: This game is identical to Game 1 except that it generates a simulated proof for the oracle Trace by programming the random oracle $\mathcal{H}_{\mathsf{FS}}$ even though the challenger $\mathcal{B}$ has correct witnesses to generate a real proof. The view of the adversary, however, is statistically indistinguishable between Game 1 and Game 2 by statistical zero-knowledge property of our argument system generating Π_{trace}. Therefore we have $\Pr[W_1 = 1] \approx \Pr[W_2 = 1]$.

Experiment: $\mathbf{Exp}^{\mathsf{corr}}_{\mathsf{FDGS},\mathcal{A}}(\lambda)$

$\mathsf{pp} \leftarrow \mathsf{GSetup}(\lambda), \mathsf{HUL} := \emptyset.$
$\langle(\mathsf{info}, \mathsf{mpk}, \mathsf{msk}); (\mathsf{tpk}, \mathsf{tsk})\rangle \leftarrow \langle \mathsf{GKgen}_{\mathsf{GM}}(\mathsf{pp}), \mathsf{GKgen}_{\mathsf{TM}}(\mathsf{pp})\rangle.$
Set $\mathsf{gpk} = (\mathsf{pp}, \mathsf{mpk}, \mathsf{tpk})$. $(\mathsf{uid}, M, \tau) \leftarrow \mathcal{A}^{\mathsf{AddU},\mathsf{RReg},\mathsf{GUpdate}}(\mathsf{gpk}, \mathsf{info})$.
If $\mathsf{uid} \notin \mathsf{HUL}$ or $\mathsf{gsk}[\mathsf{uid}] = \bot$ or $\mathsf{info}_\tau = \bot$ or $\mathsf{IsActive}(\mathsf{info}_\tau, \mathbf{reg}, \mathsf{uid}) = 0$, return 0.
$\Sigma \leftarrow \mathsf{Sign}(\mathsf{gpk}, \mathsf{gsk}[\mathsf{uid}], \mathsf{info}_\tau, M)$, $(\mathsf{uid}^*, \Pi_{\mathsf{trace}}) \leftarrow \mathsf{Trace}(\mathsf{gpk}, \mathsf{tsk}, \mathsf{info}_\tau, \mathbf{reg}, M, \Sigma)$.
Return 1 if $(\mathsf{Verify}(\mathsf{gpk}, \mathsf{info}_\tau, M, \Sigma) = 0 \vee \mathsf{Judge}(\mathsf{gpk}, \mathsf{uid}, \mathsf{info}_\tau, \Pi_{\mathsf{trace}}, m, \Sigma) = 0 \vee \mathsf{uid} \neq \mathsf{uid}^*)$.

Experiment: $\mathbf{Exp}^{\mathsf{anon-b}}_{\mathsf{FDGS},\mathcal{A}}(\lambda)$

$\mathsf{pp} \leftarrow \mathsf{GSetup}(\lambda), \mathsf{HUL}, \mathsf{CUL}, \mathsf{BUL}, \mathsf{CL}, \mathsf{SL} := \emptyset.$
$(\mathsf{st}_{\mathsf{init}}, \mathsf{info}, \mathsf{mpk}, \mathsf{msk}) \leftarrow \mathcal{A}^{\langle\cdot, \mathsf{GKgen}_{\mathsf{TM}}\rangle(\mathsf{pp})}(\mathsf{init} : \mathsf{pp})$.
Return 0 if $\mathsf{GKgen}_{\mathsf{TM}}$ did not accept or $\mathcal{A}$'s output is not well-formed.
Denote the output of $\mathsf{GKgen}_{\mathsf{TM}}$ as $(\mathsf{tpk}, \mathsf{tsk})$, and set $\mathsf{gpk} = (\mathsf{pp}, \mathsf{mpk}, \mathsf{tpk})$.
$b^* \leftarrow \mathcal{A}^{\mathsf{AddU},\mathsf{CrptU},\mathsf{RevealU},\mathsf{SndToU},\mathsf{Trace},\mathsf{MReg},\mathsf{Chal}_\mathsf{b}}(\mathsf{play} : \mathsf{st}_{\mathsf{init}}, \mathsf{gpk})$. Return b^*.

Experiment: $\mathbf{Exp}^{\mathsf{non-frame}}_{\mathsf{FDGS},\mathcal{A}}(\lambda)$

$\mathsf{pp} \leftarrow \mathsf{GSetup}(\lambda), \mathsf{HUL}, \mathsf{CUL}, \mathsf{BUL}, \mathsf{SL} = \emptyset.$
$(\mathsf{st}_{\mathsf{init}}, \mathsf{info}, \mathsf{mpk}, \mathsf{msk}, \mathsf{tpk}, \mathsf{tsk}) \leftarrow \mathcal{A}(\mathsf{init} : \mathsf{pp})$.
Return 0 if $\mathcal{A}$'s output is not well-formed, otherwise set $\mathsf{gpk} = (\mathsf{pp}, \mathsf{mpk}, \mathsf{tpk})$.
$(M, \Sigma, \mathsf{uid}, \Pi_{\mathsf{trace}}, \mathsf{info}_\tau) \leftarrow \mathcal{A}^{\mathsf{CrptU},\mathsf{RevealU},\mathsf{SndToU},\mathsf{MReg},\mathsf{Sign}}(\mathsf{play} : \mathsf{st}_{\mathsf{init}}, \mathsf{gpk})$.
Return 1 if $(\mathsf{Verify}(\mathsf{gpk}, \mathsf{info}_\tau, M, \Sigma) = 1 \wedge \mathsf{Judge}(\mathsf{gpk}, \mathsf{uid}, \mathsf{info}_\tau, \Pi_{\mathsf{trace}}, M, \Sigma) = 1 \wedge \mathsf{uid} \in \mathsf{HUL} \setminus \mathsf{BUL} \wedge (M, \Sigma, \tau) \notin \mathsf{SL})$.

Experiment: $\mathbf{Exp}^{\mathsf{trace}}_{\mathsf{FDGS},\mathcal{A}}(\lambda)$

$\mathsf{pp} \leftarrow \mathsf{GSetup}(\lambda), \mathsf{HUL}, \mathsf{CUL}, \mathsf{BUL}, \mathsf{SL} = \emptyset.$
$(\mathsf{st}_{\mathsf{init}}, \mathsf{tpk}, \mathsf{tsk}) \leftarrow \mathcal{A}^{\langle \mathsf{GKgen}_{\mathsf{GM}}(\mathsf{pp}), \cdot\rangle}(\mathsf{init} : \mathsf{pp})$.
Return 0 if $\mathsf{GKgen}_{\mathsf{GM}}$ did not accept or $\mathcal{A}$'s output is not well-formed.
Denote the output of $\mathsf{GKgen}_{\mathsf{GM}}$ as $(\mathsf{mpk}, \mathsf{msk}, \mathsf{info})$, and set $\mathsf{gpk} = (\mathsf{pp}, \mathsf{mpk}, \mathsf{tpk})$.
$(M, \Sigma, \tau) \leftarrow \mathcal{A}^{\mathsf{AddU},\mathsf{CrptU},\mathsf{SndToGM},\mathsf{RevealU},\mathsf{MReg},\mathsf{Sign},\mathsf{GUpdate}}(\mathsf{play} : \mathsf{st}_{\mathsf{init}}, \mathsf{gpk}, \mathsf{info})$.
If $\mathsf{Verify}(\mathsf{gpk}, \mathsf{info}_\tau, M, \Sigma) = 0$, return 0.
$(\mathsf{uid}, \Pi_{\mathsf{trace}}) \leftarrow \mathsf{Trace}(\mathsf{gpk}, \mathsf{tsk}, \mathsf{info}_\tau, \mathbf{reg}, M, \Sigma)$.
Return 1 if $(\mathsf{IsActive}(\mathsf{info}_\tau, \mathbf{reg}, \mathsf{uid}) = \bot \vee \mathsf{Judge}(\mathsf{gpk}, \mathsf{uid}, \mathsf{info}_\tau, \Pi_{\mathsf{trace}}, M, \Sigma) = 0 \vee \mathsf{uid} = 0)$.

Experiment: $\mathbf{Exp}^{\mathsf{trace-sound}}_{\mathsf{FDGS},\mathcal{A}}(\lambda)$

$\mathsf{pp} \leftarrow \mathsf{GSetup}(\lambda), \mathsf{CUL} = \emptyset.$ $(\mathsf{st}_{\mathsf{init}}, \mathsf{info}, \mathsf{mpk}, \mathsf{msk}, \mathsf{tpk}, \mathsf{tsk}) \leftarrow \mathcal{A}(\mathsf{init} : \mathsf{pp})$.
Return 0 if $\mathcal{A}$'s output is not well-formed, otherwise set $\mathsf{gpk} = (\mathsf{pp}, \mathsf{mpk}, \mathsf{tpk})$.
$(M, \Sigma, \mathsf{uid}_0, \Pi_{\mathsf{trace},0}, \mathsf{uid}_1, \Pi_{\mathsf{trace},1}, \mathsf{info}_\tau) \leftarrow \mathcal{A}^{\mathsf{CrptU},\mathsf{MReg}}(\mathsf{play} : \mathsf{st}_{\mathsf{init}}, \mathsf{gpk})$.
Return 1 if $(\mathsf{Verify}(\mathsf{gpk}, \mathsf{info}_\tau, M, \Sigma) = 1 \wedge \mathsf{uid}_0 \ (\neq \bot) \neq \mathsf{uid}_1 \ (\neq \bot) \wedge \mathsf{Judge}(\mathsf{gpk}, \mathsf{uid}_b, \mathsf{info}_\tau, \Pi_{\mathsf{trace},b}, M, \Sigma) = 1$ for $b \in \{0, 1\})$.

Fig. 3: Experiments to define security requirements of FDGS.

Game 3: This game uses $\mathbf{S}_2$ instead of $\mathbf{S}_1$ to simulate the oracle Trace. The view of $\mathcal{A}$ is the same as in Game 2 until event F_1, where $\mathcal{A}$ queries the trace oracle a valid signature $(M, \Pi_{\mathsf{gs}}, \mathbf{c}_1, \mathbf{c}_2, \mathsf{info}_\tau)$ with $\mathbf{c}_1, \mathbf{c}_2$ encrypting distinct bit strings, happens. Since F_1 breaks the soundness of our argument generating Π_{gs}, we have $|\Pr[W_2 = 1] - \Pr[W_3 = 1]| \leq \Pr[F_1] \leq \mathsf{Adv}^{\mathsf{sound}}_{\Pi_{\mathsf{gs}}}(\lambda) = \mathsf{negl}(\lambda)$.

Game 4: This game generates a simulated proof for the oracle Chal. The view is indistinguishable to $\mathcal{A}$ by statistical zero-knowledge property of our argument system generating Π_{gs}. Therefore we have $\Pr[W_3 = 1] \approx \Pr[W_4 = 1]$.

Game 5: This is the same as Game 4 except that $\mathbf{c}_1$ is now encryption of $\mathsf{bin}(i_1)$ while $\mathbf{c}_2$ is still encryption of $\mathsf{bin}(i_0)$ for the Chal query. By the semantic security of our encryption scheme for public key $(\mathbf{B}, \mathbf{P}_1)$ (which relies on LWE assumption), this change is negligible to the adversary. Recall that we use $\mathbf{S}_2$ for the Trace queries. So the change of $\mathbf{c}_1$ makes no difference to the view of the adversary. Therefore we have $|\Pr[W_4 = 1] - \Pr[W_5 = 1]| = \mathsf{negl}(\lambda)$.

Game 6: This game follows Game 5 with one change: it switches back to use $\mathbf{S}_1$ for the Trace queries and discards $(\mathbf{S}_2, \mathbf{E}_2)$. This modification is indistinguishable to the adversary until event F_2, where $\mathcal{A}$ queries the trace oracle a valid signature $(M, \Pi_{\mathsf{gs}}, \mathbf{c}_1, \mathbf{c}_2, \mathsf{info}_\tau)$ with $\mathbf{c}_1, \mathbf{c}_2$ encrypting different bit strings, happens. However event F_2 breaks the simulation soundness of the argument system generating Π_{gs}. Therefore we have $|\Pr[W_5 = 1] - \Pr[W_6 = 1]| \leq \mathsf{Adv}^{\mathsf{ss}}_{\Pi_{\mathsf{gs}}}(\lambda) = \mathsf{negl}(\lambda)$.

Game 7: In this game, it changes $\mathbf{c}_2$ to be encryption of $\mathsf{bin}(i_1)$ for the Chal query while the remaining parts are the same as in Game 6. By the semantic security of our encryption scheme for public key $(\mathbf{B}, \mathbf{P}_2)$, this change is negligible to the adversary. Note that we now use $\mathbf{S}_1$ for the Trace queries. So changing $\mathbf{c}_2$ makes no difference to the view of the adversary. Therefore we have $|\Pr[W_6 = 1] - \Pr[W_7 = 1]| = \mathsf{negl}(\lambda)$.

Game 8: We now generate real proof for the Chal query instead of simulated proof. By the statistical zero-knowledge property of our argument system generating Π_{gs}, the view of the adversary in Game 7 and Game 8 are statistically indistinguishable, i.e., we have $\Pr[W_7 = 1] \approx \Pr[W_8 = 1]$.

Game 9: This game produces real proof, for the Trace queries. By the statistical zero-knowledge property of our argument system generating Π_{trace}, we have Game 8 and Game 9 are statistically indistinguishable to the adversary, i.e., we have $\Pr[W_8 = 1] \approx \Pr[W_9 = 1]$. This is actually the experiment $\mathbf{Exp}^{\mathsf{anon}-1}_{\mathsf{FDGS},\mathcal{A}}(\lambda)$. Hence, we have $\Pr[W_9 = 1] = \Pr[\mathbf{Exp}^{\mathsf{anon}-1}_{\mathsf{FDGS},\mathcal{A}}(\lambda) = 1]$.

As a result, we have that

$$|\Pr[\mathbf{Exp}^{\mathsf{anon}-1}_{\mathsf{FDGS},\mathcal{A}}(\lambda) = 1] - \Pr[\mathbf{Exp}^{\mathsf{anon}-0}_{\mathsf{FDGS},\mathcal{A}}(\lambda) = 1]| = \mathsf{negl}(\lambda),$$

and hence our scheme is anonymous. □

Lemma 3. *Assume that the* $\mathsf{SIS}^{\infty}_{n,m,q,1}$ *problem is hard. Then the given* FDGS *scheme provides non-frameability in the random oracle model.*

Proof. We prove non-frameability by contradiction. Suppose that $\mathcal{A}$ succeeds with non-negligible advantage ϵ. Then we can build a PPT algorithm $\mathcal{B}$ that either breaks the security of our accumulator or solves $\mathsf{SIS}^{\infty}_{n,q,m,1}$ problem, which are featured by $\mathbf{A}$, with also non-negligible probability.

Given a matrix $\mathbf{A}$ from either environment that $\mathcal{B}$ is in, it first generates all parameters pp as we do in GSetup, then invokes $\mathcal{A}$ with pp, and then proceeds as described in the experiment. Here $\mathcal{B}$ can consistently answer all the oracle queries made by $\mathcal{A}$. When $\mathcal{A}$ outputs $(M^*, \Sigma^*, \mathsf{bin}(j^*), \Pi^*_{\mathsf{trace}}, \mathsf{info}_\tau)$ and wins, $\mathcal{B}$ does following.

Parse Σ^* as $(\Pi^*_{\mathsf{gs}}, \mathbf{c}^*_1, \mathbf{c}^*_2)$, where $\Pi^*_{\mathsf{gs}} = (\{\mathsf{CMT}^*_i\}^{\kappa}_{i=1}, \mathsf{CH}^*, \{\mathsf{RSP}^*_i\}^{\kappa}_{i=1})$ and RSP^*_i is a valid response w.r.t CMT^*_i and CH^*_i for $i \in [\kappa]$ since $\mathcal{A}$ wins. We claim that $\mathcal{A}$ queried the tuple $(M^*, \{\mathsf{CMT}^*_i\}^{k}_{i=1}, \mathbf{A}, \mathbf{u}_\tau, \mathbf{B}, \mathbf{P}_1, \mathbf{P}_2, \mathbf{c}^*_1, \mathbf{c}^*_2)$, denoted as ξ^*, to the hash oracle $\mathcal{H}_{\mathsf{FS}}$ with overwhelming probability. Otherwise guessing correctly this value occurs only with probability $3^{-\kappa}$, which is negligible. Therefore, with probability $\epsilon' = \epsilon - 3^{-\kappa}$, the tuple ξ^* has been an input of one hash query, denoted as $t^* \in \{1, 2 \cdots, Q_H\}$, where Q_H is the total number of hash queries made by $\mathcal{A}$. To employ the Forking lemma of Brickell et al. [11], $\mathcal{B}$ runs polynomial-number executions of $\mathcal{A}$ exactly the same as in the original run until the t^* hash query, that is, from t^* hash query on, $\mathcal{B}$ will answer $\mathcal{A}$ with fresh and independent values for hash queries on each new execution. Note that the input of t^* hash query must be ξ^* as in the original run. By the Forking Lemma [11], with probability $\geq \frac{1}{2}$, $\mathcal{B}$ obtains 3-fork involving the same tuple ξ^* with pairwise distinct hash value $\mathsf{CH}^{(1)}_{t^*}$,$\mathsf{CH}^{(2)}_{t^*}$,$\mathsf{CH}^{(3)}_{t^*} \in \{1,2,3\}^{\kappa}$. We have $(\mathsf{CH}^{(1)}_{t^*,j}, \mathsf{CH}^{(2)}_{t^*,j}, \mathsf{CH}^{(3)}_{t^*,j}) = (1,2,3)$ with probability $1 - (\frac{7}{9})^{\kappa}$ for some $j \in \{1, 2, \ldots, \kappa\}$. Then by the argument of knowledge of the system generating Π_{gs}, from the valid responses $(\mathsf{RSP}^{(1)}_{t^*,j}, \mathsf{RSP}^{(2)}_{t^*,j}, \mathsf{RSP}^{(3)}_{t^*,j})$, we can extract the witnesses $\zeta' = (\mathbf{x}', \mathbf{p}', w'_\tau, \mathbf{r}'_1, \mathbf{r}'_2)$, where

$$w'_\tau = (\mathsf{bin}(j'), \mathbf{w'}_{\ell,\tau}, \ldots, \mathbf{w'}_{1,\tau}) \in \{0,1\}^{\ell} \times (\{0,1\}^{nk})^{\ell},$$

such that we have:

(i) $\mathsf{TVerify}_{\mathbf{A}}(\mathbf{u}_\tau, \mathbf{p}', w'_\tau) = 1, \mathbf{p}' \neq 0$;
(ii) $\mathbf{A} \cdot \mathbf{x}' = \mathbf{G} \cdot \mathbf{p}' \mod q$;
(iii) $\mathbf{c}^*_1, \mathbf{c}^*_2$ are encryptions of $\mathsf{bin}(j')$ with randomness $\mathbf{r}'_1$ and $\mathbf{r}'_2$.

By the correctness of our encryption scheme, $\mathbf{c}^*_1$ is decrypted to $\mathsf{bin}(j')$. The fact $\mathcal{A}$ wins implies algorithm Judge outputs 1. By the soundness of our system generating Π_{trace}, $\mathsf{bin}(j^*)$ is decrypted from $\mathbf{c}^*_1$. Hence we have $\mathsf{bin}(j') = \mathsf{bin}(j^*)$ with overwhelming probability.

Now we consider the following two cases:

Case 1: $\mathsf{bin}(j')$ is a inactive user at epoch τ, i.e., the value at leaf $\mathsf{bin}(j')$ is $\mathbf{0}^{nk}$. This violates the accumulator security since we have $\mathsf{TVerify}_{\mathbf{A}}(\mathbf{u}_\tau, \mathbf{p}', w'_\tau) = 1$ with $\mathbf{p}' \neq \mathbf{0}$ at leaf $\mathsf{bin}(j')$.

Case 2: $\mathsf{bin}(j^*)$ is an active user at epoch τ. The fact $\mathsf{bin}(j^*) \in \mathsf{HUL} \setminus \mathsf{BUL}$ indicates $\mathcal{A}$ does not know $\mathsf{gsk}[j^*] = (\mathsf{bin}(j^*), \mathbf{p}', \mathbf{x}_{j^*})$, where $\mathbf{x}_{j^*}$ was initially

chosen by $\mathcal{B}$ and satisfies $\mathbf{A} \cdot \mathbf{x}_{j^*} = \mathbf{G} \cdot \mathbf{p}' \mod q$. Following the same argument of [29], we claim that $\mathbf{x}_{j^*} \neq \mathbf{x}'$ with probability at least 1/2. Then we find a nonzero vector $\mathbf{z} = \mathbf{x}_{j^*} - \mathbf{x}'$ satisfying $\mathbf{A}\mathbf{z} = 0 \mod q$. Recall $\mathbf{x}_{j^*}, \mathbf{x}' \in \{0,1\}^m$. Thus $\mathcal{B}$ solves a $\mathsf{SIS}^{\infty}_{n,q,m,1}$ problem with non-negligible probability if $\mathcal{A}$ breaks the non-frameability of our construction with non-negligible probability. □

Lemma 4. *Assume that the* $\mathsf{SIS}^{\infty}_{n,m,q,1}$ *problem is hard. Then the given* FDGS *scheme satisfies traceability in the random oracle model.*

Proof. Recall that adversary $\mathcal{A}$ wins traceability experiment if it generates a valid signature that either: (i) traces to an inactive user; or (ii) traces to an active user but we cannot generate a proof accepted by the algorithm Judge. We will prove that both cases occur with negligible probability.

Let $(\mathsf{info}_\tau, M, \Sigma)$ be the output of $\mathcal{A}$ in the experiment $\mathbf{Exp}^{\mathsf{trace}}_{\mathsf{FDGS},\mathcal{A}}(\lambda)$ and we compute $(\mathsf{bin}(j), \Pi_{\mathsf{trace}})$ by running the algorithm Trace . Parse Σ as $(\Pi_{\mathsf{gs}}, \mathbf{c}_1, \mathbf{c}_2)$, where $\Pi_{\mathsf{gs}} = (\{\mathsf{CMT}_i\}_{i=1}^{\kappa}, \mathsf{CH}, \{\mathsf{RSP}_i\}_{i=1}^{\kappa})$, and RSP_i is a valid response w.r.t CMT_i and CH_i for $i \in [\kappa]$ since $(\mathsf{info}_\tau, M, \Sigma)$ is a valid signature outputted by $\mathcal{A}$. Now we are in the same situation as in Lemma 3. By the same technique, we can extract witnesses $\zeta = (\mathbf{x}, \mathbf{p}, w_\tau, \mathbf{r}_1, \mathbf{r}_2)$, where $w_\tau = (\mathsf{bin}(j'), \mathbf{w}_{\ell,\tau}, \ldots, \mathbf{w}_{1,\tau}) \in \{0,1\}^\ell \times (\{0,1\}^{nk})^\ell$ satisfying $\mathsf{TVerify}_{\mathbf{A}}(\mathbf{u}_\tau, \mathbf{p}, w_\tau) = 1$, $\mathbf{p} \neq \mathbf{0}^{nk}$, and $\mathbf{c}_1, \mathbf{c}_2$ are encryptions of $\mathsf{bin}(j')$ using randomness $\mathbf{r}_1$ and $\mathbf{r}_2$. Correctness of our encryption scheme implies $\mathbf{c}_1$ is decrypted to $\mathsf{bin}(j')$. Correct decryption of the Trace algorithm, which is run by the challenger, indicates that $\mathsf{bin}(j)$ is decrypted from $\mathbf{c}_1$. Hence we have $\mathsf{bin}(j') = \mathsf{bin}(j)$ with overwhelming probability.

Now, we note that case (i) considered above only happens with negligible probability. In fact, if $\mathsf{bin}(j)$ is not active at epoch τ, and $\mathsf{TVerify}_{\mathbf{A}}(\mathbf{u}_\tau, \mathbf{p}, w_\tau) = 1$ with $\mathbf{p} \neq \mathbf{0}$ at leaf $\mathsf{bin}(j)$, then the security of our accumulator is violated. Furthermore, case (ii) also occurs with negligible probability since the challenger possesses valid witnesses to generate the proof Π_{trace}, which will be accepted by the Judge algorithm, thanks to the completeness of the underlying argument system. It then follows that our scheme satisfies the traceability requirement.

□

Lemma 5. *The given* FDGS *scheme satisfies tracing soundness in the random oracle model.*

Proof. Let $(M, \Sigma, \mathsf{bin}(j_0), \Pi_{\mathsf{trace},0}, \mathsf{bin}(j_1), \Pi_{\mathsf{trace},1}, \mathsf{info}_\tau)$ be the output of the adversary in the experiment $\mathbf{Exp}^{\mathsf{trace-sound}}_{\mathsf{FDGS},\mathcal{A}}(\lambda)$. Recall that adversary wins if the following conditions hold:

(i) $\mathsf{bin}(j_0) \neq \mathsf{bin}(j_1)$;

(ii) $\mathsf{Verify}(\mathsf{gpk}, \mathsf{info}_\tau, M, \Sigma) = 1$;

(iii) $\mathsf{Judge}(\mathsf{gpk}, \mathsf{bin}(j_b), \mathsf{info}_\tau, \Pi_{\mathsf{trace},b}, M, \Sigma) = 1$ for $b \in \{0,1\}$;

(iv) $\mathsf{bin}(j_b) \neq \perp$ for $b \in \{0,1\}$.

Let us parse $\Pi_{\mathsf{trace},b}$ as $(\{\mathsf{CMT}_{i,b}\}_{i=1}^{\kappa}, \mathsf{CH}_b, \{\mathsf{RSP}_{i,b}\}_{i=1}^{\kappa})$. The fact that the Judge algorithm outputs 1 implies that $\mathsf{RSP}_{i,b}$ is a valid response w.r.t $\mathsf{CMT}_{i,b}$

and $\mathsf{CH}_{i,b}$ for $i \in [\kappa]$ and $b \in \{0,1\}$. By using the extraction technique for argument system generating Π_{trace} as in Lemma 3, we can extract the witnesses $\mathbf{S}_{1,b}, \mathbf{E}_{1,b}, \mathbf{y}_b$ for $b \in \{0,1\}$ satisfying:

$$\begin{cases} \|\mathbf{S}_{1,b}\|_\infty \leq \beta;\ \|\mathbf{E}_{1,b}\|_\infty \leq \beta;\ \|\mathbf{y}_b\|_\infty \leq \lceil q/5 \rceil; \\ \mathbf{S}_{1,b}^\top \cdot \mathbf{B} + \mathbf{E}_{1,b} = \mathbf{P}_1 \bmod q; \\ \mathbf{c}_{1,2} - \mathbf{S}_{1,b}^\top \cdot \mathbf{c}_{1,1} = \mathbf{y}_b + \lfloor q/2 \rfloor \cdot \mathsf{bin}(j_b) \bmod q. \end{cases} \tag{9}$$

Note that, we have $(\mathbf{S}_{1,0}^\top - \mathbf{S}_{1,1}^\top) \cdot \mathbf{c}_{1,1} = \mathbf{y}_1 - \mathbf{y}_0 + \lfloor q/2 \rfloor \cdot (\mathsf{bin}(j_1) - \mathsf{bin}(j_0)) \bmod q$. Suppose that $\mathsf{bin}(j_0) \neq \mathsf{bin}(j_1)$, then $\|\lfloor q/2 \rfloor \cdot (\mathsf{bin}(j_1) - \mathsf{bin}(j_0))\|_\infty = \lfloor q/2 \rfloor$. On the other hand, we have $\|\mathbf{y}_1 - \mathbf{y}_0\|_\infty \leq 2 \cdot \lceil q/5 \rceil$. It then follows that

$$\|\mathbf{y}_1 - \mathbf{y}_0 + \lfloor q/2 \rfloor \cdot (\mathsf{bin}(j_1) - \mathsf{bin}(j_0))\|_\infty > 0,$$

which implies that $\mathbf{S}_{1,0}^\top \neq \mathbf{S}_{1,1}^\top$. In other words, we can obtain two solutions for the equation $\mathbf{S}^\top \cdot \mathbf{B} + \mathbf{E} = \mathbf{P}_1 \bmod q$, which contradicts the fact that there exists at most one solution for LWE samples $(\mathbf{B}, \mathbf{P}_1)$.

By contradiction, we have $\mathsf{bin}(j_0) = \mathsf{bin}(j_1)$ with overwhelming probability. This implies that case (i) does not hold with overwhelming probability once cases (ii),(iii) and (iv) hold. As a result, the advantage of the adversary attacking the tracing soundness of our scheme is negligible (in λ). In other words, our construction satisfies tracing soundness. □