Everlasting Fully Dynamic Group Signatures

Yimeng He[1], San Ling[1], Khai Hanh Tang[1], and Huaxiong Wang[1]

Nanyang Technological University, 50 Nanyang Ave, Singapore
yimeng002@e.ntu.edu.sg, {lingsan,khaihanh.tang,hxwang}@ntu.edu.sg

Abstract. Group signatures allow a user to sign anonymously on behalf of a group of users while allowing a tracing authority to trace the signer's identity in case of misuse. In Chaum and van Heyst's original model (EUROCRYPT'91), the group needs to stay fixed. Throughout various attempts, including partially dynamic group signatures and revocations, Bootle et al. (ACNS'16, J. Cryptol.) formalized the notion of fully dynamic group signatures (FDGS), enabling both enrolling and revoking users of the group. However, in their scheme, the verification process needs to take into account the latest system information, and a previously generated signature will be invalidated as soon as, for example, there is a change in the group. We therefore raise a research question: Is it possible to construct an FDGS under which the validity of a signature can survive future changes in the system information?
In this paper, we propose Everlasting Fully Dynamic Group Signatures (EFDGS) that allow signers to generate signatures that do not require verification with any specific epoch. Specifically, once the signatures are created, they are valid forever. It also guarantees that the signer can only output such a signature when she is a valid user of the system. We realize the above new model by constructing a plausibly post-quantum standard-lattice-based EFDGS.

Keywords: Group signatures · full dynamicity · privacy-preserving

1 Introduction

Group Signatures and Dynamicity. Group signatures, originally proposed by Chaum and van Heyst [30] and later formalized by Bellare, Micciancio, and Warinschi [12], allow a person, among a group of users, to sign on behalf of this group. Moreover, the output group signatures do not leak the signers' identities. In addition, a group signature scheme must have a special authority, whom we call the tracing authority, responsible for tracing signers' identities behind the signatures in case of misconduct. Specifically, a group signature has a group public key gpk and a group of users where each user, with an identifier id and a signing key $\mathsf{sk}_{\mathsf{id}}$, is allowed to sign on behalf of all users in the group. The tracing authority has a special secret key $\mathsf{sk}_{\mathsf{ta}}$ enabling her to trace the identifiers behind the signatures produced by the group.

Nevertheless, early results of group signatures focus on the "static setting": once created, the group users cannot be changed. Although this kind of setting attracted a lot of developments [31,23,27,76,26,5,4,3], it restricts the applicability of group signatures to many potential applications. A trivial way to change the group of users is by generating a new group with the desired users. However, as we can imagine, this will cause a lot of inefficiency since we need to create new keys for the users and the tracing authority. A related approach, by updating the keys, is implemented by [81,16].

Partial Dynamicity. During the development of group signatures, new models and constructions of group signatures were proposed for obtaining the dynamicity of group signature schemes. Bellare, Shi, and Zhang [13] introduce the new notion called dynamic group signatures. In a dynamic group signature scheme, besides the users and the tracing authority, there is another authority called the issuer, who enrolls new users into the system by issuing new signing keys for newly enrolled users. The dynamic group setting in [13] is usually called partial dynamicity, while the issuer can be called the group manager in other results. In this manuscript, we call the issuer (as well as the group manager) the system authority. The model of partially dynamic group signatures was later improved by Kiayias and Yung [48] to allow the system authority to enroll new users via some interactive protocol to prevent the system authority from learning the signing keys of the users in the system. This model leads to the formalization of partially dynamic group signatures, which requires such a secure scheme to satisfy correctness, traceability (a.k.a. security against misidentification attacks), and non-frameability. Correctness guarantees that, if the signatures are generated honestly, they are always valid. Traceability prevents any corrupt user from creating new signatures traceable by the tracing authority outside the set of corrupt users. Non-frameability prevents framing a signature to some honest user that the adversary does not have the corresponding signing key.

The notion of partial dynamicity certainly does not support user revocation. Hence, this leads to certain security and efficiency concerns, e.g., a revoked user still keeps the signing key for authorizing messages in the future (security concern), or, to revoke, simply re-generating the system with the group of unrevoked users only (efficiency concern). Consequently, another line of research work about revocation and full dynamicity has appeared.

Revocation and Full Dynamicity. Early attempts to enable revocations include the following themes. The first theme from [20,65,55,54] is for showing that the signer's identifier is not in a public revocation list. Another theme from [25,84,24,32,73] is for using updatable accumulators to accumulate the set of unrevoked users' identifiers into a short representation. [81] and [16] update the signing keys of unrevoked users. Other themes including verifier-local revocation (VLR) [21,17,66,56,49], direct anonymous attestation (DAA) [22], and traceable signatures [47] are also taken into consideration in which only verifier knows the revocation list while signers do not.

Until 2016, the concept of fully dynamic group signatures (FDGS) was formally proposed by Bootle et al. [18,19], encompassing a long line of approaches attempting to formulate and incorporate the revocation mechanism into the group signature schemes. An FDGS scheme is not only partially dynamic, where the system authority can enroll new users via some joining mechanism, but also allows the system authority to revoke users when necessary. This notion is formalized by having the system maintain a public system information of each epoch. Here, an epoch is an interval of time in which there is no change in the group of users. Suppose a change to the group happens, namely, by enrolling new users or revoking some users. In that case, the system is moved to the next epoch by having the system information updated accordingly. In other words, if we do not update the system information when changing the epoch, a revoked user still remembers her signing key for further signings. Hence, when verifying the FDGS signatures, the verifier must verify with inputs containing the corresponding public system information, the message, and the signature.

This model of Bootle et al. is later followed by [60,33,83,82,9], and other variants including fully dynamic attribute-based signatures [58], fully dynamic group encryptions [70,75,71], and fully dynamic secret handshakes [2].

Drawbacks of FDGS Model from Bootle et al. [18,19]. As said above, the FDGS signatures in the model from Bootle et al. require verification with the public system information of a specific epoch. That is, creating a signature with system information of epoch τ while verifying it with system information of epoch τ' can only succeed if and only if $\tau = \tau'$. If $\tau \neq \tau'$, the FDGS signature is deemed to be invalid. When implemented in real systems, this model requires the signer to create each FDGS signature, with respect to system information of the current epoch (current time or latest epoch), and to send it immediately in the same current epoch for verification. However, this strategy of creating and verifying signatures with the same system information of a specific epoch may incur some drawbacks. We consider the following examples.

Inability to Delay Publishing Signatures. If a user is an employee of a company and will leave the company soon, in this case, we assume that this user is revoked in the next epoch of the group signature scheme. However, when being an employee of the company, this user authorizes a valid signature of some contract for some secret deal with another partner of the company. For some confidential purposes, this contract should be kept private for verification between the two companies. The signature may later need to be publicly verified, for example, under police investigations or when the signer would like to claim some rewards from signing the contract for the company to obtain some achievement. However, at the moment, the group information has already changed, and the signer is no longer with the company. We cannot verify the validity of the signature since the mentioned signature was authorized in the past. This can be explained as follows. Suppose the signer was revoked from the company. How can we know either (i) the signature was created when the signer was an employee of the company or (ii) the signer was revoked while keeping the signing key for authorizing with respect to the old system information?

A potential solution for this situation is to publish the message and the signature immediately. However, as said earlier, some private deals should not be published early due to compliance with business secrets.

Denying Ownership of Old Signatures. Consider another example. Assume that some users abuse the system by signing bad messages, e.g., corruption, which may affect the company's reputation. For later investigation, if this user is revoked from the system, she can deny the signatures she created by stating that those signatures are invalid with respect to the current system information, as she is no longer a valid user. Moreover, since she keeps the old keys, she can create garbage signatures that cannot be distinguished from those before being revoked.

Nature of Signatures. Notice that the nature of signatures, i.e., ordinary signatures, ensures that signatures are valid forever without being limited in any time interval. Additionally, the signer cannot deny the signatures that she created.

Hence, practical necessity demands an FDGS scheme whose verifications at different epochs do not require the associated system information. In addition, we can trust that the signatures were generated appropriately when the signer was a valid user in the system. In other words, whenever a valid signature is created, we trust that it is generated honestly by an existing user in the group and not from anyone outside the group (including revoked users). At the same time, verification is not necessarily immediate (at the current epoch) with respect to the most updated system information, and abuse users cannot deny the signature when epochs flow. Hence, we ask the following question.

Can we construct an FDGS scheme whose signatures are valid forever while simultaneously allowing delaying publishing them?

A recent result [86] attempted to answer the above question. However, their construction requires updating the group public key after every change of epochs. Doing so is not fundamentally different from altering the system information, as a once-valid signature will become invalid under the updated group public key.

Heavy Cost of Maintaining System. As a trivial approach, one may consider compiling an FDGS [18,60,19] into some other scheme allowing validity forever as follows. A user issues an FDGS signature and sends it to the system authority for publishing to some public ledger. Then, the user later can produce a ZK proof saying that there is an FDGS in the ledger at some epoch (only known to the user), she is a valid member at the time of publishing the FDGS, and the FDGS signature is valid with respect to the mentioned epoch. By the ZK property, we can hide the epoch. However, we should note that the ledger expands when more FDGS signatures are added. Hence, producing and verifying such a ZK proof requires the ledger's state to be included as an input. We would need a large and immutable database representing the ledger (realizable by blockchains with additional costs for miners/validators to maintain).

1.1 Our Contributions

We answer the above question with the following contributions.

We propose a new FDGS scheme called everlasting fully dynamic group signature (EFDGS) scheme that allows signers to generate signatures that do not require verification with any specific epoch. Specifically, once the signatures are created, they are valid forever (hence "everlasting"). It also guarantees that the signer can only output such a signature when she is a valid user of the system. In brief, we model this privacy-preserving signature scheme by attaching an additional mechanism for the signer to request pre-signatures authorized by the system authority for binding the message with the most updated system information. Additionally, we prohibit the system authority from knowing the message being requested for pre-signatures. We only know the message when its pre-signature is transformed into an actual EFDGS signature. And no one but the signer can link the pre-signature and actual EFDGS together. By doing so, we can address the above issues as follows.

- We enable delaying publishing EFDGS signatures. Specifically, even being revoked/unrevoked, the user can publish a valid EFDGS if its corresponding message was issued a pre-signature at the time she was a valid member. For example, secret deals between companies can be delayed from being published due to some confidential policy, police investigations about some cases should not be published early, or ballots may be delayed for some time.
- Signers of EFDGS signatures cannot deny ownership of old EFDGS signatures even after being revoked, since the signatures are everlasting.
- No cost incurs for maintaining a database of signatures.

We realize the above new model by constructing a plausibly post-quantum lattice-based EFDGS based on standard lattices.

2 Notations, Conventions, and Structure of the Paper

Notations and conventions are in Section 2.1. This paper's structure is presented in Section 2.2.

2.1 Notations and Conventions

Define $\bar{b} := 1 - b$ for $b \in \{0,1\}$. We denote by $\mathbb{N}$ the set of positive integers, namely, $\{1,2,3,\dots\}$. For a set S, we denote by $x \xleftarrow{\$} S$ the action of uniformly sampling x from S. For $\ell \in \mathbb{N}$ and $\mathbf{x} = (x_1,\dots,x_n) \in \mathbb{Z}_q^n$, we denote by $\mathsf{bin}_\ell(\mathbf{x}) \in \{0,1\}^{n\ell}$ its binary representation where, for $i \in [n]$, x_i is decomposed into ℓ bits $x_{i,1},\dots,x_{i,\ell}$ such that $x_i = \sum_{j\in[\ell]} 2^{j-1} \cdot x_{i,j}$. All vectors in this paper are column vectors, except otherwise specified. For any two vectors $\mathbf{a}$ and $\mathbf{b}$, we denote by $(\mathbf{a}\|\mathbf{b}) = [\mathbf{a}^\top|\mathbf{b}^\top]^\top$. We write $p_\lambda \leq \mathsf{negl}(\lambda)$ (respectively, $p_\lambda = \mathsf{poly}(\lambda)$) for $(p_\lambda)_\lambda$ to indicate that there is some negligible function $\epsilon(\lambda)$ (respectively, a polynomial $f(\lambda)$) and a value λ_0 such that $p_\lambda \leq \epsilon(\lambda)$ (respectively, $p_\lambda \leq f(\lambda)$) for $\lambda \geq \lambda_0$.

2.2 Structure of the Paper

The technical overview for defining EFDGS and its lattice-based instantiation is presented in Section 3. In Section 4, we define EFDGS. In Section 6, we briefly describe our lattice-based instantiation of EFDGS and defer the detailed description to Appendix B. Some necessary preliminaries for discussing Section 6 are presented in Section 5, while other more detailed preliminaries are presented in Appendix A.

3 Technical Overview

Section 3.1 is for defining EFDGS. Section 3.2 is for an instantiation of EFDGS.

3.1 Defining EFDGS

As introduced, we propose an EFDGS scheme such that whenever an EFDGS signature is generated by some user in the system, it is guaranteed that (i) the signer is a valid user in the system at the time of signing and (ii) the output EFDGS is valid forever whenever being verified by any public party. We define EFDGS as follows.

First, we follow the model of Bootle et al. [18,19] to design an epoch-based system to have two authorities, namely, system authority SA (with public-secret key pair $(\mathsf{pk}_{\mathsf{sa}}, \mathsf{sk}_{\mathsf{sa}})$) for enrolling and revoking users, and tracing authority TA (with public-secret key pair $(\mathsf{pk}_{\mathsf{ta}}, \mathsf{sk}_{\mathsf{ta}})$) for tracing users' identifiers behind signatures. For each epoch, this system has a corresponding system information. Different epochs, hence, have distinct system information.

Removing System Information as Input to Verification. To allow the signatures generated from this system to be valid forever while verification does not require additional input as the most updated system information, we associate to the system an additional mechanism that requires the signer to request a pre-signature to be issued by SA, authorizing the system information for the to-be-signed message. In fact, SA is considered an honest party in most group signature schemes since, otherwise, SA can introduce new malicious members to the system. Leveraging this fact, we hence additionally put the trust in SA for authorizing pre-signature with the most updated epoch.

We call the above process the pre-signature issuance mechanism. Specifically, this mechanism enforces that whenever a user U wants to sign a message M, U first sends M to SA. Without consideration of privacy/anonymity, since SA is honest, SA takes the most updated system information $\mathsf{info}^{\mathrm{cur}}$ honestly and authorizes the concatenated message $(M, \mathsf{info}^{\mathrm{cur}})$ to obtain some pre-signature Γ.

Then, when U issues an EFDGS signature Σ for M, U must show that she knows a pre-signature Γ that is valid with respect to the concatenated message $(M, \mathsf{info}^{\mathrm{cur}})$. In other words, Σ contains a proof showing that there exist $\mathsf{info}^{\mathrm{cur}}$ and Γ such that Γ is a valid pre-signature with respect to $(M, \mathsf{info}^{\tau})$. Hence, if the proof is zero-knowledge, i.e., leaking nothing about info^{τ} and Γ beyond the validity of their existence, the verification does not require info^{τ} as input since the proof underlying Σ guarantees the existence of info^{τ} and Γ.

Notice that, since we bind the message with some epoch τ via SA's issued pre-signature, we allow the signer to keep the message private for a long time, even after she is revoked from the system, before creating the EFDGS signature by using the signing key at epoch τ. This is because the message was first available when requesting a pre-signature. Hence, either publishing immediately or not, it guarantees that the signer can sign at epoch τ as long as she is a valid user of the system. To explain more convincingly, it is similar to a context in which the signer creates a signature at some epoch τ, keeps it secret, and publishes the signature at some point in the future.

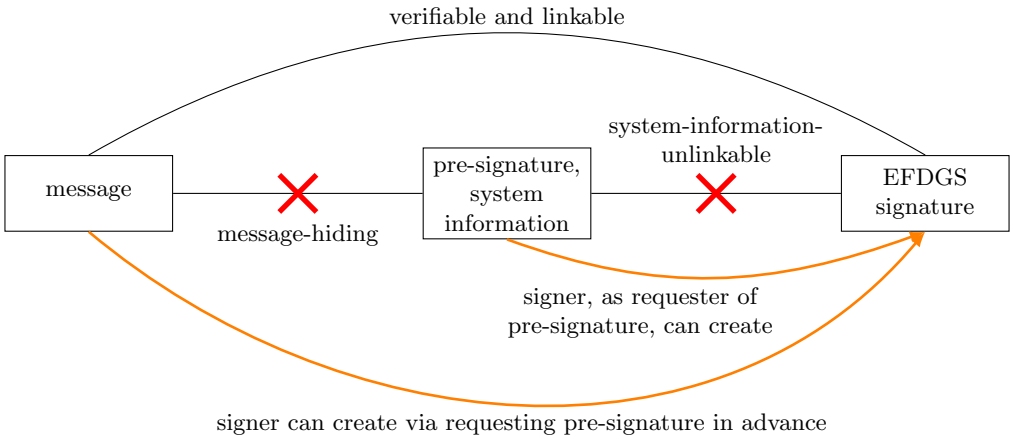

Fig. 1. Graphical illustration of security features of an EFDGS scheme.

Privacy. Nevertheless, the above mechanism does not guarantee the privacy of the signer. As we see that, U may need to send M to SA for authorizing $(M, \mathsf{info}^{\tau})$. Although SA is honest, SA can link the output signature with the signer's identifier since SA knows who requested the pre-signature for M. Sending M directly to SA may allow some eavesdropper to observe the communication between U and SA, compromising the privacy of U. To avoid this issue, we may require SA to issue the pre-signature for $(M, \mathsf{info}^{\mathrm{cur}})$ without allowing SA to see M. This reminds us of the notion of blind signatures [29,14]. In particular, we model the interaction between U and SA such that U obtains a pre-signature Γ (in some form of a blind signature) for $(M, \mathsf{info}^{\tau})$ verifiable by SA's public key $\mathsf{pk}_{\mathsf{sa}}$ while SA does not know anything about M. For any public party, including SA, it must guarantee that a pre-signature and the corresponding transcript for requesting this pre-signature say nothing about the message. Hence, we call this property *message hiding*. On the other hand, when the output EFDGS signatures are created by the signer, we also would like to make sure that no party, except the signer, can link the EFDGS signatures to their respective system information. Hence, we call this property *system information unlinkability*. Figure 1 is a graphical illustration of message hiding and system information unlinkability. Specifically, in the public view, a message and its corresponding EFDGS can be linked together via verification while their respective pre-signature and system information are unlinkable to them.

In modeling privacy, we capture two properties, including the well-known property called anonymity (e.g., see [30,80,12,53]) and the newly introduced properties, i.e., message hiding and system information unlinkability. Anonymity captures the fact that, for any two potential signers, if there is a signature output from one of them, any adversary cannot distinguish who is the actual signer behind the EFDGS signature. Since we incorporate the pre-signature issuance mechanism, in modeling anonymity, we require both signers to obtain the corresponding pre-signatures first before creating the EFDGS signature to challenge the adversary.

Regarding message hiding and system information unlinkability, as said above, since these notions relate to blind signatures [29,14], we require the two potential signers to request pre-signatures from two different epochs for two potential messages and then create the respective EFDGS signatures. However, the order of requesting pre-signatures is shuffled by a secret bit b, to be discussed in detail in Section 4.2. In the end, we compute the output EFDGS signatures for the respective two messages with shuffled pieces of system information (as the order of requesting pre-signatures is random). We ask whether the adversary can link the pairs of messages and corresponding EFDGS signatures to their respective system information. The inability to link requested messages (respectively, EFDGS signatures) to the corresponding system information implies message hiding (respectively, system information unlinkability).

Unforgeability. As U requests the pre-signature Γ from SA to generate EFDGS signature Σ, we must ensure that U is unable to forge the pre-signature from SA in order to create valid Σ. Hence, we define system information unforgeability to prevent the forging of any pre-signature in this context. Consequently, we aim to capture the following five sub-properties in modeling unforgeability. (i) Extractability ensures that TA can obtain an identifier from any valid signature. The traced identifier, in addition, must match the identifier of the signer. (ii) Traceability guarantees that an adversary cannot forge a valid EFDGS signature traced to an identifier not within the group when signing. (iii) Non-frameability guarantees that an adversary cannot forge a valid EFDGS signature that frames an honest user. (iv) System information unforgeability

prevents an adversary from creating a signature with respect to a pre-signature not previously issued by SA. (v) Tracing soundness guarantees that TA cannot "open" a valid signature to two distinct identifiers.

We follow [11,53,68,69] to define the simulated setup of the system, which allows the extraction of the identity as well as the pairs of message and system information behind the pre-signatures, i.e., putting the definition into the theme of simulation-sound extractability (sim-ext) [28]. Then, we can easily capture extractability, traceability, non-frameability, and tracing soundness by simply following [18,19,68]. To capture system information unforgeability, we extract the pairs of messages and requested pre-signatures and form a list dubbed EPL. Then, when the adversary forges a pair of message M and EFDGS signature Σ, we extract the system information from the forged EFDGS signature. If the pair of messages and corresponding extracted system information does not belong to EPL, we consider that $\mathcal{A}$ wins the system information unforgeability.

3.2 A Lattice-Based Instantiation of EFDGS

We realize our notion of EFDGS by constructing a plausibly post-quantum lattice-based EFDGS scheme as follows. Our construction is based on standard lattices. Specifically, we organize the system using the LLNW accumulator [51,52] in which the leaves of the underlying Merkle tree contain the public keys of existing users.

When enrolling or revoking users, SA updates the Merkle tree following the path corresponding to the affected users' identifiers (c.f. [60,61,70,58,71]).

To realize the pre-signature issuance mechanism, we employ the JRS oblivious signing protocol $\Pi_{\mathsf{obl\text{-}sign}}$ from [45] (c.f. Section 5.2 and Appendix A.7). In particular, the system authority SA of our lattice-based EFDGS plays the role of the signer in $\Pi_{\mathsf{obl\text{-}sign}}$. Then, the signatures on committed messages output by SA are the pre-signatures for the requesting user.

When signing a message M, U encrypts her identifier, by some PKE, to obtain a ciphertext ct and shows, in zero knowledge, the existence of system information info^τ and a pre-signature Γ such that (i) U was a valid user who at epoch τ possessed a signing key corresponding to a public key accumulated into the root of the epoch τ Merkle tree, (ii) Γ authorizes the concatenated message (M, info^τ), and (iii) U correctly encrypts her identifier.

To trace signers in case of abuse, tracing authority TA has a decryption key to decrypt the identifier of the signer. Here, we follow [18,19] and require TA to provide a proof of correct decryption.

4 Everlasting Fully Dynamic Group Signatures

We define EFDGS via the syntax and security in Sections 4.1 and 4.2, respectively.

4.1 Syntax

The syntax of an EFDGS scheme is defined in the following Definition 1. Notice that we follow the original model of FDGS from Bootle et al. [18,19] with some modifications. We associate a pre-signature issuance mechanism for issuing pre-signatures by system authority SA via protocol PreSign and algorithm PreVerify for issuing and verifying pre-signatures, respectively. As said in Section 3.1, the pre-signatures aim to bind the "blinded" requested message from the user with the most updated system information. The signing algorithm Sign hence additionally requires an appropriate pre-signature and some auxiliary inputs (e.g., used by the signer to hide the message when requesting pre-signature) to generate the EFDGS signature. The verification algorithm PreVerify no longer requires system information as input.

Definition 1 (Syntax). *An everlasting fully dynamic group signature (EFDGS) scheme is a tuple of interactive protocols and algorithms*

$$\mathcal{EFDGS} = (\mathsf{Setup}, \mathsf{Join}, \mathsf{Update}, \mathsf{PreSign}, \mathsf{PreVerify}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Trace}, \mathsf{Judge}).$$

$\mathsf{Setup}(1^\lambda) \to (\mathsf{pp}, \mathsf{info}^{\mathrm{init}}, \mathsf{reg}^{\mathrm{init}}, \mathsf{SA}(\mathsf{sk}_{\mathsf{sa}}), \mathsf{TA}(\mathsf{sk}_{\mathsf{ta}}))$**:** *On input* 1^λ*,* Setup *specifies the public parameter* pp *containing identifier set* $\mathcal{ID}$*, epoch set* $\mathcal{T}$*, message set* $\mathcal{M}$ *and the group public key* gpk*. Let* $\mathsf{pp} = (\mathcal{ID}, \mathcal{T}, \mathcal{M}, \mathsf{gpk})$ *be implicit in the following algorithms and protocols. This algorithm determines public system information* $\mathsf{info}^{\mathrm{init}}$ *and a public empty registry* $\mathsf{reg}^{\mathrm{init}}$ *to record information of users in the system at the initial epoch* $\mathrm{init} \in \mathcal{T}$*. It also specifies the secret keys* $\mathsf{sk}_{\mathsf{sa}}$ *and* $\mathsf{sk}_{\mathsf{ta}}$ *and distributes them to the system authority* SA *and the tracing authority* TA*, respectively.*

$\mathsf{Join}\ \langle \mathsf{U}, \mathsf{SA}(\mathsf{sk_{sa}})\rangle\ (\mathsf{pp}, \mathsf{info}^{\mathrm{cur}}, \mathsf{reg}^{\mathrm{cur}}, \mathsf{id}) \to (\mathsf{info}^{\mathrm{next}}, \mathsf{reg}^{\mathrm{next}}, \mathsf{pk_{id}}, \mathsf{U}(\mathsf{sk_{id}}))$**:** *This is an interactive protocol between a user* U *and the system authority* SA*, holding secret key* $\mathsf{sk_{sa}}$*, for enrolling* U*, with a proposed identifier* $\mathsf{id} \in \mathcal{ID}$*, to the system at the epoch* $\mathrm{cur} \in \mathcal{T}$*. Eventually, this protocol advances the system to the next epoch* $\mathrm{next} \in \mathcal{T}$ *together with updated system information* $\mathsf{info}^{\mathrm{next}}$ *and registry* $\mathsf{reg}^{\mathrm{next}}$*. This protocol returns a public key* $\mathsf{pk_{id}}$ *while* U *achieves a secret signing key* $\mathsf{sk_{id}}$ *corresponding to* $\mathsf{pk_{id}}$*.*

$\mathsf{Update}(\mathsf{pp}, \mathsf{info}^{\mathrm{cur}}, \mathsf{reg}^{\mathrm{cur}}, \mathsf{sk_{sa}}, \mathcal{S}) \to (\mathsf{info}^{\mathrm{next}}, \mathsf{reg}^{\mathrm{next}})$**:** *This algorithm is run by the system authority* SA *to revoke users whose identifiers belong to the to-be-revoked set of identifiers* $\mathcal{S}$*. On inputs current system information* $\mathsf{info}^{\mathrm{cur}}$*, current registry* $\mathsf{reg}^{\mathrm{cur}}$*,* SA*'s secret key* $\mathsf{sk_{sa}}$*, and the set* $\mathcal{S}$*, this algorithm advances the system to the next epoch* next *and returns the new system information* $\mathsf{info}^{\mathrm{next}}$ *and registry* $\mathsf{reg}^{\mathrm{next}}$*.*

$\mathsf{PreSign}\ \langle \mathsf{U}(M), \mathsf{SA}(\mathsf{sk_{sa}})\rangle\ (\mathsf{pp}, \mathsf{info}^{\mathrm{cur}}) \to (\Gamma, \mathsf{trans}, \mathsf{U}(\mathsf{aux}))$**:** *This is an interactive protocol between a user* U*, on input a message* M*, and the honest system authority* SA*, on input private key* $\mathsf{sk_{sa}}$*, for issuing a pre-signature* Γ *for* U *with respect to* M *and current system information* $\mathsf{info}^{\mathrm{cur}}$ *at the current epoch* cur *(namely, current epoch). If* cur *is not the current epoch, then* SA *aborts. Otherwise, at the end of the interaction,* U *receives a pre-signature* Γ*.* PreSign *also returns a transcript* trans*, for this protocol, and auxiliary secret* aux *to* U*, i.e., containing additional secret generated by* U*.*

$\mathsf{PreVerify}(\mathsf{pp}, \mathsf{info}^{\tau}, M, \mathsf{aux}, \Gamma) \to \{0,1\}$**:** *This deterministic algorithm, on inputs* info^{τ}*, a message* $M \in \mathcal{M}$*, an auxiliary input* aux*, and a pre-signature* Γ*, returns* $b \in \{0,1\}$ *indicating acceptance* $(b = 1)$ *or rejection* $(b = 0)$*.*

$\mathsf{Sign}(\mathsf{pp}, \mathsf{info}^{\tau}, \mathsf{sk_{id}}, M, \mathsf{aux}, \Gamma) \to \Sigma$**:** *This is a PPT algorithm run by a user* U*, on inputs a system information* info^{τ}*, signing key* $\mathsf{sk_{id}}$*, a message* $M \in \mathcal{M}$*, an auxiliary input* aux*, and a pre-signature* Γ*. It returns a signature* Σ*.*

$\mathsf{Verify}(\mathsf{pp}, M, \Sigma) \to \{0,1\}$**:** *This is a deterministic algorithm executable by any public party. On inputs a message* $M \in \mathcal{M}$*, and a signature* Σ*, it returns a bit* $b \in \{0,1\}$ *indicating acceptance* $(b = 1)$ *or rejection* $(b = 0)$*.*

$\mathsf{Trace}(\mathsf{pp}, \mathsf{sk_{ta}}, M, \Sigma) \to (\mathcal{ID} \cup \{\perp\}, \Xi)$**:** *This algorithm is run by the tracing authority* TA*. On input* TA*'s private key* $\mathsf{sk_{ta}}$*, a message* $M \in \mathcal{M}$*, and a signature* Σ*, this algorithm returns an identifier* $\mathsf{id}' \in \mathcal{ID}$ *or* $\perp$ *(indicating unability to trace) and a proof* Ξ*.*

$\mathsf{Judge}(\mathsf{pp}, M, \Sigma, \mathsf{id}, \Xi) \to \{0,1\}$**:** *This is a deterministic algorithm executable by any public party. On input a message signature pair* (M, Σ)*,* TA*'s claimed identifier* id *and the proof* Ξ*, outputs verdict acceptance* $(b = 1)$ *if* id *is the correct identifier of signer behind* Σ*, and rejection* $(b = 0)$*, otherwise.*

Remark 2. Diverting from previous state-of-the-art privacy-preserving signatures [18,60,61,19,58], the algorithms Verify, Trace, and Judge no longer require associated system information to decide whether the EFDGS signature Σ is valid.

Remark 3. In PreSign, the user U requests a pre-signature for the concatenated message $(M, \mathsf{info}^{\mathrm{cur}})$. Here, M should not be leaked to SA. Therefore, U may need to generate additional private inputs, captured by aux, when communicating with SA to protect the privacy of M.

4.2 Security

We consider a stronger version of SA, whom an adversary $\mathcal{A}$ may corrupt in a less restricted fashion. For example, in the definition of privacy, $\mathcal{A}$ may instruct SA to generate outdated pre-signatures. In the following, we model an adversary's ability to influence SA through a set of oracles. Aside from these oracle calls, we assume that SA performs semi-honestly. This is captured by protocol $\mathsf{PreSign_{aug}}$, where "aug" stands for "augmented". In practice, accessing this protocol is restricted since a revoked user can create EFDGS signatures with the old signing keys. To define the security for EFDGS, we additionally define the following algorithms for an EFDGS scheme in Definition 4 for convenience.

Definition 4 (Additional Algorithm and Protocol for Security).

$\mathsf{IsActive}(\mathsf{pp}, \mathsf{info}^{\tau}, \mathsf{id}) \to \{0,1\}$**:** *Return* 1 *only if identifier* id *is in the system at the epoch* τ *whose system information is* info^{τ}*. Otherwise, it returns* 0*.*

$\mathsf{PreSign_{aug}}\ \langle \mathsf{U}(M), \mathsf{SA}(\mathsf{sk_{sa}})\rangle\ (\mathsf{pp}, \mathsf{info}^{\tau}) \to (\Gamma, \mathsf{trans}, \mathsf{U}(\mathsf{aux}))$**:** *This protocol is an augmented version of* PreSign *in Definition 1. Its execution is the same as* PreSign *except that* info^{τ} *is not restricted to be the current system information.*

Correctness. The correctness of EFDGS captures the following conditions.

- The updated system information $\mathsf{info}^{\mathrm{next}}$ and registry $\mathsf{reg}^{\mathrm{next}}$ computed from honest executions of either Join and Update are always correct.
- Any honest signer, who is active with respect to info^{τ} for any epoch τ, can always generate valid EFDGS signatures if the corresponding pre-signatures are requested and run honestly via PreSign.

The correctness of EFDGS is formally defined in the following Definition 5.

Definition 5 (Correctness of EFDGS). *Let* pp, $\mathsf{info}^{\mathrm{init}}$, $\mathsf{reg}^{\mathrm{init}}$, $\mathsf{sk}_{\mathsf{sa}}$, *and* $\mathsf{sk}_{\mathsf{ta}}$ *be outputs from honestly running* $\mathsf{Setup}(1^{\lambda})$. *The correctness of an EFDGS scheme captures the following sub-properties.*

1. *If* U *with identifier* id *is enrolled to the system via an honest execution of* Join, *i.e., both* U *and* SA *are honest, to obtain key pair* $(\mathsf{pk}_{\mathsf{id}}, \mathsf{sk}_{\mathsf{id}})$, *then (i) the updated system information* $\mathsf{info}^{\mathrm{next}}$ *and registry* $\mathsf{reg}^{\mathrm{next}}$ *are correctly computed, i.e., non-zero entries in* $\mathsf{reg}^{\mathrm{next}}$ *are those of existing users and the newly enrolled one, and (ii) at any epoch* τ, *if* $\mathsf{IsActive}(\mathsf{pp}, \mathsf{info}^{\tau}, \mathsf{id}) = 1$, *then*

$$\Pr\left[\begin{array}{l|l} \mathsf{PreVerify}(\mathsf{pp}, \mathsf{info}^{\tau}, M, \mathsf{aux}, \Gamma) = 1 & (\Gamma, \mathsf{trans}, \mathsf{U}(\mathsf{aux})) \leftarrow \mathsf{PreSign}_{\mathsf{aug}} \\ \wedge \mathsf{Verify}(\mathsf{pp}, M, \Sigma) = 1 & \quad \langle \mathsf{U}(M), \mathsf{SA}(\mathsf{sk}_{\mathsf{sa}}) \rangle (\mathsf{pp}, \mathsf{info}^{\tau}), \\ \wedge \mathsf{id}' = \mathsf{id} & \Sigma \leftarrow \mathsf{Sign}(\mathsf{pp}, \mathsf{info}^{\tau}, \mathsf{sk}_{\mathsf{id}}, M, \mathsf{aux}, \Gamma), \\ \wedge \mathsf{Judge}(\mathsf{pp}, M, \Sigma, \mathsf{id}', \Xi) = 1 & (\mathsf{id}', \Xi) \leftarrow \mathsf{Trace}(\mathsf{pp}, \mathsf{sk}_{\mathsf{ta}}, M, \Sigma) \end{array}\right]$$

   is overwhelming, i.e., at least $1 - \mathsf{negl}(\lambda)$, *if* SA *is honest.*
2. *For any execution of* Update *executed by an honest* SA *at epoch* cur, *the next epoch* next *has the updated system information* $\mathsf{info}^{\mathrm{next}}$ *and registry* $\mathsf{reg}^{\mathrm{next}}$ *correctly computed, i.e., non-zero entries in* $\mathsf{reg}^{\mathrm{next}}$ *are of unrevoked users.*

Oracles. Following previous privacy-preserving signatures (e.g., [18,19,68]), before defining the security of EFDGS, including unforgeability and privacy, we define necessary oracles in the following Definition 6.

Definition 6 (Oracles). *We define the following oracles.*

OSA**:** *This oracle allows* $\mathcal{A}$ *to obtain* SA*'s private key* $\mathsf{sk}_{\mathsf{sa}}$. *With* $\mathsf{sk}_{\mathsf{sa}}$, $\mathcal{A}$ *can secretly introduce new users to the system without the notice of other parties in the system. We define a variable* $\mathsf{acc\text{-}sk}_{\mathsf{sa}} \in \{0, 1\}$ *to indicate whether* $\mathcal{A}$ *has accessed to this oracle* OSA.

$\mathsf{OCrptU}(\mathsf{id})$**:** *This oracle allows* $\mathcal{A}$ *to introduce new corrupt user with identifier* id *at the current system information* $\mathsf{info}^{\mathrm{cur}}$, *whose signing keys are known to* $\mathcal{A}$. *Accessing this oracle does not require* $\mathcal{A}$ *to possess* $\mathsf{sk}_{\mathsf{sa}}$. *Every party must be informed of changes in users' identifiers via this oracle. This is the difference from* OSA, *since, by accessing* $\mathsf{sk}_{\mathsf{sa}}$ *via* OSA, $\mathcal{A}$ *can secretly introduce new users without notice from any other party. This oracle maintains a list* CL *(say, the corrupt list) such that any identifier* id *introduced via this oracle is then inserted into* CL.

$\mathsf{OCrptSA}(\mathsf{id})$**:** *This oracle allows* $\mathcal{A}$ *to instruct* SA *to (i) introduce new honest members to the system at the current system information* $\mathsf{info}^{\mathrm{cur}}$, *and (ii) authorize pre-signature requests (via protocols* PreSign *and* $\mathsf{PreSign}_{\mathsf{aug}}$ *in Definitions 1 and 4, respectively) from users. Accessing this oracle does not require* $\mathcal{A}$ *to possess* $\mathsf{sk}_{\mathsf{sa}}$. *Every party must be informed of changes in users' identifiers via this oracle. This oracle maintains a list* HL *(say, honest list) such that any* id *introduced via this oracle is then inserted into* HL.

$\mathsf{OSign}(\mathsf{id}, \mathsf{info}^{\tau}, M, \mathsf{aux}, \Gamma) \to \Sigma$**:** *This oracle allows* $\mathcal{A}$ *to request an EFDGS signature* Σ *from signing key* $\mathsf{sk}_{\mathsf{id}}$ *of an honest user* $\mathsf{id} \in \mathsf{HL}$, *corresponding system information* info, *message* M, *auxiliary input* aux, *and a pre-signature* Γ. *This oracle aborts and returns* $\perp$ *if* $\mathsf{id} \notin \mathsf{HL}$ *or* id *is not active in the system at epoch* τ. *Otherwise, it runs* $\mathsf{Sign}(\mathsf{pp}, \mathsf{info}^{\tau}, \mathsf{sk}_{\mathsf{id}}, M, \mathsf{aux}, \Gamma)$ *to obtain and return EFDGS signature* Σ *to* $\mathcal{A}$. *This oracle also maintains a set* SIGS *of message-signature pairs such that, for every invocation to this oracle with respect to input* M *and output* Σ, *it inserts* (M, Σ) *to* SIGS.

$\mathsf{OTrace}(M, \Sigma) \to \mathcal{ID} \cup \{\perp\}$**:** *This oracle allows* $\mathcal{A}$ *to see the identifier of signer behind EFDGS signature* Σ. *It first checks whether* $\mathsf{Verify}(\mathsf{pp}, M, \Sigma) = 1$. *If so, it returns* id *from computing* $(\mathsf{id}, \Xi) \leftarrow \mathsf{Trace}(\mathsf{sk}_{\mathsf{ta}}, M, \Sigma)$.

Privacy. In an EFDGS system, privacy captures the following sub-properties.

- *Anonymity.* The signer's identifier from an adversary who tries to extract either the partial or entire identifier of the signer. This property is well-known for previous privacy-preserving signatures as well as group signatures (e.g., see [30,80,12,53]).

– *Message Hiding.* This property prevents an adversary who corrupts SA from linking the issued pre-signature to the requested message, supposed to be hidden from SA, via $\mathsf{PreSign}$ and $\mathsf{PreSign}_{\mathsf{aug}}$, and any other party. Although, in the beginning, we assume that SA is honest and always issues pre-signatures authorizing the requested messages with the suitable system information, SA may be curious about the (partial) information about the requested message. This property further implies that, except the signer, no other party can link the output EFDGS signature and its corresponding pre-signature. This is because the message (e.g., M) and the EFDGS signature (e.g., Σ) are linked together via Verify. Hence, if the corresponding pre-signature Γ is linked to Σ, it is also linked to M, implying a violation of this property.
– *System Information Unlinkability.* Any EFDGS message-signature pair does not reveal the corresponding system information. In other words, SA, via $\mathsf{PreSign}$ and $\mathsf{PreSign}_{\mathsf{aug}}$, and any other party, except the signer, cannot link a valid EFDGS message-signature pair to its system information with respect to the epoch when requesting the corresponding pre-signature. Figure 1 is a graphical illustration of the message hiding and system information unlinkability.

We now discuss in detail our approach to model privacy.

Anonymity. Originated from Bootle et al. [18], state-of-the-art fully dynamic privacy-preserving signatures [18,60,61,19,58] model anonymity by challenging the adversary to distinguish the output signature from the two chosen identifiers at the same epoch. However, in EFDGS, we do not require verifying the EFDGS signatures with the associated system information. Hence, the privacy experiment in an EFDGS scheme does not enforce choosing two identifiers at the same epoch for distinguishing. Specifically, as in previous results [18,19], different epochs would help $\mathcal{A}$ to distinguish since, if so, $\mathcal{A}$ knows the epoch of the challenged signature for verifying and hence knows the identifier corresponding to this epoch. However, for EFDGS, we allow $\mathcal{A}$ to choose two identifiers with respect to two potentially different epochs for requesting the challenged EFDGS signature. On the other hand, as later we will employ NIZK proof systems for realizing the EFDGS signatures, zero knowledge guarantees that, even though $\mathcal{A}$ knows all potential witnesses supporting constructing the proofs, there is no way for $\mathcal{A}$ to decide which witness was employed to create the proof. In fact, this is the notion of witness indistinguishability [36], which is implied by zero knowledge. Hence, to formalize anonymity, we allow the adversary $\mathcal{A}$ to gain access to any signing key $\mathsf{sk}_{\mathsf{id}}$ of any user id. This says that even though $\mathcal{A}$ knows all signing keys of the users, $\mathcal{A}$ cannot distinguish the signer behind the output EFDGS signature from any two potential chosen users (in potentially different epochs). To capture anonymity, we follow [60,58] to define oracle $\mathsf{OSigChal}_b$ (see Definition 7), for $b \in \{0,1\}$, that allows $\mathcal{A}$ to choose a message M and two tuples $\{(\mathsf{id}^{(i)}, \mathsf{info}^{\tau^{(i)}}, \mathsf{aux}^{(i)}, \Gamma^{(i)})\}_{i\in\{0,1\}}$. For $i \in \{0,1\}$, $\mathsf{aux}^{(i)}$ is the auxiliary input that was used to request pre-signature $\Gamma^{(i)}$ with respect to $\mathsf{info}^{\tau^{(i)}}$. Then, by Definition 7, $\mathsf{OSigChal}_b$ generates the EFDGS signature Σ based on $\mathsf{id}^{(b)}$ and $\mathsf{info}^{\tau^{(b)}}$.

Message Hiding and System Information Unlinkability. As said in Section 3.1, these notions relate to blind signatures [29,14]. In Definition 7, with an implicit bit b, we define oracle $\mathsf{OPreChal}_b$ receiving $(\mathsf{id}^{(0)}, M^{(0)}, \mathsf{id}^{(1)}, M^{(1)})$ and a tuple $(\mathsf{info}^{\tau^{(0)}}, \mathsf{info}^{\tau^{(1)}})$ from $\mathcal{A}$. This oracle first samples the auxiliary inputs and then requests pre-signatures, via $\mathsf{PreSign}_{\mathsf{aug}}$, in a random order of $M^{(0)}$ and $M^{(1)}$. In particular, for a private bit b unknown to $\mathcal{A}$, we use $M^{(i\oplus b)}$ to request pre-signature with respect to $\mathsf{info}^{\tau^{(i)}}$ for $i \in \{0,1\}$. Here, $\oplus$ denotes the XOR operation. Then, $\Gamma^{(i)}$ and $\mathsf{aux}^{(i\oplus b)}$ are returned. Notice that $\mathcal{A}$ cannot know $\mathsf{aux}^{(i\oplus b)}$. Making such a random order of messages helps hide the message and hence makes the system information unlinkable to the message. Hence, even $\mathcal{A}$ corrupts SA, $\mathcal{A}$ cannot know for which message $\mathcal{A}$ issues the pre-signature. Then, $\mathsf{OPreChal}_b$ returns $\{\Sigma^{(i)}\}_{i\in\{0,1\}}$ to $\mathcal{A}$, where $\Sigma^{(i\oplus b)}$ is computed from $\mathsf{Sign}(\mathsf{pp}, \mathsf{info}^{\tau^{(i)}}, \mathsf{sk}_{\mathsf{id}^{(i\oplus b)}}, M^{(i\oplus b)}, \mathsf{aux}^{(i\oplus b)}, \Gamma^{(i)})$. Then, $\mathcal{A}$ is challenged to guess b. If $\mathcal{A}$ cannot guess b, system information unlinkability is guaranteed. Hence, in summary, $\mathcal{A}$ can break both message hiding and system information unlinkability if $\mathcal{A}$ has a non-negligible advantage in guessing b.

Regarding the case in which SA and TA collude, i.e., $\mathcal{A}$ can corrupt both SA and TA, for seeking some advantage from TA's tracing capability. We allow $\mathcal{A}$ to access $\mathsf{sk}_{\mathsf{ta}}$. In fact, $\mathsf{sk}_{\mathsf{ta}}$ is only used for tracing identifiers from EFDGS signatures. Hence, in the pre-signing process, having $\mathsf{sk}_{\mathsf{ta}}$ would give no advantage in compromising the privacy of M. When an EFDGS signature is created, even though having $\mathsf{sk}_{\mathsf{ta}}$ can help trace the identifier, $\mathsf{sk}_{\mathsf{ta}}$ does not help link the EFDGS signature, the respective message, and the identifier to the corresponding system information employed for signing.

See Figure 2 for a graphical illustration of how $\mathsf{OPreChal}_b$ works.

Ability of $\mathcal{A}$ in Modeling Privacy. Regarding the computational power of $\mathcal{A}$, as EFDGS allows TA to trace the identifiers of the signers behind the EFDGS signatures, in many constructions of group signatures

[50,60,72,74], this tracing mechanism is usually realized by PKE encryptions which are only secure against PPT adversaries. Hence, in modeling anonymity, we restrict $\mathcal{A}$ to be PPT. However, having $\mathsf{sk_{ta}}$ would not help $\mathcal{A}$ break message hiding and system information unlinkability. We assume $\mathcal{A}$ to be computationally unbounded.

We allow $\mathcal{A}$ to access all oracles, except $\mathsf{sk_{ta}}$ and OTrace for the case of anonymity. Specifically, in the case of anonymity, if $\mathcal{A}$ possesses $\mathsf{sk_{ta}}$, $\mathcal{A}$ can easily trace the identifiers behind the signatures, which gives $\mathcal{A}$ some advantage in guessing the identifier from the output of $\mathsf{OSigChal}_b$. On the other hand, in our consideration, since OTrace is related to the decryption of some PKE, it is hence associated with the notions IND-CPA, IND-CCA, and IND-CCA2. Therefore, for simplicity, we only focus on IND-CPA by not allowing $\mathcal{A}$ to have access to OTrace. This is because IND-CCA and IND-CCA2 require more complicated mechanisms or primitives in constructing, e.g., [67,38,39]. Hence, we omit the case of IND-CCA and IND-CCA2 in this result.

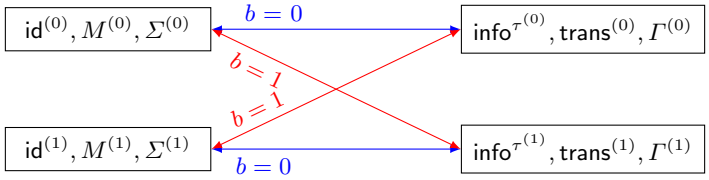

Fig. 2. Graphical illustration for $\mathsf{OPreChal}_b$.

Experiments for and Definition of Privacy. The above discussion is summarized in Experiments $\mathsf{E}_{\mathcal{A}}^{\mathsf{anon}\text{-}b}(\lambda)$ and $\mathsf{E}_{\mathcal{A}}^{\mathsf{pre\text{-}unlk}\text{-}b}(\lambda)$ in Figure 3. Here, $\mathsf{E}_{\mathcal{A}}^{\mathsf{anon}\text{-}b}(\lambda)$ is for modeling anonymity while $\mathsf{E}_{\mathcal{A}}^{\mathsf{pre\text{-}unlk}\text{-}b}(\lambda)$ is for modeling both message hiding and system information unlinkability. At the end of each experiment, it requires $\mathcal{A}$ to return a bit $b' \in \{0,1\}$. An EFDGS scheme satisfies privacy if $\mathcal{A}$ can correctly guess $b' = b$ with overwhelming probability in neither $\mathsf{E}_{\mathcal{A}}^{\mathsf{anon}\text{-}b}(\lambda)$ nor $\mathsf{E}_{\mathcal{A}}^{\mathsf{pre\text{-}unlk}\text{-}b}(\lambda)$. The formal definition of privacy is presented in Definition 8.

Definition 7 (Oracles for Privacy). *We define the following oracles.*

$\mathsf{OPreChal}_b((\mathsf{id}^{(0)}, M^{(0)}, \mathsf{id}^{(1)}, M^{(1)}), (\mathsf{info}^{\tau^{(0)}}, \mathsf{info}^{\tau^{(1)}})) \to \{\Sigma^{(i)}\}_{i\in\{0,1\}}$ **or** $\perp$**:**

This oracle is called only once. If $\mathsf{info}^{\tau^{(0)}}$ *or* $\mathsf{info}^{\tau^{(1)}}$ *or both have never existed, or exist* $i, j \in \{0,1\}$ *satisfying* $\mathsf{IsActive}(\mathsf{pp}, \mathsf{info}^{\tau^{(i)}}, \mathsf{id}^{(j)}) = 0$*, it returns* $\perp$*. Otherwise, it works as follows.*

1. *For* $i \in \{0,1\}$*, obtain* $(\Gamma^{(i)}, \mathsf{trans}^{(i)}, \mathsf{U}(\mathsf{aux}^{(i\oplus b)}))$ *by invoking the protocol*

$$\mathsf{PreSign}_{\mathsf{aug}}\left\langle \mathsf{U}(M^{(i\oplus b)}), \mathsf{SA}(\mathsf{sk_{sa}}) \right\rangle (\mathsf{pp}, \mathsf{info}^{\tau^{(i)}}).$$

   Notice that, for $i \in \{0,1\}$*,* $(\Gamma^{(i)}, \mathsf{trans}^{(i)})$ *is public.*
2. $\Sigma^{(i\oplus b)} \leftarrow \mathsf{Sign}(\mathsf{pp}, \mathsf{info}^{\tau^{(i)}}, \mathsf{sk}_{\mathsf{id}^{(i\oplus b)}}, M^{(i\oplus b)}, \mathsf{aux}^{(i\oplus b)}, \Gamma^{(i)})$ $\forall i \in \{0,1\}$.
3. *Return* $\{\Sigma^{(i)}\}_{i\in\{0,1\}}$.

$\mathsf{OSigChal}_b(M, (\mathsf{id}^{(0)}, \mathsf{info}^{\tau^{(0)}}, \mathsf{aux}^{(0)}, \Gamma^{(0)}), (\mathsf{id}^{(1)}, \mathsf{info}^{\tau^{(1)}}, \mathsf{aux}^{(1)}, \Gamma^{(1)})) \to \Sigma$ **or** $\perp$**:** *This oracle is called only once. If both* $\mathsf{info}^{\tau^{(0)}}$ *and* $\mathsf{info}^{\tau^{(1)}}$ *have existed,* $\mathsf{IsActive}(\mathsf{pp}, \mathsf{info}^{\tau^{(i)}}, \mathsf{id}^{(i)}) = 1$ *for* $i \in \{0,1\}$*, message* $M \in \mathcal{M}$*, and it holds that* $\mathsf{PreVerify}(\mathsf{pp}, \mathsf{info}^{\tau^{(i)}}, M, \mathsf{aux}^{(i)}, \Gamma^{(i)}) = 1$ *for* $i \in \{0,1\}$*, then return* Σ *obtained from running* $\mathsf{Sign}(\mathsf{pp}, \mathsf{info}^{\tau^{(b)}}, \mathsf{sk}_{\mathsf{id}^{(b)}}, M, \mathsf{aux}^{(b)}, \Gamma^{(b)})$*. Otherwise, return* $\perp$*.*

Definition 8 (Privacy). *An EFDGS scheme satisfies privacy if*

$$\left|\Pr[\mathsf{E}_{\mathcal{A}}^{\mathsf{anon}\text{-}0}(\lambda) = 1] - \Pr[\mathsf{E}_{\mathcal{A}}^{\mathsf{anon}\text{-}1}(\lambda) = 1]\right| \leq \mathsf{negl}(\lambda) \qquad \forall \textit{PPT } \mathcal{A}, \textit{ and}$$
$$\left|\Pr[\mathsf{E}_{\mathcal{A}}^{\mathsf{pre\text{-}unlk}\text{-}0}(\lambda) = 1] - \Pr[\mathsf{E}_{\mathcal{A}}^{\mathsf{pre\text{-}unlk}\text{-}1}(\lambda) = 1]\right| \leq \mathsf{negl}(\lambda) \quad \forall \textit{(PPT) } \mathcal{A}.$$

where $\mathsf{E}_{\mathcal{A}}^{\mathsf{anon}\text{-}b}(\lambda)$ *and* $\mathsf{E}_{\mathcal{A}}^{\mathsf{pre\text{-}unlk}\text{-}b}(\lambda)$*, for* $b \in \{0,1\}$*, are in Figure 3.*

$(\mathsf{pp}, \mathsf{info}^{\mathrm{init}}, \mathsf{sk}_{\mathsf{sa}}, \mathsf{sk}_{\mathsf{ta}}) \leftarrow \mathsf{Setup}(1^\lambda)$, $\mathsf{SIGS} := \emptyset$.
$\mathsf{acc\text{-}sk}_{\mathsf{sa}} := 0, \mathsf{CL} := \emptyset, \mathsf{HL} := \emptyset, \mathsf{PL} := \emptyset$.
$b' \leftarrow \mathcal{A}^{\mathsf{OSA},\mathsf{OCrptU},\mathsf{OCrptSA},\mathsf{OSign},\mathsf{OSigChal}_b,\mathsf{OPreChal}_b}(\mathsf{pp}, \mathsf{info}^{\mathrm{init}}, \mathsf{sk}_{\mathsf{ta}})$.

Fig. 3. $\mathsf{E}_{\mathcal{A}}^{\text{anon-}b}(\lambda)$ (respectively, $\mathsf{E}_{\mathcal{A}}^{\text{pre-unlk-}b}(\lambda)$) including texts in light gray (respectively, dark gray) background and excluding texts with dark gray (respectively, light gray) background.

Unforgeability. Recall from FDGS [18,19], this original formalization of FDGS captures traceability and non-frameability. Traceability prevents a malicious signer from creating a valid FDGS signature, at some epoch τ, that is traced to an identifier (including $\perp$) outside the group of users at this respective epoch τ. Non-frameability prevents an adversary $\mathcal{A}$ from outputting an FDGS signature traced to a user whose signing key is unknown to $\mathcal{A}$. In EFDGS, we follow the formalization of Multimodal Private Signatures [68] to define unforgeability, capturing sub-properties including traceability (aka type-1 unforgeability) and non-frameability (aka type-2 unforgeability).

Moreover, considering SA to be an honest-but-curious party, we also model unforgeability to prohibit the adversary from generating EFDGS signatures with respect to the past system information (not the current one). This sub-property is called *system information unforgeability*.

To this end, we follow previous results [11,53,68,69] to put the security of the system into the simulation-sound extractability (sim-ext) approach [28] by defining additional algorithms in the following Definition 9.

Definition 9 (Additional Algorithms for Sim-Ext).

$\mathsf{SimSetup}(1^\lambda) \rightarrow (\mathsf{pp}, \mathsf{info}^{\mathrm{init}}, \mathsf{SA}(\mathsf{sk}_{\mathsf{sa}}), \mathsf{TA}(\mathsf{sk}_{\mathsf{ta}}), \mathsf{td}_{\mathsf{ext}})$**:** *This algorithm is the simulated setup algorithm that returns similar outputs as of* $\mathsf{Setup}(1^\lambda)$*. However, it also returns an additional extraction trapdoor* $\mathsf{td}_{\mathsf{ext}}$*.*

$\mathsf{ExtPreSig}(\mathsf{pp}, \mathsf{td}_{\mathsf{ext}}, \mathsf{info}^\tau, \Gamma, \mathsf{trans}) \rightarrow (\widetilde{M}, \widetilde{\mathsf{aux}})$**:** *This is the extraction algorithm of the pre-signing process from* PreSign*. On inputs extraction trapdoor* $\mathsf{td}_{\mathsf{ext}}$*, the pre-signature* Γ*, and transcript* trans *(obtained from running* ExtPreSig*), it returns a message* $\widetilde{M}$*, and an auxiliary input* $\widetilde{\mathsf{aux}}$ *subjected to pre-signature verification* $\mathsf{PreVerify}(\mathsf{pp}, \mathsf{info}^\tau, \widetilde{M}, \widetilde{\mathsf{aux}}, \Gamma) = 1$*.*

$\mathsf{ExtSig}(\mathsf{pp}, \mathsf{td}_{\mathsf{ext}}, M, \Sigma) \rightarrow (\widetilde{\mathsf{id}}, \widetilde{\mathsf{info}}^\tau)$**:** *This is for extracting from EFDGS signature* Σ*. On inputs extraction trapdoor* $\mathsf{td}_{\mathsf{ext}}$*, message* M *and EFDGS signature* Σ*, it returns an identifier* $\widetilde{\mathsf{id}}$ *and a system information* $\widetilde{\mathsf{info}}^\tau$*.*

Correctness and Setup Indistinguishability for Sim-Ext. With algorithms above, we can put the system into a real or a simulated setup by running Setup or SimSetup, respectively. As from [68], we require that

- The correctness in Definition 5 also holds with respect to the simulated setup.
- The real and simulated setups are indistinguishable by any PPT $\mathcal{A}$.

Formalizing Unforgeability. With the above algorithms and properties for the simulated setup, we can turn back to defining the unforgeability of the system. In particular, by putting the system into simulated setup, we can have the power to extract the message M' and system information $\mathsf{info}^{\tau'}$ from the transcript of the pre-signing process via ExtPreSig, and the identifier id' and system information $\mathsf{info}^{\tau'}$ from the EFDGS signature Σ via ExtSig thanks to the extraction trapdoor $\mathsf{td}_{\mathsf{ext}}$. Then, we define unforgeability to capture the sub-properties as follows.

- *Extractability.* Since we introduced the simulated setup, as from [68], we must guarantee that the extracted identifier (from ExtSig) must match the traced one (from Trace) for an EFDGS signature. This sub-property is called *extractability*. Here, we allow $\mathcal{A}$ to access TA's private key $\mathsf{sk}_{\mathsf{ta}}$ and all oracles in Definition 6. $\mathcal{A}$ is considered to break this property if $\mathcal{A}$ can provide a valid EFDGS signature whose extracted and traced identifiers are distinct.
- *System Information Unforgeability.* This property prohibits $\mathcal{A}$ from creating an EFDGS signature Σ whose the corresponding message M has never been invoked via PreSign to receive an authorized system information info (by SA) equal to the extracted system information $\mathsf{info}^{\tau'}$ from Σ (via ExtSig). To this end, we allow $\mathcal{A}$ to access TA's private key $\mathsf{sk}_{\mathsf{ta}}$ and all oracles, except OSA, in Definition 6. We will

explain why accessing OSA is prohibited below after introducing the lists PL and EPL. To capture system information unforgeability, we use an additional list, dubbed PL, to record all pre-signatures and the corresponding transcripts (via either $\mathsf{PreSign}$ or $\mathsf{PreSign}_{\mathsf{aug}}$) issued by SA. For each invocation to either $\mathsf{PreSign}$ or $\mathsf{PreSign}_{\mathsf{aug}}$, the triple $(\mathsf{info}^\tau, \Gamma, \mathsf{trans})$, including the request system information info^τ ($\mathsf{info}^{\mathrm{cur}}$ if invoked via $\mathsf{PreSign}$) and outputs Γ and trans, is then inserted into PL. In the end, we convert the list PL into another list EPL via $\mathsf{ExtPreSig}$, containing the extracted messages and system information as follows.

$$\mathsf{EPL} = \left\{ (M', \mathsf{info}^\tau) \;\middle|\; \begin{array}{l} (\mathsf{info}^\tau, \Gamma, \mathsf{trans}) \in \mathsf{PL} \\ \wedge (\widetilde{M}, \widetilde{\mathsf{aux}}) \leftarrow \mathsf{ExtPreSig}(\mathsf{pp}, \mathsf{td}_{\mathsf{ext}}, \mathsf{info}^\tau, \Gamma, \mathsf{trans}) \end{array} \right\}.$$

$\mathcal{A}$ is considered breaking the system information unforgeability if $\mathcal{A}$ can output a message M and a corresponding valid EFDGS signature Σ such that, via invoking $\mathsf{ExtSig}(\mathsf{pp}, \mathsf{td}_{\mathsf{ext}}, M, \Sigma) \rightarrow (\widetilde{\mathsf{id}}, \widetilde{\mathsf{info}^\tau})$, the pair $(M, \widetilde{\mathsf{info}^\tau})$ does not belong to EPL, i.e., $\mathcal{A}$ is able to create EFDGS signatures without prior authorization from SA.

We now explain why accessing OSA is prohibited. Notice that, having $\mathsf{sk}_{\mathsf{sa}}$, $\mathcal{A}$ can secretly authorize pre-signatures without notice from any party. Hence, the list EPL cannot capture the related pairs of message and system information that $\mathcal{A}$ secretly authorized by issuing pre-signatures.

- *Traceability.* As from [18,19,60,61], this property prevents $\mathcal{A}$ from creating a pair (M, Σ) of message and a valid EFDGS signature Σ that can be traced (via Trace) to an id' that is not in the system with respect to the extracted system formation, is unable to generate a proof of tracing, or is not a valid identifier. Here, we allow $\mathcal{A}$ to have access to TA's private key $\mathsf{sk}_{\mathsf{ta}}$ and all oracles in Definition 6 except OSA (for accessing SA's private key $\mathsf{sk}_{\mathsf{sa}}$). Accessing $\mathsf{sk}_{\mathsf{sa}}$ allows $\mathcal{A}$ to secretly introduce new corrupt users without being recorded in CL.
- *Non-Frameability.* As from [18,19,60,61], this property prevents $\mathcal{A}$ to create a pair (M, Σ) of message and a valid EFDGS signature Σ that can be traced (via Trace) to id' in the honest list HL. We allow $\mathcal{A}$ to have access to SA's private key $\mathsf{sk}_{\mathsf{sa}}$, TA's private key $\mathsf{sk}_{\mathsf{ta}}$ and all oracles in Definition 6. Here, accessing $\mathsf{sk}_{\mathsf{sa}}$ is allowed because, even if $\mathcal{A}$ can introduce new secret users to the system without any record, they are impossible to be the ones in HL who are certainly recorded. Then, $\mathcal{A}$ is considered breaking the traceability if $\mathcal{A}$ can return a pair (M, Σ) such that Σ is a valid EFDGS signature with respect to M and the traced $\mathsf{id}' \in \mathsf{HL}$ is active with respect to the extracted system information.
- *Tracing Soundness.* As from [18,19,60,61], this says that $\mathcal{A}$ is unable to produce an EFDGS signature that is traced to two distinct identifiers even $\mathcal{A}$ has accesses to all oracles and secret keys of SA and TA.

To make the above four properties captured by a unified experiment, we observe that $\mathcal{A}$ has access to all possible oracles in Definition 6 except that, for system information unforgeability and traceability, $\mathcal{A}$ is prohibited from accessing $\mathsf{sk}_{\mathsf{sa}}$ (via OSA). To encounter this circumstance, recall that, in Definition 6, we introduced the variable $\mathsf{acc\text{-}sk}_{\mathsf{sa}} \in \{0, 1\}$ indicating whether $\mathcal{A}$ queried oracle OSA. Hence, $\mathcal{A}$ is considered to win system information unforgeability or traceability if $\mathcal{A}$ has never queried OSA, i.e., by checking whether $\mathsf{acc\text{-}sk}_{\mathsf{sa}} = 0$.

For unforgeability, we use $\mathsf{E}_{\mathcal{A}}^{\mathsf{unf\text{-}wo\text{-}tracesnd}}(\lambda)$ and $\mathsf{E}_{\mathcal{A}}^{\mathsf{tracesnd}}(\lambda)$ in Figure 4 to capture all mentioned sub-properties for tracing soundness. Here, $\mathsf{E}_{\mathcal{A}}^{\mathsf{unf\text{-}wo\text{-}tracesnd}}(\lambda)$ captures extractability, system information unforgeability, traceability, and non-frameability while $\mathsf{E}_{\mathcal{A}}^{\mathsf{tracesnd}}(\lambda)$ captures tracing soundness by challenging $\mathcal{A}$ to return 1 in either $\mathsf{E}_{\mathcal{A}}^{\mathsf{unf\text{-}wo\text{-}tracesnd}}(\lambda)$ or $\mathsf{E}_{\mathcal{A}}^{\mathsf{tracesnd}}(\lambda)$. We have Definition 10 for capturing the unforgeability of EFDGS.

Definition 10 (Unforgeability). *$\mathcal{EFDGS}$ is unforgeable if, for any (PPT) $\mathcal{A}$,* $\Pr[\mathsf{E}_{\mathcal{A}}^{\mathsf{unf\text{-}wo\text{-}tracesnd}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$ *and* $\Pr[\mathsf{E}_{\mathcal{A}}^{\mathsf{tracesnd}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$ *where* $\mathsf{E}_{\mathcal{A}}^{\mathsf{unf\text{-}wo\text{-}tracesnd}}(\lambda)$ *and* $\mathsf{E}_{\mathcal{A}}^{\mathsf{tracesnd}}(\lambda)$ *are in Figure 4.*

5 Preliminaries for Our Lattice-Based Instantiation

5.1 LLNW Accumulator (Brief)

We briefly recall LLNW accumulator [51,52]. Its detailed description and security are deferred to Appendix A.3. The LLNW accumulator allows to accumulate a sequence of $N = 2^L$ vectors $D = (\mathbf{d}_i)_{i \in [0, N-1]} \in (\{0,1\}^{nk})^N$, for some $n, k, K \in \mathbb{N}$, by employing a hash function $h_{\mathbf{T}} : \{0,1\}^{nk} \times \{0,1\}^{nk} \rightarrow \{0,1\}^{nk}$. Specifically, this accumulator maps $(\mathbf{d}_i)_{i \in [0, N-1]}$ into a single vector $\mathbf{u}_\epsilon \in \{0,1\}^{nk}$.

$\underline{\mathsf{E}^{\mathsf{unf\text{-}wo\text{-}tracesnd}}_{\mathcal{A}}(\lambda)}$
$(\mathsf{pp}, \mathsf{info}^{\mathrm{init}}, \mathsf{sk}_{\mathsf{sa}}, \mathsf{sk}_{\mathsf{ta}}, \mathsf{td}_{\mathsf{ext}}) \leftarrow \mathsf{SimSetup}(1^\lambda)$.
$\mathsf{acc\text{-}sk}_{\mathsf{sa}} := 0, \mathsf{CL} := \emptyset, \mathsf{HL} := \emptyset, \mathsf{PL} := \emptyset, \mathsf{SIGS} := \emptyset$.
$(M, \Sigma) \leftarrow \mathcal{A}^{\mathsf{OSA},\mathsf{OCrptU},\mathsf{OCrptSA},\mathsf{OSign},\mathsf{OTrace}}(\mathsf{pp}, \mathsf{sk}_{\mathsf{ta}}, \mathsf{info}^{\mathrm{init}})$.
Return 0 if $(M, \Sigma) \in \mathsf{SIGS}$ or $\mathsf{Verify}(\mathsf{pp}, M, \Sigma) = 0$.
$\mathsf{EPL} := \left\{ (\widetilde{M}, \mathsf{info}^\tau) \;\middle|\; \begin{array}{l} (\mathsf{info}^\tau, \Gamma, \mathsf{trans}) \in \mathsf{PL} \\ \wedge (\widetilde{M}, \widetilde{\mathsf{aux}}) \leftarrow \mathsf{ExtPreSig}(\mathsf{pp}, \mathsf{td}_{\mathsf{ext}}, \mathsf{info}^\tau, \Gamma, \mathsf{trans}) \end{array} \right\}$.
$(\mathsf{id}', \Xi) \leftarrow \mathsf{Trace}(\mathsf{sk}_{\mathsf{ta}}, M, \Sigma), \quad (\widetilde{\mathsf{id}}, \widetilde{\mathsf{info}}^\tau) \leftarrow \mathsf{ExtSig}(\mathsf{pp}, \mathsf{td}_{\mathsf{ext}}, M, \Sigma)$.
Return 1 if one of the following conditions holds.
1. $\mathsf{Judge}(\mathsf{pp}, M, \Sigma, \mathsf{id}', \Xi) = 1 \wedge \mathsf{id}' \in \mathcal{ID} \wedge \widetilde{\mathsf{id}} \neq \mathsf{id}'$. // extractability
2. $(M, \widetilde{\mathsf{info}}^\tau) \notin \mathsf{EPL} \wedge \mathsf{acc\text{-}sk}_{\mathsf{sa}} = 0$. // system information unforgeability
3. $(\mathsf{IsActive}(\mathsf{pp}, \widetilde{\mathsf{info}}^\tau, \mathsf{id}') = 0 \vee \mathsf{Judge}(\mathsf{pp}, M, \Sigma, \mathsf{id}', \Xi) = 0 \vee \mathsf{id}' = \perp)$
$\wedge \mathsf{id}' \notin \mathsf{CL} \wedge \mathsf{acc\text{-}sk}_{\mathsf{sa}} = 0$. // traceability
4. $\mathsf{IsActive}(\mathsf{pp}, \widetilde{\mathsf{info}}^\tau, \mathsf{id}') = 1 \wedge \mathsf{Judge}(\mathsf{pp}, M, \Sigma, \mathsf{id}', \Xi) = 1 \wedge \mathsf{id}' \in \mathsf{HL}$. // non-frameability
Otherwise, return 0.

$\underline{\mathsf{E}^{\mathsf{tracesnd}}_{\mathcal{A}}(\lambda)}$
$(\mathsf{pp}, \mathsf{info}^{\mathrm{init}}, \mathsf{sk}_{\mathsf{sa}}, \mathsf{sk}_{\mathsf{ta}}, \mathsf{td}_{\mathsf{ext}}) \leftarrow \mathsf{SimSetup}(1^\lambda)$.
$\mathsf{acc\text{-}sk}_{\mathsf{sa}} := 0, \mathsf{CL} := \emptyset, \mathsf{HL} := \emptyset, \mathsf{PL} := \emptyset, \mathsf{SIGS} := \emptyset$.
$(M, \Sigma, \mathsf{id}^{(0)}, \Xi^{(0)}, \mathsf{id}^{(1)}, \Xi^{(1)}) \leftarrow \mathcal{A}^{\mathsf{OSA},\mathsf{OCrptU},\mathsf{OCrptSA},\mathsf{OSign},\mathsf{OTrace}}(\mathsf{pp}, \mathsf{sk}_{\mathsf{ta}}, \mathsf{info}^{\mathrm{init}})$.
If $\mathsf{Verify}(\mathsf{pp}, M, \Sigma) = 0$, return 0.
Return 1 if $\mathsf{id}^{(0)} \neq \mathsf{id}^{(1)} \wedge \mathsf{id}^{(0)} \neq \perp \wedge \mathsf{id}^{(1)} \neq \perp \wedge \mathsf{Judge}(\mathsf{pp}, M, \Sigma, \mathsf{id}^{(b)}, \Xi^{(b)}) = 1 \; \forall b \in \{0,1\}$. Otherwise, return 0.

Fig. 4. Experiments $\mathsf{E}^{\mathsf{unf\text{-}wo\text{-}tracesnd}}_{\mathcal{A}}(\lambda)$ and $\mathsf{E}^{\mathsf{tracesnd}}_{\mathcal{A}}(\lambda)$.

We can verify whether a vector $\mathbf{d} \in \{0,1\}^{nk}$ belongs to D if there exist $w = ((j_1, \ldots, j_L), (\mathbf{w}_1, \ldots, \mathbf{w}_L)) \in \{0,1\}^L \times (\{0,1\}^{nk})^L$ and $\mathbf{v}_0, \ldots, \mathbf{v}_L$ such that $\mathbf{v}_L = \mathbf{d}$, $\mathbf{v}_0 = \mathbf{u}_\epsilon$, and, for all $i \in [0, L-1]$, it holds that $\mathbf{v}_i = h_{\mathbf{T}}(\mathbf{v}_{i+1}, \mathbf{w}_{i+1})$ if $j_{i+1} = 0$, and $\mathbf{v}_i = h_{\mathbf{T}}(\mathbf{w}_{i+1}, \mathbf{v}_{i+1})$, otherwise.

This accumulator also supports an algorithm (say, $\mathsf{A.ModLeaf}$) that can change the accumulated value $\mathbf{u}$ to $\mathbf{u}'$ corresponding to an updated sequence D' obtained by changing $\mathbf{d}_i$ to $\mathbf{d}'_i$ for some $i \in \mathbb{N}$.

5.2 JRS Oblivious Signing Interactive Protocol (Brief)

In this section, we briefly recall the JRS oblivious signing interactive protocol from Jeudy, Roux-Langlois, and Sanders [45]. This protocol is employed to realize the pre-signature issuance mechanism that we introduced previously in Section 3. However, as introduced, the system authority SA generates a pre-signature, in the form of a signature on a committed message (or "blind" signature with a similar sense), for a concatenated message (M, info^τ) where M is only known to the requester while info^τ is public. Therefore, we slightly modify the JRS oblivious signing interactive protocol.

Assume that a user has a private message $\mathbf{m}$ and would like to obtain a signature on a commitment to the message $(\mathbf{m} \| \mathbf{m}')$ where $\mathbf{m}'$ is a public message. The signer in this protocol receives as a commitment $\mathbf{c}$ (see Appendix A.2 for the definition of commitment schemes) to the binary message $\mathbf{m}$ computed as $\mathbf{c} := \mathbf{A} \cdot \mathbf{r}' + \mathbf{D}_1 \cdot \mathbf{m} \mod q$ where $\mathbf{A}$ and $\mathbf{D}_1$ are public matrices while $\mathbf{r}'$ is the randomness protecting the privacy of $\mathbf{m}$.

When receiving $\mathbf{c}$, the signer can transform into a commitment of $(\mathbf{m} \| \mathbf{m}')$ by computing $\mathbf{c} + \mathbf{D}_2 \cdot \mathbf{m}' \mod q = \mathbf{A} \cdot \mathbf{r}' + [\mathbf{D}_1 | \mathbf{D}_2] \cdot (\mathbf{m} \| \mathbf{m}') \mod q$ which is a commitment to $(\mathbf{m} \| \mathbf{m}')$. Here $\mathbf{D}_2$ is another public matrix. Then, the signer creates a signature (say, Γ) on the commitment $\mathbf{c} + \mathbf{D}_2 \cdot \mathbf{m}'$ and sends it back to the user.

The user can eventually transform Γ into a signature authorized by the signer by removing from Γ the randomness $\mathbf{r}'$ that she previously used to create $\mathbf{c}$.

The full description of the protocol is recalled in Appendix A.7.

6 Our Lattice-Based Instantiation of EFDGS (Brief)

We briefly describe our construction, based on [60,61,70,58,71] for managing the system by Setup, Join, and Update, following the syntax specified in Definition 1. Recall that the entities involved are (i) users (U), (ii) system authority SA, and (iii) tracing authority TA.

According to the technical overview in Section 3.2, our lattice-based instantiation needs the following components: (i) LLNW accumulator $\mathcal{ACC}_{\mathsf{llnw}}$ (built upon a hash function $h_{\mathbf{T}}$) [51,52], (ii) JRS oblivious signing protocol $\Pi_{\mathsf{obl\text{-}sign}}$ [45], (iii) Lindner-Peikert IND-CPA-secure PKE $\mathcal{PKE}_{\mathsf{lp}}$ [57] (recalled in Appendix A.5), and (iv) NIZK proof systems for circuit satisfiability over $\mathbb{Z}_q$. Notice that we choose Lindner-Peikert PKE because it can be instantiated with smaller keys, ciphertexts, and encryption randomness without degrading concrete security on the LWE problem. Following Section 3.2, we briefly describe the instantiation below. The full description and analysis of the instantiation are discussed in Appendix B.

The algorithm Setup initializes the empty registry containing N vectors $\mathbf{0}^{nk}$ and accumulate them using $\mathcal{ACC}_{\mathsf{llnw}}$. The accumulated value is the initial system information $\mathsf{info}^{\text{init}}$ corresponding to the initial epoch init. It also runs setup/key generation algorithms $\Pi_{\mathsf{obl}}.\mathsf{Setup}$ and $\mathsf{PKE.Gen}$ of $\Pi_{\mathsf{obl\text{-}sign}}$ and $\mathcal{PKE}_{\mathsf{lp}}$. The secret keys output by $\Pi_{\mathsf{obl\text{-}sign}}$ and $\mathcal{PKE}_{\mathsf{lp}}$ are respectively $\mathsf{sk}_{\mathsf{sa}}$ and $\mathsf{sk}_{\mathsf{ta}}$, while all public keys are included in gpk, hence included in pp.

A user id has a secret key $\mathsf{sk}_{\mathsf{id}}$ and a corresponding public key $\mathsf{pk}_{\mathsf{id}}$. If SA accepts this user to the system, SA employs algorithm $\mathsf{A.ModLeaf}$ (discussed in Section 5.1) to recompute the accumulated value and view it as the next system information $\mathsf{info}^{\text{next}}$ and advance the system to the next epoch.

To revoke users by algorithm Update, SA simply removes the public keys of to-be-revoked users by setting the entries corresponding to them as $\mathbf{0}^{nk}$ in the registry. Then, run $\mathsf{A.ModLeaf}$ to recompute the accumulated value, view it as the next system information $\mathsf{info}^{\text{next}}$, and advance the system to the next epoch.

Regarding algorithm $\mathsf{PreSign}$ for a message $\mathbf{m}$, a user needs to obtain a signature Γ on a commitment to $(\mathbf{m}\|\mathsf{info}^{\text{cur}})$ where $\mathsf{info}^{\text{cur}}$ is the current system information, via interacting with SA to run $\Pi_{\mathsf{obl\text{-}sign}}$. Γ is then considered the pre-signature. Issuing Γ also requires SA to issue a ZK proof for generating Γ to certify that SA actually authorizes $(\mathbf{m}\|\mathsf{info}^{\text{cur}})$. Hence, the algorithm $\mathsf{PreVerify}$ simply verifies Γ output by SA according to the specification of $\Pi_{\mathsf{obl\text{-}sign}}$.

When issuing an EFDGS signature for $\mathbf{m}$ (via algorithm Sign), a signer must show that, via a NIZK proof $\mathsf{NIZK}_{\mathsf{sig}}$, she possesses a valid pre-signature for $\mathbf{m}$, is active at the epoch in which she requested the pre-signature, and correctly encrypts (by TA's public key $\mathsf{pk}_{\mathsf{ta}}$) her identity (namely, $\mathsf{pk}_{\mathsf{id}}$), via $\mathcal{PKE}_{\mathsf{lp}}$, into a ciphertext ct, for tracing purpose. Hence, the output EFDGS signature Σ includes ct and $\mathsf{NIZK}_{\mathsf{sig}}$. Verifying Σ (via Verify) is simply verifying the NIZK proof $\mathsf{NIZK}_{\mathsf{sig}}$.

When tracing an EFDGS signature $\Sigma = (\mathsf{ct}, \mathsf{NIZK}_{\mathsf{sig}})$ via algorithm Trace, TA decrypts ct, by $\mathsf{sk}_{\mathsf{ta}}$, to obtain the signer's public key and additionally produces a NIZK proof $\mathsf{NIZK}_{\mathsf{trace}}$ for the decryption. The algorithm Judge simply verifies $\mathsf{NIZK}_{\mathsf{trace}}$.

Regarding the NIZK proofs, we can employ any post-quantum NIZK proof systems for circuits, e.g., [40,85,63,10].

Acknowledgments

The research was supported by Singapore Ministry of Education Academic Research Fund Tier 2 Grant MOE-000623-00. We thank Hien Chu, Chan Nam Ngo, Minh Pham, Yanhong Xu, Yingfei Yan, and Yaxi Yang (lexicographically ordered by surnames) for their helpful comments and suggestions.

References

1. Ajtai, M.: Generating hard instances of the short basis problem. In: Wiedermann, J., van Emde Boas, P., Nielsen, M. (eds.) Automata, Languages and Programming – ICAL 1999. Lecture Notes in Computer Science, vol. 1644, pp. 1–9. Springer Berlin Heidelberg (1999). https://doi.org/doi.org/10.1007/3-540-48523-6_1
2. An, Z., Pan, J., Wen, Y., Zhang, F.: Secret handshakes: Full dynamicity, deniability and lattice-based design. Theor. Comput. Sci. **940**(Part), 14–35 (2023). https://doi.org/10.1016/J.TCS.2022.10.035
3. Ateniese, G., Camenisch, J., Joye, M., Tsudik, G.: A Practical and Provably Secure Coalition-Resistant Group Signature Scheme. In: Bellare, M. (ed.) Advances in Cryptology — CRYPTO 2000. Lecture Notes in Computer Science, vol. 1880, pp. 255–270. Springer Berlin Heidelberg (2000). https://doi.org/10.1007/3-540-44598-6_16

4. Ateniese, G., Tsudik, G.: Group signatures á la carte. In: Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms – SODA 1999. p. 848–849. Society for Industrial and Applied Mathematics (1999)
5. Ateniese, G., Tsudik, G.: Some Open Issues and New Directions in Group Signatures. In: Franklin, M. (ed.) Financial Cryptography – FC 1999. Lecture Notes in Computer Science, vol. 1648, pp. 196–211. Springer Berlin Heidelberg (1999). https://doi.org/10.1007/3-540-48390-X_15
6. Attema, T., Cramer, R.: Compressed Σ-Protocol Theory and Practical Application to Plug & Play Secure Algorithmics. In: Micciancio, D., Ristenpart, T. (eds.) Advances in Cryptology – CRYPTO 2020. Lecture Notes in Computer Science, vol. 12172, pp. 513–543. Springer International Publishing (2020). https://doi.org/10.1007/978-3-030-56877-1_18
7. Attema, T., Fehr, S., Klooß, M.: Fiat-Shamir Transformation of Multi-Round Interactive Proofs (Extended Version). J. Cryptol. **36**(4), 36 (2023). https://doi.org/10.1007/S00145-023-09478-Y
8. Attema, T., Fehr, S., Klooß, M., Resch, N.: The fiat–shamir transformation of $(\gamma_1, \ldots, \gamma_\mu)$-special-sound interactive proofs. Cryptology ePrint Archive, Paper 2023/1945 (2023), https://eprint.iacr.org/2023/1945
9. Ayebie, E.B., Souidi, E.M.: New code-based cryptographic accumulator and fully dynamic group signature. Des. Codes Cryptogr. **90**(12), 2861–2891 (2022). https://doi.org/10.1007/S10623-022-01007-5
10. Baum, C., Braun, L., de Saint Guilhem, C.D., Klooß, M., Orsini, E., Roy, L., Scholl, P.: Publicly Verifiable Zero-Knowledge and Post-Quantum Signatures from VOLE-in-the-Head. In: Handschuh, H., Lysyanskaya, A. (eds.) Advances in Cryptology – CRYPTO 2023. Lecture Notes in Computer Science, vol. 14085, pp. 581–615. Springer Nature Switzerland (2023). https://doi.org/10.1007/978-3-031-38554-4_19
11. Bellare, M., Fuchsbauer, G.: Policy-Based Signatures. In: Krawczyk, H. (ed.) Public-Key Cryptography – PKC 2014. Lecture Notes in Computer Science, vol. 8383, pp. 520–537. Springer Berlin Heidelberg (2014). https://doi.org/10.1007/978-3-642-54631-0_30
12. Bellare, M., Micciancio, D., Warinschi, B.: Foundations of Group Signatures: Formal Definitions, Simplified Requirements, and a Construction Based on General Assumptions. In: Biham, E. (ed.) Advances in Cryptology — EUROCRYPT 2003. Lecture Notes in Computer Science, vol. 2656, pp. 614–629. Springer Berlin Heidelberg (2003). https://doi.org/10.1007/3-540-39200-9_38
13. Bellare, M., Shi, H., Zhang, C.: Foundations of Group Signatures: The Case of Dynamic Groups. In: Menezes, A. (ed.) Topics in Cryptology – CT-RSA 2005. Lecture Notes in Computer Science, vol. 3376, pp. 136–153. Springer Berlin Heidelberg (2005). https://doi.org/10.1007/978-3-540-30574-3_11
14. Beullens, W., Lyubashevsky, V., Nguyen, N.K., Seiler, G.: Lattice-Based Blind Signatures: Short, Efficient, and Round-Optimal. In: Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security – CCS 2023. p. 16–29. Association for Computing Machinery (2023). https://doi.org/10.1145/3576915.3616613
15. Blum, M., Feldman, P., Micali, S.: Non-interactive zero-knowledge and its applications. In: Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing – STOC 1988. p. 103–112. Association for Computing Machinery (1988). https://doi.org/10.1145/62212.62222
16. Boneh, D., Boyen, X., Shacham, H.: Short Group Signatures. In: Franklin, M. (ed.) Advances in Cryptology – CRYPTO 2004. Lecture Notes in Computer Science, vol. 3152, pp. 41–55. Springer Berlin Heidelberg (2004). https://doi.org/10.1007/978-3-540-28628-8_3
17. Boneh, D., Shacham, H.: Group signatures with verifier-local revocation. In: Proceedings of the 11th ACM Conference on Computer and Communications Security – CCS 2004. p. 168–177. Association for Computing Machinery (2004). https://doi.org/10.1145/1030083.1030106
18. Bootle, J., Cerulli, A., Chaidos, P., Ghadafi, E., Groth, J.: Foundations of Fully Dynamic Group Signatures. In: Manulis, M., Sadeghi, A.R., Schneider, S. (eds.) Applied Cryptography and Network Security – ACNS 2016. Lecture Notes in Computer Science, vol. 9696, pp. 117–136. Springer International Publishing (2016). https://doi.org/10.1007/978-3-319-39555-5_7
19. Bootle, J., Cerulli, A., Chaidos, P., Ghadafi, E., Groth, J.: Foundations of Fully Dynamic Group Signatures. J. Cryptol. **33**(4), 1822–1870 (2020). https://doi.org/10.1007/S00145-020-09357-W
20. Bresson, E., Stern, J.: Efficient Revocation in Group Signatures. In: Kim, K. (ed.) Public Key Cryptography – PKC 2001. vol. 1992, pp. 190–206. Springer Berlin Heidelberg (2001). https://doi.org/10.1007/3-540-44586-2_15
21. Brickell, E.: An efficient protocol for anonymously providing assurance of the container of a private key. Submission to the Trusted Computing Group (2004)
22. Brickell, E., Camenisch, J., Chen, L.: Direct anonymous attestation. In: Proceedings of the 11th ACM Conference on Computer and Communications Security – CCS 2004. p. 132–145. Association for Computing Machinery (2004). https://doi.org/10.1145/1030083.1030103
23. Camenisch, J.: Efficient and Generalized Group Signatures. In: Fumy, W. (ed.) Advances in Cryptology — EUROCRYPT 1997. Lecture Notes in Computer Science, vol. 1233, pp. 465–479. Springer Berlin Heidelberg (1997). https://doi.org/10.1007/3-540-69053-0_32
24. Camenisch, J., Groth, J.: Group Signatures: Better Efficiency and New Theoretical Aspects. In: Blundo, C., Cimato, S. (eds.) Security in Communication Networks – SCN 2004. Lecture Notes in Computer Science, vol. 3352, pp. 120–133. Springer Berlin Heidelberg (2005). https://doi.org/10.1007/978-3-540-30598-9_9

25. Camenisch, J., Lysyanskaya, A.: Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials. In: Yung, M. (ed.) Advances in Cryptology — CRYPTO 2002. Lecture Notes in Computer Science, vol. 2442, pp. 61–76. Springer Berlin Heidelberg (2002). https://doi.org/10.1007/3-540-45708-9_5
26. Camenisch, J., Michels, M.: A Group Signature Scheme with Improved Efficiency (Extended Abstract). In: Ohta, K., Pei, D. (eds.) Advances in Cryptology — ASIACRYPT 1998. Lecture Notes in Computer Science, vol. 1514, pp. 160–174. Springer Berlin Heidelberg (1998). https://doi.org/10.1007/3-540-49649-1_14
27. Camenisch, J., Stadler, M.: Efficient group signature schemes for large groups. In: Kaliski, B.S. (ed.) Advances in Cryptology — CRYPTO 1997. Lecture Notes in Computer Science, vol. 1294, pp. 410–424. Springer Berlin Heidelberg (1997). https://doi.org/10.1007/BFb0052252
28. Chase, M., Lysyanskaya, A.: On Signatures of Knowledge. In: Dwork, C. (ed.) Advances in Cryptology - CRYPTO 2006. Lecture Notes in Computer Science, vol. 4117, pp. 78–96. Springer Berlin Heidelberg (2006). https://doi.org/10.1007/11818175_5
29. Chaum, D.: Blind Signatures for Untraceable Payments. In: Chaum, D., Rivest, R.L., Sherman, A.T. (eds.) Advances in Cryptology – CRYPTO 1982. pp. 199–203. Springer US, Boston, MA (1983). https://doi.org/10.1007/978-1-4757-0602-4_18
30. Chaum, D., van Heyst, E.: Group Signatures. In: Davies, D.W. (ed.) Advances in Cryptology — EUROCRYPT 1991. Lecture Notes in Computer Science, vol. 547, pp. 257–265. Springer Berlin Heidelberg (1991). https://doi.org/10.1007/3-540-46416-6_22
31. Chen, L., Pedersen, T.P.: New group signature schemes. In: De Santis, A. (ed.) Advances in Cryptology — EUROCRYPT 1994. Lecture Notes in Computer Science, vol. 950, pp. 171–181. Springer Berlin Heidelberg (1995). https://doi.org/10.1007/BFb0053433
32. Dodis, Y., Kiayias, A., Nicolosi, A., Shoup, V.: Anonymous Identification in Ad Hoc Groups. In: Cachin, C., Camenisch, J.L. (eds.) Advances in Cryptology - EUROCRYPT 2004. Lecture Notes in Computer Science, vol. 3027, pp. 609–626. Springer Berlin Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_36
33. El Kaafarani, A., Katsumata, S., Solomon, R.: Anonymous Reputation Systems Achieving Full Dynamicity from Lattices. In: Meiklejohn, S., Sako, K. (eds.) Financial Cryptography and Data Security – FC 2018. Lecture Notes in Computer Science, vol. 10957, pp. 388–406. Springer Berlin Heidelberg (2018). https://doi.org/10.1007/978-3-662-58387-6_21
34. Esgin, M.F., Steinfeld, R., Sakzad, A., Liu, J.K., Liu, D.: Short Lattice-Based One-out-of-Many Proofs and Applications to Ring Signatures. In: Deng, R.H., Gauthier-Umaña, V., Ochoa, M., Yung, M. (eds.) Applied Cryptography and Network Security. Lecture Notes in Computer Science, vol. 11464, pp. 67–88. Springer International Publishing (2019). https://doi.org/10.1007/978-3-030-21568-2_4
35. Feige, U., Lapidot, D., Shamir, A.: Multiple non-interactive zero knowledge proofs based on a single random string. In: Proceedings [1990] 31st Annual Symposium on Foundations of Computer Science – FOCS 1990. pp. 308–317 vol.1. IEEE (1990). https://doi.org/10.1109/FSCS.1990.89549
36. Feige, U., Shamir, A.: Witness indistinguishable and witness hiding protocols. In: Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing – STOC 1990. p. 416–426. Association for Computing Machinery (1990). https://doi.org/10.1145/100216.100272
37. Fiat, A., Shamir, A.: How To Prove Yourself: Practical Solutions to Identification and Signature Problems. In: Odlyzko, A.M. (ed.) Advances in Cryptology — CRYPTO 1986. Lecture Notes in Computer Science, vol. 263, pp. 186–194. Springer Berlin Heidelberg (1987). https://doi.org/10.1007/3-540-47721-7_12
38. Fujisaki, E., Okamoto, T.: Secure Integration of Asymmetric and Symmetric Encryption Schemes. In: Wiener, M. (ed.) Advances in Cryptology — CRYPTO 1999. Lecture Notes in Computer Science, vol. 1666, pp. 537–554. Springer Berlin Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_34
39. Fujisaki, E., Okamoto, T.: Secure Integration of Asymmetric and Symmetric Encryption Schemes. J. Cryptol. **26**(1), 80–101 (2013). https://doi.org/10.1007/S00145-011-9114-1
40. Giacomelli, I., Madsen, J., Orlandi, C.: ZKBoo: Faster Zero-Knowledge for Boolean Circuits. In: 25th USENIX Security Symposium – USENIX Security 2016. pp. 1069–1083. USENIX Association (2016), https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/giacomelli
41. Groth, J., Kohlweiss, M.: One-Out-of-Many Proofs: Or How to Leak a Secret and Spend a Coin. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology - EUROCRYPT 2015. Lecture Notes in Computer Science, vol. 9057, pp. 253–280. Springer Berlin Heidelberg (2015). https://doi.org/10.1007/978-3-662-46803-6_9
42. Groth, J., Sahai, A.: Efficient Non-interactive Proof Systems for Bilinear Groups. In: Smart, N. (ed.) Advances in Cryptology – EUROCRYPT 2008. Lecture Notes in Computer Science, vol. 4965, pp. 415–432. Springer Berlin Heidelberg (2008). https://doi.org/10.1007/978-3-540-78967-3_24
43. Håstad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A pseudorandom generator from any one-way function. SIAM J. Comput. **28**(4), 1364–1396 (1999). https://doi.org/10.1137/S0097539793244708
44. Jain, A., Krenn, S., Pietrzak, K., Tentes, A.: Commitments and Efficient Zero-Knowledge Proofs from Learning Parity with Noise. In: Wang, X., Sako, K. (eds.) Advances in Cryptology – ASIACRYPT 2012. Lecture Notes in Computer Science, vol. 7658, pp. 663–680. Springer Berlin Heidelberg (2012). https://doi.org/10.1007/978-3-642-34961-4_40

45. Jeudy, C., Roux-Langlois, A., Sanders, O.: Lattice signature with efficient protocols, application to anonymous credentials. In: Handschuh, H., Lysyanskaya, A. (eds.) Advances in Cryptology – CRYPTO 2023. Lecture Notes in Computer Science, vol. 14082, pp. 351–383. Springer Nature Switzerland (2023). https://doi.org/10.1007/978-3-031-38545-2_12, full version: https://ia.cr/2022/509
46. Katz, J., Lindell, Y.: Introduction to modern cryptography: principles and protocols. Chapman and hall/CRC (2007)
47. Kiayias, A., Tsiounis, Y., Yung, M.: Traceable Signatures. In: Cachin, C., Camenisch, J.L. (eds.) Advances in Cryptology - EUROCRYPT 2004. Lecture Notes in Computer Science, vol. 3027, pp. 571–589. Springer Berlin Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_34
48. Kiayias, A., Yung, M.: Secure scalable group signature with dynamic joins and separable authorities. Int. J. Secur. Networks **1**(1/2), 24–45 (2006). https://doi.org/10.1504/IJSN.2006.010821
49. Langlois, A., Ling, S., Nguyen, K., Wang, H.: Lattice-Based Group Signature Scheme with Verifier-Local Revocation. In: Krawczyk, H. (ed.) Public-Key Cryptography – PKC 2014. Lecture Notes in Computer Science, vol. 8383, pp. 345–361. Springer Berlin Heidelberg (2014). https://doi.org/10.1007/978-3-642-54631-0_20
50. Libert, B., Ling, S., Mouhartem, F., Nguyen, K., Wang, H.: Signature Schemes with Efficient Protocols and Dynamic Group Signatures from Lattice Assumptions. In: Cheon, J.H., Takagi, T. (eds.) Advances in Cryptology – ASIACRYPT 2016. Lecture Notes in Computer Science, vol. 10032, pp. 373–403. Springer Berlin Heidelberg (2016). https://doi.org/10.1007/978-3-662-53890-6_13
51. Libert, B., Ling, S., Nguyen, K., Wang, H.: Zero-Knowledge Arguments for Lattice-Based Accumulators: Logarithmic-Size Ring Signatures and Group Signatures Without Trapdoors. In: Fischlin, M., Coron, J.S. (eds.) Advances in Cryptology – EUROCRYPT 2016. Lecture Notes in Computer Science, vol. 9666, pp. 1–31. Springer Berlin Heidelberg (2016). https://doi.org/10.1007/978-3-662-49896-5_1
52. Libert, B., Ling, S., Nguyen, K., Wang, H.: Zero-knowledge arguments for lattice-based accumulators: Logarithmic-size ring signatures and group signatures without trapdoors. J. Cryptol. **36**(3), 23 (2023). https://doi.org/10.1007/S00145-023-09470-6
53. Libert, B., Nguyen, K., Peters, T., Yung, M.: Bifurcated Signatures: Folding the Accountability vs. Anonymity Dilemma into a Single Private Signing Scheme. In: Canteaut, A., Standaert, F.X. (eds.) Advances in Cryptology – EUROCRYPT 2021. Lecture Notes in Computer Science, vol. 12698, pp. 521–552. Springer International Publishing (2021). https://doi.org/10.1007/978-3-030-77883-5_18
54. Libert, B., Peters, T., Yung, M.: Group Signatures with Almost-for-Free Revocation. In: Safavi-Naini, R., Canetti, R. (eds.) Advances in Cryptology – CRYPTO 2012. Lecture Notes in Computer Science, vol. 7417, pp. 571–589. Springer Berlin Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_34
55. Libert, B., Peters, T., Yung, M.: Scalable Group Signatures with Revocation. In: Pointcheval, D., Johansson, T. (eds.) Advances in Cryptology – EUROCRYPT 2012. Lecture Notes in Computer Science, vol. 7237, pp. 609–627. Springer Berlin Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_36
56. Libert, B., Vergnaud, D.: Group Signatures with Verifier-Local Revocation and Backward Unlinkability in the Standard Model. In: Garay, J.A., Miyaji, A., Otsuka, A. (eds.) Cryptology and Network Security – CANS 2009. Lecture Notes in Computer Science, vol. 5888, pp. 498–517. Springer Berlin Heidelberg (2009). https://doi.org/10.1007/978-3-642-10433-6_34
57. Lindner, R., Peikert, C.: Better Key Sizes (and Attacks) for LWE-Based Encryption. In: Kiayias, A. (ed.) Topics in Cryptology – CT-RSA 2011. Lecture Notes in Computer Science, vol. 6558, pp. 319–339. Springer Berlin Heidelberg (2011). https://doi.org/10.1007/978-3-642-19074-2_21
58. Ling, S., Nguyen, K., Phan, D.H., Tang, K.H., Wang, H., Xu, Y.: Fully Dynamic Attribute-Based Signatures for Circuits from Codes. In: Tang, Q., Teague, V. (eds.) Public-Key Cryptography – PKC 2024. Lecture Notes in Computer Science, vol. 14601, pp. 37–73. Springer Nature Switzerland (2024). https://doi.org/10.1007/978-3-031-57718-5_2
59. Ling, S., Nguyen, K., Stehlé, D., Wang, H.: Improved Zero-Knowledge Proofs of Knowledge for the ISIS Problem, and Applications. In: Kurosawa, K., Hanaoka, G. (eds.) Public-Key Cryptography – PKC 2013. Lecture Notes in Computer Science, vol. 7778, pp. 107–124. Springer Berlin Heidelberg (2013). https://doi.org/10.1007/978-3-642-36362-7_8
60. Ling, S., Nguyen, K., Wang, H., Xu, Y.: Lattice-Based Group Signatures: Achieving Full Dynamicity with Ease. In: Gollmann, D., Miyaji, A., Kikuchi, H. (eds.) Applied Cryptography and Network Security – ACNS 2017. Lecture Notes in Computer Science, vol. 10355, pp. 293–312. Springer International Publishing (2017). https://doi.org/10.1007/978-3-319-61204-1_15
61. Ling, S., Nguyen, K., Wang, H., Xu, Y.: Lattice-based group signatures: Achieving full dynamicity (and deniability) with ease. Theor. Comput. Sci. **783**, 71–94 (2019). https://doi.org/10.1016/J.TCS.2019.03.023
62. Lyubashevsky, V.: Lattice-Based Identification Schemes Secure Under Active Attacks. In: Cramer, R. (ed.) Public Key Cryptography – PKC 2008. Lecture Notes in Computer Science, vol. 4939, pp. 162–179. Springer Berlin Heidelberg, Berlin, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78440-1_10
63. Lyubashevsky, V., Nguyen, N.K., Plançon, M.: Lattice-Based Zero-Knowledge Proofs and Applications: Shorter, Simpler, and More General. In: Dodis, Y., Shrimpton, T. (eds.) Advances in Cryptology – CRYPTO 2022. Lecture Notes in Computer Science, vol. 13508, pp. 71–101. Springer Nature Switzerland (2022). https://doi.org/10.1007/978-3-031-15979-4_3

64. Micciancio, D., Peikert, C.: Trapdoors for Lattices: Simpler, Tighter, Faster, Smaller. In: Pointcheval, D., Johansson, T. (eds.) Advances in Cryptology – EUROCRYPT 2012. Lecture Notes in Computer Science, vol. 7237, pp. 700–718. Springer Berlin Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_41
65. Nakanishi, T., Fujii, H., Hira, Y., Funabiki, N.: Revocable Group Signature Schemes with Constant Costs for Signing and Verifying. In: Jarecki, S., Tsudik, G. (eds.) Public Key Cryptography – PKC 2009. Lecture Notes in Computer Science, vol. 5443, pp. 463–480. Springer Berlin Heidelberg (2009). https://doi.org/10.1007/978-3-642-00468-1_26
66. Nakanishi, T., Funabiki, N.: Verifier-Local Revocation Group Signature Schemes with Backward Unlinkability from Bilinear Maps. In: Roy, B. (ed.) Advances in Cryptology - ASIACRYPT 2005. Lecture Notes in Computer Science, vol. 3788, pp. 533–548. Springer Berlin Heidelberg (2005). https://doi.org/10.1007/11593447_29
67. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing - STOC 1990. p. 427–437. Association for Computing Machinery (1990). https://doi.org/10.1145/100216.100273
68. Nguyen, K., Guo, F., Susilo, W., Yang, G.: Multimodal private signatures. In: Dodis, Y., Shrimpton, T. (eds.) Advances in Cryptology – CRYPTO 2022. Lecture Notes in Computer Science, vol. 13508, pp. 792–822. Springer Nature Switzerland (2022). https://doi.org/10.1007/978-3-031-15979-4_27
69. Nguyen, K., Roy, P.S., Susilo, W., Xu, Y.: Bicameral and Auditably Private Signatures. In: Guo, J., Steinfeld, R. (eds.) Advances in Cryptology – ASIACRYPT 2023. Lecture Notes in Computer Science, vol. 14439, pp. 313–347. Springer Nature Singapore (2023). https://doi.org/10.1007/978-981-99-8724-5_10
70. Nguyen, K., Safavi-Naini, R., Susilo, W., Wang, H., Xu, Y., Zeng, N.: Group Encryption: Full Dynamicity, Message Filtering and Code-Based Instantiation. In: Garay, J.A. (ed.) Public-Key Cryptography – PKC 2021. Lecture Notes in Computer Science, vol. 12711, pp. 678–708. Springer International Publishing (2021). https://doi.org/10.1007/978-3-030-75248-4_24
71. Nguyen, K., Safavi-Naini, R., Susilo, W., Wang, H., Xu, Y., Zeng, N.: Group encryption: Full dynamicity, message filtering and code-based instantiation. Theor. Comput. Sci. **1007**, 114678 (2024). https://doi.org/10.1016/J.TCS.2024.114678
72. Nguyen, K., Tang, H., Wang, H., Zeng, N.: New Code-Based Privacy-Preserving Cryptographic Constructions. In: Galbraith, S.D., Moriai, S. (eds.) Advances in Cryptology – ASIACRYPT 2019. Lecture Notes in Computer Science, vol. 11922, pp. 25–55. Springer International Publishing (2019). https://doi.org/10.1007/978-3-030-34621-8_2
73. Nguyen, L.: Accumulators from Bilinear Pairings and Applications. In: Menezes, A. (ed.) Topics in Cryptology – CT-RSA 2005. Lecture Notes in Computer Science, vol. 3376, pp. 275–292. Springer Berlin Heidelberg (2005). https://doi.org/10.1007/978-3-540-30574-3_19
74. Ouyang, Y., Tang, D., Xu, Y.: Code-Based Zero-Knowledge from VOLE-in-the-Head and Their Applications: Simpler, Faster, and Smaller. ASIACRYPT 2024, to appear (2024), https://eprint.iacr.org/2024/1414
75. Pan, J., Chen, X., Zhang, F., Susilo, W.: Lattice-Based Group Encryption with Full Dynamicity and Message Filtering Policy. In: Tibouchi, M., Wang, H. (eds.) Advances in Cryptology – ASIACRYPT 2021. Lecture Notes in Computer Science, vol. 13093, pp. 156–186. Springer International Publishing (2021). https://doi.org/10.1007/978-3-030-92068-5_6
76. Petersen, H.: How to convert any digital signature scheme into a group signature scheme. In: Christianson, B., Crispo, B., Lomas, M., Roe, M. (eds.) Security Protocols. Lecture Notes in Computer Science, vol. 1361, pp. 177–190. Springer Berlin Heidelberg (1998). https://doi.org/10.1007/BFb0028168
77. Pointcheval, D., Vaudenay, S.: On provable security for digital signature algorithms. Technical Report LIENS-96-17, Laboratoire d'Informatique de l'Ecole Normale Supérieure (1996)
78. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing – STOC 2005. p. 84–93. Association for Computing Machinery (2005). https://doi.org/10.1145/1060590.1060603
79. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. J. ACM **56**(6) (Sep 2009). https://doi.org/10.1145/1568318.1568324
80. Rivest, R.L., Shamir, A., Tauman, Y.: How to Leak a Secret. In: Boyd, C. (ed.) Advances in Cryptology — ASIACRYPT 2001. Lecture Notes in Computer Science, vol. 2248, pp. 552–565. Springer Berlin Heidelberg (2001). https://doi.org/10.1007/3-540-45682-1_32
81. Song, D.X.: Practical forward secure group signature schemes. In: Proceedings of the 8th ACM Conference on Computer and Communications Security – CCS 2001. p. 225–234. Association for Computing Machinery (2001). https://doi.org/10.1145/501983.502015
82. Sun, Y., Liu, Y.: An efficient fully dynamic group signature with message dependent opening from lattice. Cybersecur. **4**(1) (2021). https://doi.org/10.1186/S42400-021-00076-8
83. Sun, Y., Liu, Y., Wu, B.: An efficient full dynamic group signature scheme over ring. Cybersecur. **2**(1), 21 (2019). https://doi.org/10.1186/S42400-019-0037-8
84. Tsudik, G., Xu, S.: Accumulating Composites and Improved Group Signing. In: Laih, C.S. (ed.) Advances in Cryptology - ASIACRYPT 2003. Lecture Notes in Computer Science, vol. 2894, pp. 269–286. Springer Berlin Heidelberg (2003). https://doi.org/10.1007/978-3-540-40061-5_16

85. Yang, R., Au, M.H., Zhang, Z., Xu, Q., Yu, Z., Whyte, W.: Efficient Lattice-Based Zero-Knowledge Arguments with Standard Soundness: Construction and Applications. In: Boldyreva, A., Micciancio, D. (eds.) Advances in Cryptology – CRYPTO 2019. Lecture Notes in Computer Science, vol. 11692, pp. 147–175. Springer International Publishing (2019). https://doi.org/10.1007/978-3-030-26948-7_6
86. Zhao, Y., Yang, S., Huang, X.: Lattice-based dynamic universal accumulator: Design and application. Comput. Stand. Interfaces **89**, 103807 (2024). https://doi.org/10.1016/J.CSI.2023.103807

A Preliminaries (Extended)

Notations and Conventions. In complement to notations and conventions in Section 2.1, we additionally have those for lattices. Let $\rho_{s,\mathbf{c}}(\mathbf{x}) = \exp\left(-\pi \frac{\|\mathbf{x}-\mathbf{c}\|^2}{s^2}\right)$ be the standard Gaussian function of width s shifted by $\mathbf{c}$. For any full rank n-dimensional lattice $\Lambda \subseteq \mathbb{R}^{n\times n}$ and any $\mathbf{x} \in \Lambda$, we define the discrete Gaussian distribution to be $\mathcal{D}_{\Lambda,s,\mathbf{c}}(\mathbf{x}) = \frac{\rho_{s,\mathbf{c}}(\mathbf{x})}{\rho_{s,\mathbf{c}}(\Lambda)}$. When $\mathbf{c} = \mathbf{0}$, we simply write $\mathcal{D}_{\Lambda,s}$. For a relation of the form $\mathcal{R} = \left\{ (s; w) \,\middle|\, \text{conditions capturing } (s, w) \right\}$, we denote by $\mathcal{L}(\mathcal{R})$ the language capturing $\mathcal{R}$ such that, when writing $s \in \mathcal{L}(\mathcal{R})$, we understand that there exists w satisfying $(s; w) \in \mathcal{R}$.

A.1 A Lemma for Parsing into Binary

Let $\mathbb{Z}_q = \{0, \ldots, q-1\}$ for some $q \in \mathbb{N}$. We recall a method, from [59], for parsing numbers in the range $[0, v) = [0, v-1]$, for some $v \in \mathbb{Z}_q$, into binary. This method is used for constructing zero-knowledge proofs that x belongs to $[0, v)$ for $x \in \mathbb{Z}_q$. This method is summarized in Lemma 11.

Lemma 11. *Let $q \in \mathbb{N}$ and $\mathbb{Z}_q = \{0, \ldots, q-1\}$. Let $v \in \mathbb{Z}_q$, $k_v = \lfloor \log_2(v-1) \rfloor + 1$, $B_i = 2^{i-1}$ for $i \in [k_v - 1]$, and $B_{k_v} = q - (2^{k_v-1} - 1)$. For any $x \in \mathbb{Z}_q$, $x \in [0, v)$ if and only if there exist $b_1, \ldots, b_{k_v} \in \{0, 1\}$ such that $x = \sum_{i\in[k_v]} b_i \cdot B_i$.*

According to Lemma 11, to show that a private number $x \in [0, v)$ for some public $v \in \mathbb{Z}_q$, we can find a sequence $b_1, \ldots, b_{k_v}$ such that we can prove $x \in [0, v)$ by equivalently proving that $b_1, \ldots, b_{k_v} \in \{0, 1\}$ and $x = \sum_{i\in[k_v]} b_i \cdot B_i$.

A.2 Commitment Schemes

We recall the definition of commitment schemes (e.g., see [44]) in the following Definition 12.

Definition 12. *A commitment scheme is a tuple of algorithms*

$$\mathcal{COM} = (\mathsf{C.Gen}, \mathsf{C.Commit}, \mathsf{C.Open})$$

satisfying

$\mathsf{C.Gen}(1^\lambda) \to \mathsf{ck}$**:** *On input the security parameter 1^λ, this randomized algorithm returns a commitment key* ck.

$\mathsf{C.Commit}(\mathsf{ck}, \mathbf{m}; \mathbf{r}) \to \mathsf{cmt}$**:** *On inputs the commitment key* ck *and a message $\mathbf{m}$, this randomized algorithm, using randomness $\mathbf{r}$ (sampled secretly by the committer), produces a commitment* cmt.

$\mathsf{C.Open}(\mathsf{ck}, \mathbf{m}, \mathbf{r}, \mathsf{cmt}) \to \{0, 1\}$**:** *On inputs the commitment key* ck*, a message $\mathbf{m}$, a commitment randomness $\mathbf{r}$, and a commitment* cmt*, this deterministic algorithm outputs a bit $b = \{0, 1\}$ indicating rejection or acceptance.*

Definition 13. *We say the commitment scheme is (perfectly) correct if for all $\mathbf{m}$ in the appropriate message space, all $\mathbf{r}$ random coins used in* $\mathsf{C.Commit}$*:*

$$\Pr\left[\mathsf{C.Open}(\mathsf{ck}, \mathbf{m}, \mathbf{r}, \mathsf{cmt}) = 1 | \mathsf{cmt} \leftarrow \mathsf{C.Commit}(\mathsf{ck}, \mathbf{m}; \mathbf{r})\right] = 1$$

Definition 14. *We say that the commitment scheme is statistically hiding if, for all $\mathbf{m}_0, \mathbf{m}_1$, $\mathsf{cmt}_0 \leftarrow \mathsf{C.Commit}(\mathsf{ck}, \mathbf{m}_0)$ and $\mathsf{cmt}_1 \leftarrow \mathsf{C.Commit}(\mathsf{ck}, \mathbf{m}_1)$, generated with i.i.d. randomness, are statistically indistinguishable.*

The commitment scheme is computationally hiding if no PPT adversary $\mathcal{A}$, except with negligible probability, can distinguish Experiment 0 from Experiment 1 where, for $b \in \{0, 1\}$, Experiment b is defined as follows.

1. *Wait for the adversary to produce* 2 *equal length messages* $\mathbf{m}_0$ *and* $\mathbf{m}_1$. *Return* $\mathsf{cmt}_b \leftarrow \mathsf{C.Commit}(\mathsf{ck}, \mathbf{m}_b; \mathbf{r})$ *to the adversary.*
2. *The adversary then outputs a bit.*

Definition 15. *The commitment scheme is computationally binding if no PPT adversary* $\mathcal{A}$*, except with negligible probability, can produce a tuple* $(\mathsf{cmt}, (\mathbf{m}_0, \mathbf{r}_0), (\mathbf{m}_1, \mathbf{r}_1))$ *satisfying*

$$\mathbf{m}_0 \neq \mathbf{m}_1 \wedge \mathsf{C.Open}(\mathsf{ck}, \mathbf{m}_0, \mathbf{r}_0, \mathsf{cmt}) = 1 \wedge \mathsf{C.Open}(\mathsf{ck}, \mathbf{m}_1, \mathbf{r}_1, \mathsf{cmt}) = 1$$

A.3 LLNW Accumulator (Detailed)

We recall LLNW Merkle-tree-based accumulator [51,52], dubbed LLNW accumulator and denoted by $\mathcal{ACC}_{\mathsf{llnw}}$. This is presented in detail with respect to Section 5.1.

Following the strategy from [60,61,58], we will use this accumulator to accumulate public keys of users into a single accumulated value representing the public system information of the system in Section 6 whose detailed description is in Appendix B. This scheme works as follows. For a security parameter λ, let $n = \mathcal{O}(\lambda)$, $q = \tilde{\mathcal{O}}(n^{1.5})$, $k = \lfloor \log_2 q \rfloor + 1$, and $m = 2nk$. Define $\mathbf{G}_{\mathsf{acc}} = \mathbf{I}_n \otimes (1, 2, \ldots, 2^{k-1})^\top$. According to Appendix A.1, for every $\mathbf{v} \in \mathbb{Z}_q^n$, we can always find its binary representation $\mathsf{bin}_k(\mathbf{v}) \in \{0,1\}^{nk}$ satisfying $\mathbf{v} = \mathbf{G}_{\mathsf{acc}} \cdot \mathsf{bin}_k(\mathbf{v})$. The accumulator employs the following SIS-based collision-resistant hash function. For a matrix $\mathbf{T} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$, the hash function $h_{\mathbf{T}} : \{0,1\}^{nk} \times \{0,1\}^{nk} \to \{0,1\}^{nk}$ maps $(\mathbf{x}_0, \mathbf{x}_1)$ to

$$h_{\mathbf{T}}(\mathbf{x}_0, \mathbf{x}_1) = \mathsf{bin}_k(\mathbf{T} \cdot (\mathbf{x}_0 \| \mathbf{x}_1)) \in \{0,1\}^{nk}. \tag{1}$$

This hash function satisfies collision resistance.

The Merkle tree underlying LLNW accumulator $\mathcal{ACC}_{\mathsf{llnw}}$ has exactly $N = 2^L$ leaves for $L \in \mathbb{N}$. $\mathcal{ACC}_{\mathsf{llnw}}$ is a tuple

$$\mathcal{ACC}_{\mathsf{llnw}} = (\mathsf{A.Setup}, \mathsf{A.Acc}, \mathsf{A.GetRoot}, \mathsf{A.ModLeaf}, \mathsf{A.Witness}, \mathsf{A.Verify})$$

described as follows.

$\mathsf{A.Setup}(1^\lambda) \to \mathbf{T}$**:** This algorithm works as follows.
1. Determine n, q, k, and m as specified above.
2. Sample $\mathbf{T} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$ and return $\mathbf{T}$.

$\mathsf{A.Acc}_{\mathbf{T}}(R = (\mathbf{d}_i)_{i \in [0, N-1]} \in (\{0,1\}^{nk})^N)$**:** Let $\mathbf{u}_{\mathsf{bin}_L(j)} = \mathbf{u}_{j_1, \ldots, j_L} := \mathbf{d}_j$.
1. For each i from $L-1$ down to 1, for all $j \in [0, 2^i - 1]$ and $\mathsf{bin}_i(j) = (j_1, \ldots, j_i)$, compute $\mathbf{u}_{j_1, \ldots, j_i} := h_{\mathbf{T}}(\mathbf{u}_{j_1, \ldots, j_i, 0}, \mathbf{u}_{j_1, \ldots, j_i, 1})$.
2. Compute the root $\mathbf{u}_\epsilon := h_{\mathbf{T}}(\mathbf{u}_0, \mathbf{u}_1)$.

$\mathsf{A.GetRoot}_{\mathbf{T}}() \to \mathbf{u}_\epsilon \in \{0,1\}^{nk}$**:** Return the root node $\mathbf{u}_\epsilon$.

$\mathsf{A.ModLeaf}_{\mathbf{T}}\left((j^{(1)}, \ldots, j^{(S)}) \in [0, N)^S, (\mathbf{d}_{j^{(1)}}, \ldots, \mathbf{d}_{j^{(S)}}) (\{0,1\}^{nk})^S\right)$**:**
1. For each i from 1 to S, set $\mathbf{u}_{\mathsf{bin}_\ell(j^{(i)})} := \mathbf{d}_{j^{(i)}}$.
2. Recompute the intermediate nodes as in $\mathsf{A.Acc}$. We only need to update the parent node if any children have been modified.
3. Recompute the root $\mathbf{u}_\epsilon$.

$\mathsf{A.Witness}_{\mathbf{T}}(R, \mathbf{u}_\epsilon, \mathbf{d}) \to w \in \{0,1\}^L \times (\{0,1\}^{nk})^L$ **or** $\perp$**:** If $\mathbf{d} \notin R$, return $\perp$. Otherwise, determine $j \in [0, N)$ where $\mathbf{u}_{\mathsf{bin}_L(j)} = \mathbf{d}$. Let $(j_1, \ldots, j_L) = \mathsf{bin}_L(j)$. Output

$$w = ((j_1, \ldots, j_L), (\mathbf{u}_{\overline{j_1}}, \ldots, \mathbf{u}_{j_1, \ldots, j_{L-1}, \overline{j_L}})) \in \{0,1\}^L \times (\{0,1\}^{nk})^L.$$

$\mathsf{A.Verify}_{\mathbf{T}}(\mathbf{u}_\epsilon, \mathbf{d}, w) \to b \in \{0,1\}$**:** Parse $w = ((j_1, \ldots, j_L), (\mathbf{w}_1, \ldots, \mathbf{w}_L)) \in \{0,1\}^L \times (\{0,1\}^{nk})^L$. Let $\mathbf{v}_L := \mathbf{d}$. For i from $L-1$ down to 0, compute

$$\mathbf{v}_i := \begin{cases} h_{\mathbf{T}}(\mathbf{v}_{i+1}, \mathbf{w}_{i+1}) & \text{if } j_{i+1} = 0, \\ h_{\mathbf{T}}(\mathbf{w}_{i+1}, \mathbf{v}_{i+1}) & \text{if } j_{i+1} = 1. \end{cases}$$

Return $b = 1$ if $\mathbf{v}_0 = \mathbf{u}_\epsilon$. Otherwise, return $b = 0$.

Definition 16 ([1]). *The SIS problem* $\mathsf{SIS}^\infty_{n,m,q,\beta}$ *is defined as following: Given a uniformly random matrix* $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$*, find a short integer vector* $\mathbf{x} \neq \mathbf{0} \in \mathbb{Z}^m$ *such that* $\|\mathbf{x}\|_\infty \leq \beta$, $\mathbf{A} \cdot \mathbf{x} \equiv 0 \pmod q$.

As pointed out by [51], when $\beta = 1, q = \tilde{\mathcal{O}}(n), m = 2n\lfloor \log_2 q \rfloor$, $\mathsf{SIS}^\infty_{n,m,q,1}$ is at least as hard as the worst case problem $\mathsf{SIVP}_{\tilde{\mathcal{O}}(n)}$

Lemma 17 (Security of $\mathcal{ACC}_{\mathsf{llnw}}$ [51,52]). *$\mathcal{ACC}_{\mathsf{llnw}}$ is correct and secure assuming the hardness of problem* $\mathsf{SIS}^\infty_{n,m,q,1}$.

A.4 Definition of Public-Key Encryptions

We recall the definition of public-key encryption schemes, which can be found in various textbooks (e.g., see [46]), in the following Definition 18.

Definition 18. *A public-key encryption scheme (short for PKE) is a triple of randomized algorithms*

$$\mathcal{PKE} = (\mathsf{PKE.Gen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec}).$$

described as follows.

$\mathsf{PKE.Gen}(1^\lambda) \to (\mathsf{ek}, \mathsf{dk})$**:** *On input the security parameter 1^λ, the algorithm outputs encryption and decryption key $(\mathsf{ek}, \mathsf{dk})$. Optionally, the algorithm may also take a length parameter l to designate the message bit length. If l is not provided, the message bit length is a polynomial function $l = \mathsf{poly}(\lambda)$.*

$\mathsf{PKE.Enc}(\mathsf{ek}, \mathbf{m}; \mathbf{r}) \to \mathsf{ct}$**:** *On input the encryption key ek and message $\mathbf{m}$ of length l, the algorithm outputs a ciphertext ct. We let $\mathbf{r}$ denote the randomness used in the encryption algorithm. Sometimes, we will omit $\mathbf{r}$.*

$\mathsf{PKE.Dec}(\mathsf{dk}, \mathsf{ct}) \to \mathbf{m}$ **or** $\perp$**:** *On input the decryption key dk and a ciphertext ct, the algorithm, if successful, will output a message $\mathbf{m}$. Otherwise, decryption fails, and the output is $\perp$.*

Definition 19. *The PKE scheme $\mathcal{PKE}$ is correct if for any message $\mathbf{m}$ in the appropriate message space.*

$$\Pr\left[\mathsf{PKE.Dec}(\mathsf{dk}, \mathsf{ct}) = \mathbf{m} \middle| \begin{matrix}(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{PKE.Gen}(1^\lambda) \\ \mathsf{ct} \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}, \mathbf{m})\end{matrix}\right] \geq 1 - \mathsf{negl}(\lambda)$$

The scheme is perfectly correct if correctness holds with probability 1.

Definition 20. *We say that a PKE scheme satisfies semantic security, or indistinguishability under chosen plaintext attacks (IND-CPA), if any PPT adversary $\mathcal{A}$ cannot, except with negligible probability, distinguish the following experiments parameterized by a bit $b \in \{0, 1\}$. Experiment b is defined as follows.*

1. *Generate $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{PKE.Gen}(1^\lambda)$. Gives $\mathcal{A}$ the encryption key ek, and wait for $\mathcal{A}$ to produce 2 equal length message $(\mathbf{m}_0, \mathbf{m}_1)$.*
2. *Upon receiving $(\mathbf{m}_0, \mathbf{m}_1)$, creates ciphertext $\mathsf{ct}_b = \mathsf{PKE.Enc}(\mathsf{ek}, \mathbf{m}_b)$ and gives ct_b to $\mathcal{A}$.*
3. *Finally, $\mathcal{A}$ outputs accept or reject.*

That is,

$$|\Pr[\mathcal{A} \text{ accepts in Experiment } 0] - \Pr[\mathcal{A} \text{ accepts in Experiment } 1]| \leq \mathsf{negl}(\lambda).$$

Definition 21. *We say that a PKE scheme satisfies indistinguishability under chosen ciphertext attacks (IND-CCA1) if any PPT adversary $\mathcal{A}$ cannot, except with negligible probability, distinguish the following experiments parameterized by a bit $b \in \{0, 1\}$. Here Experiment b proceeds in 2 phases:*

Phase 1: Generate $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{PKE.Gen}(1^\lambda)$ and gives $\mathcal{A}$ the encryption key ek. In addition, it also provides $\mathcal{A}$ with a decryption oracle $\mathsf{O}_{\mathsf{Dec}(\mathsf{dk},\cdot)}$, one that runs the decryption algorithm (with the decryption key dk) on anything $\mathcal{A}$ provides. Wait for $\mathcal{A}$ to produce 2 equal length message $(\mathbf{m}_0, \mathbf{m}_1)$.

Phase 2: Upon receiving $(\mathbf{m}_0, \mathbf{m}_1)$, creates ciphertext $\mathsf{ct}_b = \mathsf{PKE.Enc}(\mathsf{ek}, \mathbf{m}_b)$ and gives ct_b to $\mathcal{A}$. During phase 2, $\mathcal{A}$ cannot access the decryption oracle $ODec$. Finally, $\mathcal{A}$ outputs accept or reject.

The PKE is IND-CCA1 secure if:

$$|\Pr[\mathcal{A} \text{ accepts in Experiment } 0] - \Pr[\mathcal{A} \text{ accepts in Experiment } 1]| \leq \mathsf{negl}(\lambda).$$

Definition 22. *We say that a PKE scheme satisfies indistinguishability under adaptive chosen ciphertext attacks (IND-CCA2) if any PPT adversary $\mathcal{A}$ cannot, except with negligible probability, distinguish the following experiments parameterized by a bit $b \in \{0, 1\}$.*

Experiment b is almost the same as that defined in Definition 21, except that in Phase 2, we also provide the adversary $\mathcal{A}$ with the decryption oracle $\mathsf{O}_{\mathsf{Dec}(\mathsf{dk},\cdot)}$. The oracle $\mathsf{O}_{\mathsf{Dec}(\mathsf{dk},\cdot)}$ in phase 2 would not produce the decryption of the challenge ciphertext ct_b. Otherwise, its functionality is the same as the decryption algorithm.

The PKE is IND-CCA2 secure if

$$|\Pr[\mathcal{A} \text{ accepts in Experiment } 0] - \Pr[\mathcal{A} \text{ accepts in Experiment } 1]| \leq \mathsf{negl}(\lambda).$$

A.5 Lindner-Peiker PKE

We recall Lindner-Peikert PKE [57]. This scheme is a modification of Regev's well-known encryption [78,79] based on the LWE assumption. Its main advantage is that it can be instantiated with smaller keys, ciphertexts, and encryption randomness without degrading concrete security on the LWE problem. Instead of relying on Leftover Hash Lemma [43] to mask the message bit, Lindner-Peikert PKE uses the LWE assumption twice, thus asymptotically reducing the overall sizes by a factor of $\log_2 q$ (Recall in Regev's encryption scheme, the LWE matrix needs at least $m \approx n \log_2 q$ columns to ensure safety).

In our construction, a user needs to generate NIZK proofs that the encryptions have been done "correctly" (in a sense to be specified later). We believe the small ciphertext sizes offered by Lindner-Peikert PKE will benefit the proof size. We now recall Lindner-Peikert PKE, denoted by

$$\mathcal{PKE}_{\mathsf{lp}} = (\mathsf{PKE.Gen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec}).$$

See Appendix A.4 for the definition of syntax and security of PKE.

First, similar to [57], we use a simple error-tolerant encoder/decoder. Specifically, let $\mathsf{encode} : \{0,1\} \to \mathbb{Z}_q$, $\mathsf{decode} : \mathbb{Z}_q \to \{0,1\}$, $t = \lfloor \frac{q}{4} \rfloor$ be $b_q = \mathsf{encode}(b) = b \cdot \lfloor \frac{q}{2} \rfloor$, $\mathsf{decode}(b_q) = 0$ if and only if $b_q \in \left[-\lfloor \frac{q}{4} \rfloor, \lfloor \frac{q}{4} \rfloor \right)$ and 1 otherwise. This encoding tolerates an error at most $t = \lfloor \frac{q}{4} \rfloor$, namely,

$$\forall b \in \{0,1\}, \forall e \text{ s.t. } |e| < t : \mathsf{decode}\left((\mathsf{encode}(b) + e) \mod q\right) = b.$$

We naturally extend the encode/decode to work with binary vectors. The descriptions of the algorithm of $\mathcal{PKE}_{\mathsf{lp}}$ are as follows.

$\mathsf{PKE.Gen}(1^\lambda, \ell \in \mathbb{N}) \to (\mathsf{ek}, \mathsf{dk})$**:** Here, ℓ is the length of messages to be encrypted. This algorithm works as follows.

1. Choose integer modulus $q > 0$, discrete Gaussian widths σ_k, σ_e, LWE rank $n_1, n_2 > 0$, error bound $t \leftarrow \lfloor \frac{q}{4} \rfloor$.
2. Sample a public matrix $\bar{\mathbf{A}} \xleftarrow{\$} \mathbb{Z}_q^{n_2 \times n_1}$.
3. Sample $\mathbf{R}_1 \leftarrow \mathcal{D}_{\mathbb{Z},\sigma_k}^{\ell \times n_1}$ and $\mathbf{R}_2 \leftarrow \mathcal{D}_{\mathbb{Z},\sigma_k}^{\ell \times n_2}$.
4. Compute $\mathbf{P} := (\mathbf{R}_1 - \mathbf{R}_2 \bar{\mathbf{A}}) \mod q \in \mathbb{Z}_q^{\ell \times n_1}$
5. Output encryption key $\mathsf{ek} = (n_1, n_2, \sigma_k, \sigma_e, \ell, \bar{\mathbf{A}}, t, \mathbf{P})$, and decryption key $\mathsf{dk} = \mathbf{R}_2$.

$\mathsf{PKE.Enc}(\mathsf{ek} = (n_1, n_2, \sigma_k, \sigma_e, \ell, \bar{\mathbf{A}} \in \mathbb{Z}_q^{n_2 \times n_1}, t, \mathbf{P} \in \mathbb{Z}_q^{\ell \times n_1}), \mathbf{m} \in \{0,1\}^\ell) \to \mathsf{ct}$**:** It works as follows.

1. Compute the encoding $\mathbf{m}_q := \mathsf{encode}(\mathbf{m}) \in \mathbb{Z}_q^\ell$.
2. Sample $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3) \leftarrow \mathcal{D}_{\mathbb{Z},\sigma_e}^{n_1} \times \mathcal{D}_{\mathbb{Z},\sigma_e}^{n_2} \times \mathcal{D}_{\mathbb{Z},\sigma_e}^{l}$.
3. Output $\mathsf{ct} = (\mathbf{c}_1 \| \mathbf{c}_2) := \begin{bmatrix} \bar{\mathbf{A}} & \mathbf{I}_{n_2} \\ \mathbf{P} & \mathbf{I}_\ell \end{bmatrix} \cdot (\mathbf{e}_1 \| \mathbf{e}_2 \| \mathbf{e}_3 + \mathbf{m}_q) \mod q$.

$\mathsf{PKE.Dec}(\mathsf{dk} = \mathbf{R}_2 \in \mathbb{Z}^{\ell \times n_2}, \mathsf{ct} = (\mathbf{c}_1 \| \mathbf{c}_2)) \to \mathbf{m}$**:** It works as follows.

1. Let $\mathbf{m}_q := (\mathbf{R}_2 \mathbf{c}_1 + \mathbf{c}_2) \mod q \in \mathbb{Z}_q^\ell$.
2. Output $\mathbf{m} \leftarrow \mathsf{decode}(\mathbf{m}_q)$. Note that

$$\mathbf{R}_2 \cdot \mathbf{c}_1 + \mathbf{c}_2 \equiv \mathbf{R}_1 \cdot \mathbf{e}_1 + \mathbf{R}_2 \cdot \mathbf{e}_2 + \mathbf{e}_3 + \mathbf{m}_q \equiv [\mathbf{R}_1 | \mathbf{R}_2 | \mathbf{I}_\ell] \cdot (\mathbf{e}_1 \| \mathbf{e}_2 \| \mathbf{e}_3) + \mathbf{m}_q \pmod q.$$

Correctness and Security. PKE $\mathcal{PKE}_{\mathsf{lp}}$ is correct with negligible decryption error and IND-CPA assuming the hardness of the decisional LWE problem.

Definition 23 ([78]). *The decisional LWE problem (with subgaussian secret)* $\mathsf{DLWE}_{n,m,q,\sigma}$ *asks to distinguish between LWE samples* $\mathbf{A}_{\sigma,\chi} \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$ *and those uniformly sampled from* $\mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$*, where* $\mathbf{A}_{\sigma,\chi} := (\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{e} \mod q), \mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{m \times n}, \mathbf{s} \leftarrow \mathcal{D}_{\mathbb{Z}^n,\sigma}, \mathbf{e} \leftarrow \mathcal{D}_{\mathbb{Z}^m,\sigma}$.

Recall from [78] that for sufficiently smooth modulus q, Decision-LWE is at least as hard as its search variant. Also, whenever $\sigma \geq 2\sqrt{n}$, Search-LWE is at least as hard as the worst case $\mathsf{SIVP}_{\tilde{\mathcal{O}}(\frac{nq}{\sigma})}$.

Lemma 24 (Correctness of $\mathcal{PKE}_{\mathsf{lp}}$ [57, Lemma 3.1]). *The probability of decryption error for each message bit is bounded above by* δ*, as long as*

$$\sigma_e \cdot \sigma_k \leq \frac{\sqrt{2}\pi}{c} \cdot \frac{t}{\sqrt{(n_1 + n_2)\ln(\frac{2}{\delta})}}$$

where c *above depends only on* $n_1 + n_2$ *and is between* $(1, 2)$ *for reasonable choices of* n_1, n_2.

Lemma 25 (Security of $\mathcal{PKE}_{\mathsf{lp}}$ [57, Lemma 3.2]). *The encryption scheme* $\mathcal{PKE}_{\mathsf{lp}}$ *above is IND-CPA-secure assuming the hardness of both* $\mathsf{DLWE}_{n_2,n_1,q,\sigma_k}$ *and* $\mathsf{DLWE}_{n_1,(n_2+\ell),q,\sigma_e}$.

A.6 Signature Schemes

We recall the definition of (ordinary) signatures, which can be found in various textbooks (e.g., see [46]), in the following Definition 26.

Definition 26. *A signature scheme is a triple of randomized algorithms*

$$\mathcal{SIG} = (\mathsf{S.Gen}, \mathsf{S.Sign}, \mathsf{S.Verify})$$

where

$\mathsf{S.Gen}(1^\lambda) \to (\mathsf{sk}, \mathsf{vk})$**:** *On input security parameter* 1^λ*, outputs signing key and verification key* sk, vk*.*
$\mathsf{S.Sign}(\mathsf{sk}, \mathbf{m}) \to \mathsf{sig}$**:** *On input signing key* sk *and message* $\mathbf{m}$*, outputs a signature* sig*.*
$\mathsf{S.Verify}(\mathsf{vk}, \mathbf{m}, \mathsf{sig}) \to \{0,1\}$**:** *On input the verification key* sk*, message signature pair* $(\mathbf{m}, \mathsf{sig})$*, outputs* 1 *if the algorithm accepts and* 0 *if it rejects.*

Definition 27. *The signature scheme* $\mathcal{SIG}$ *is correct if, for all message* $\mathbf{m}$ *in the appropriate message space, it holds that*

$$\Pr\left[\mathsf{S.Verify}(\mathsf{vk}, \mathbf{m}, \mathsf{sig}) = 1 \middle| \begin{matrix}(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{S.Gen}(1^\lambda) \\ \mathsf{sig} \leftarrow \mathsf{S.Sign}(\mathsf{sk}, \mathbf{m})\end{matrix}\right] \geq 1 - \mathsf{negl}(\lambda).$$

The scheme is perfectly correct if correctness holds with probability 1*.*

Definition 28. *We say a signature scheme satisfies universally existential unforgeability under chosen message attacks (EUF-CMA) if any PPT adversary* $\mathcal{A}$ *cannot win the following game except with negligible probability.*

1. *Generates* $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{S.Gen}(1^\lambda)$ *and gives the verification key* vk *to* $\mathcal{A}$*.*
2. *In addition, provides* $\mathcal{A}$ *with a signing oracle* OSign*.* OSign *does as follows.*
   (a) *Before any queries, it initializes an empty list* $\mathsf{QUERY} = \emptyset$*.*
   (b) *Each time* $\mathcal{A}$ *submits a signing query on message* $\mathbf{m}$*, the oracle runs* $\mathsf{sig} \leftarrow \mathsf{S.Sign}(\mathsf{sk}, \mathbf{m})$*. It appends* $(\mathbf{m}, \mathsf{sig})$ *to* QUERY *and returns* sig *to* $\mathcal{A}$*.*
3. *In the end,* $\mathcal{A}$ *outputs a pair* $(\mathbf{m}^*, \mathsf{sig}^*)$*.*

We say that $\mathcal{A}$ *wins the EUF-CMA game if with non-negligible probability,*

$$(\mathbf{m}^*, \mathsf{sig}^*) \notin \mathsf{QUERY} \wedge \mathsf{S.Verify}(\mathsf{vk}, \mathbf{m}^*, \mathsf{sig}^*) = 1.$$

A.7 JRS Oblivious Signing Protocol (Detailed)

We recall the oblivious signing interactive protocol $\Pi_{\mathsf{obl\text{-}sign}}$ from [45] in the following Figure 5. It is a more detailed description than in Section 5.2.

Recall that in Section 5.2, the signer (played by SA) receives a commitment to a private message $\mathbf{m}$ (known by the user) and transforms it into the commitment to a longer message $(\mathbf{m}\|\mathbf{m}')$ where $\mathbf{m}'$ is known by both the user and the signer.

Remark 29. In Figure 5, $\Gamma = (\mathsf{tag}, \mathbf{v}')$ can be transformed into a signature to $(\mathbf{m}\|\mathbf{m}')$ by computing $\mathbf{v} := \mathbf{v}' - (\mathbf{r}'\|\mathbf{0}^{m_2}) = (\mathbf{v}'_1\|\mathbf{v}'_2) - (\mathbf{r}'\|\mathbf{0}^{m_2}) = (\mathbf{v}'_1 - \mathbf{r}'\|\mathbf{v}'_2)$. The signature is hence $(\mathsf{tag}, \mathbf{v})$ and is verified by applying step 3 of $\Pi_{\mathsf{obl\text{-}sign}}$.

Remark 30. The signature scheme requires a pre-image Gaussian sampling algorithm $\mathsf{SampleD}$, such as the one given by Micciancio and Peikert [64] (recalled in [45, Lemma 2.6]). Using notations from Figure 5, we require that when given a matrix $[\mathbf{A}|\mathsf{tag} \cdot \mathbf{G}_{\mathsf{obl}} - \mathbf{AR}]$, a target $\mathbf{u} \in \mathbb{Z}^{n'}$ and a sufficiently large pre-image Gaussian width σ, $\mathsf{SampleD}$ outputs a vector $\mathbf{v}$ statistically closed to $\mathcal{D}_{\mathbb{Z}^{m_1+m_2},\sigma}$, conditioned on $[\mathbf{A}|\mathsf{tag} \cdot \mathbf{G}_{\mathsf{obl}} - \mathbf{AR}] \cdot \mathbf{v} \equiv \mathbf{u} \mod q$.

Setup Algorithm for JRS Oblivious Signing. In Figure 6, we present the setup algorithm $\Pi_{\mathsf{obl}}.\mathsf{Setup}$ for JRS oblivious signing protocol [45].

Security of JRS Oblivious Signing Protocol $\Pi_{\mathsf{obl\text{-}sign}}$. The discussion is according to the below definition and lemmas.

JRS Oblivious Signing Protocol $\Pi_{\mathsf{obl\text{-}sign}}^{\mathsf{U}(\mathbf{m}),\mathsf{S}(\mathbf{R},F,\mathsf{ctr})}(\mathsf{pp}_{\mathsf{obl}}, \mathbf{m}')$.

Public Input: The public input is the following tuple.

$$\mathsf{pp}_{\mathsf{obl}} = (n', k, q, q', Q, m_1, m_2, m_3, m_3', \sigma, \sigma_1, \ldots, \sigma_4, \mathbf{A}, \mathbf{B}, \mathbf{D}_1, \mathbf{D}_2, \mathbf{G}_{\mathsf{obl}}, \mathbf{u}),$$

where $\mathbf{B} = \mathbf{AR} \in \mathbb{Z}_q^{n' \times m_2}$, $\mathbf{G}_{\mathsf{obl}} = \mathbf{I}_{n'} \otimes (1, 2, \ldots, 2^k)^\top$, where $k = \lfloor \log_2 q \rfloor$, is the gadget matrix, and a public message $\mathbf{m}' \in \{0,1\}^{m_3'}$.
User U's Input: Message $\mathbf{m} \in \{0,1\}^{m_3}$.
Signer S's Inputs: Trapdoor $\mathbf{R}$, tag sampling function $F : [Q] \to \mathbb{Z}_{q'}^\times$, and counter ctr.
Execution:

1. Let $\mathbf{D} = [\mathbf{D}_1 | \mathbf{D}_2]$. This phase is done by user U as follows.
   (a) Sample $\mathbf{r}' \leftarrow \mathcal{D}_{\mathbb{Z}^{m_1}, \sigma_3}$, $\mathbf{c} := \mathbf{A} \cdot \mathbf{r}' + \mathbf{D}_1 \cdot \mathbf{m} \mod q$.
   (b) User U computes NIZK proof $\mathsf{NIZK}_{\mathsf{obl\text{-}sign}}$ showing that U knows $\mathbf{r}'$ and $\mathbf{m}$ such that

$$\mathcal{R}_{\mathsf{obl\text{-}sign}} = \left\{ (\mathbf{A}, \mathbf{D}_1, m_1, m_3, \sigma_3, \mathbf{c};\ \mathbf{m}, \mathbf{r}') \,\middle|\, \begin{array}{l} \mathbf{m} \in \{0,1\}^{m_3} \\ \wedge \|\mathbf{r}'\|_\infty \le \sigma_3 \log_2 m_1 \\ \wedge \mathbf{c} = \mathbf{A} \cdot \mathbf{r}' + \mathbf{D}_1 \cdot \mathbf{m} \mod q \end{array} \right\}. \tag{2}$$

   The transcript of this step is $\mathsf{trans} = (\mathbf{c}, \mathsf{NIZK}_{\mathsf{obl\text{-}sign}})$. Then, U sends trans to S.
2. Upon receiving $\mathsf{trans} = (\mathbf{c}, \mathsf{NIZK}_{\mathsf{obl\text{-}sign}})$, the signer S works as follows.
   (a) Verify $\mathsf{NIZK}_{\mathsf{obl\text{-}sign}}$ to check whether $(\mathbf{A}, \mathbf{D}_1, m_1, m_3, \sigma_3, \mathbf{c}) \in \mathcal{L}(\mathcal{R}_{\mathsf{obl\text{-}sign}})$. Abort and output 0 if verification fails.
   (b) $\mathbf{r}'' \leftarrow \mathcal{D}_{\mathbb{Z}^{m_1}, \sigma_4}$, $\mathbf{c}' := \mathbf{c} + \mathbf{A} \cdot \mathbf{r}'' + \mathbf{D}_2 \cdot \mathbf{m}' \mod q$.
   (c) $\mathsf{tag} \leftarrow F(\mathsf{ctr})$, $\mathbf{v}' := \mathsf{SampleD}(\mathbf{R}, \mathbf{A}, \mathsf{tag}, \mathbf{G}_{\mathsf{obl}}, \mathbf{u} + \mathbf{c}', \sigma) - (\mathbf{r}'' \| \mathbf{0}^{m_2}) \in \mathbb{Z}^{m_1 + m_2}$.
   (d) Send $\Gamma := (\mathsf{tag}, \mathbf{v}')$ to U and set $\mathsf{ctr} := \mathsf{ctr} + 1$.
3. This phase is done by the user U to verify correctness of the pair $\Gamma = (\mathsf{tag}, \mathbf{v}')$ as follows.
   (a) Parse $\mathbf{v}' = (\mathbf{v}_1' \| \mathbf{v}_2')$, where $\mathbf{v}_1' \in \mathbb{Z}^{m_1}, \mathbf{v}_2' \in \mathbb{Z}^{m_2}$
   (b) Let $\mathbf{A}_{\mathsf{tag}} = [\mathbf{A} | \mathsf{tag} \cdot \mathbf{G}_{\mathsf{obl}} - \mathbf{B}]$, where $\mathbf{G}_{\mathsf{obl}} = \mathbf{I}_{n'} \otimes (1, 2, \ldots, 2^k)^\top$. Return 1 if

$$\begin{cases} \mathbf{A}_{\mathsf{tag}} \cdot (\mathbf{v}' - (\mathbf{r}' \| \mathbf{0}^{m_2})) = \mathbf{u} + \mathbf{D} \cdot (\mathbf{m} \| \mathbf{m}') \mod q, \\ \mathsf{tag} \in \mathbb{Z}_{q'}^\times, \|\mathbf{v}_1' - \mathbf{r}'\|_\infty \le \sigma_1 \log_2 m_1, \|\mathbf{v}_2'\|_\infty \le \sigma \log_2 m_2. \end{cases}$$

   (c) Otherwise, return 0.

Fig. 5. Protocol $\Pi_{\mathsf{obl\text{-}sign}}^{\mathsf{U}(\mathbf{m}),\mathsf{S}(\mathbf{R},F,\mathsf{ctr})}(\mathsf{pp}_{\mathsf{obl}}, \mathbf{m}')$ [45].

Definition 31 (SIS Assumption [45]). *For $\beta_2 \ge \beta_\infty \ge 1$, $\mathsf{SIS}_{n,m,q,\beta_\infty,\beta_2}^{\infty,2}$ is defined as following: Given a uniformly random matrix $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$, find a short integer vector $\mathbf{x} \ne \mathbf{0} \in \mathbb{Z}^m$, $\|\mathbf{x}\|_\infty \le \beta_\infty$, $\|\mathbf{x}\| \le \beta_2$ such that $\mathbf{A} \cdot \mathbf{x} \equiv 0 \pmod q$*

Lemma 32 (Unforgeability of $\Pi_{\mathsf{obl\text{-}sign}}$, [45, Lemmas 3.2 and 3.3]). *When making at most Q signing queries, assume the adversary can forge a signature with probability δ. Let $\alpha^* = 1 + \sqrt{\pi \log_2 e} \cdot \sqrt{\log_2 \frac{1}{\delta}}$, t be the one controlling spectral norm gap, and $\mathsf{adv}_1 = \frac{\delta}{|\mathbb{Z}_q^\times| - Q}, \mathsf{adv}_2 = \delta^{\frac{\alpha^*}{\alpha^* - 1}} e^{-\alpha^* \pi} / Q$. Then there is an efficient solver that solves $\mathsf{SIS}_{n', m_1+1, q, \beta_\infty, \beta_2}^{\infty,2}$ with probability $\Omega(\mathsf{adv}_1)$, or solves $\mathsf{SIS}_{n', m_1, q, \beta_\infty', \beta_2'}^{\infty,2}$ with probability $\Omega(\mathsf{adv}_2)$, where*

$$\begin{aligned} \beta_\infty &= \sigma_1 \log_2 m_1 + m_2 \sigma \log_2 m_2 + m_3 + m_3', \\ \beta_\infty' &= 2\sigma_1 \log_2 m_1 + 2m_2 \sigma \log_2 m_2 + m_3 + m_3', \\ \beta_2 &= \sqrt{1 + (\sqrt{m_1} + \sqrt{m_2} + t)^2} \cdot \sqrt{m_1 (\sigma_1 \log_2 m_1)^2 + m_2 (\sigma \log_2 m_2)^2} \\ &\quad + \min\{2\sqrt{m_1}, \sqrt{m_1} + \sqrt{m_3 + m_3' + t}\} \sqrt{m_3 + m_3'} + 1, \\ \beta_2' &= \sqrt{1 + (\sqrt{m_1} + \sqrt{m_2} + t)^2} \cdot \sqrt{\sigma_1^2 m_1 (1 + \log_2^2 m_1) + \sigma^2 m_2 (1 + \log_2^2 m_2)} \\ &\quad + \min\{2\sqrt{m_1}, \sqrt{m_1} + \sqrt{m_3 + m_3' + t}\} \sqrt{m_3 + m_3'}. \end{aligned} \tag{3}$$

Lemma 33 (Obliviousness of $\Pi_{\mathsf{obl\text{-}sign}}$, [45, Remark 5.1]). *The obliviousness of $\Pi_{\mathsf{obl\text{-}sign}}$ is guaranteed if the user's commitment* $\mathbf{c}$ *(c.f. Figure 5) is hiding.*

Setup Algorithm $\Pi_{\mathsf{obl}}.\mathsf{Setup}(1^\lambda)$ for $\Pi_{\mathsf{obl\text{-}sign}}$ in Figure 5.

Public Input: Security parameter 1^λ.
Execution:

1. Choose an integer $n' > 0$ for SIS-rank, $Q = \mathsf{poly}(\lambda)$ an upper bound on the number of signing queries, $q > Q$ a prime modulus. // In our scheme, we use the same prime modulus q as the lattice accumulator $\mathcal{ACC}_{\mathsf{IInw}}$
2. Let the tag space be $\mathbb{Z}_{q'}^\times$ where q' is selected such that $Q \leq q' \leq q$ (see Remark F.1 in the full version of [45] for a discussion about the choice of q'). Initialize private counter $\mathsf{ctr} := 0$. Choose an injective function $F : [Q] \to \mathbb{Z}_{q'}^\times$.
3. Let $k := \lfloor \log_2 q \rfloor + 1$. Define the gadget matrix $\mathbf{G}_{\mathsf{obl}} = \mathbf{I}_{n'} \otimes (1, 2, \ldots, 2^{k-1})^\top$.
4. Define the following Length parameters.
   (a) Let $m_1 \leftarrow \left\lceil \frac{(n' \cdot \log_2 q + \omega(\log_2 \lambda))}{\log_2 3} \right\rceil$ be the commitment randomness width, where $\omega(\cdot)$ here is the asymtptotic notation.
   (b) Let $m_2 := n' \cdot k$ be the gadget length.
   (c) Choose $m_3 > 0$ as the maximum bit-length of the message.
   (d) Let $m_3' := n \cdot k$ be the bit-length public message, which is also the bit-length of the Merkle Tree root (see Appendix A.3)
5. Define the following discrete Gaussian width parameters.
   (a) Choose the spectral norm gap and smoothing parameter $t > 0$ and $\eta_\epsilon(\mathbb{Z})$ for some $\epsilon \in (0, \frac{1}{2})$.
   (b) Let $\sigma \leftarrow \eta_\epsilon(\mathbb{Z}) \cdot \sqrt{7} \cdot \sqrt{1 + (\sqrt{m_1} + \sqrt{m_2} + t)^2}$.
   (c) Choose $\sigma_3 \geq \sqrt{2} \cdot \eta_\epsilon(\mathbb{Z}^{m_1})$ and
   $$\sigma_4 \geq \max\left\{ \sqrt{\left((\sqrt{m_1} + \sqrt{m_3 + m_3'} + t)\sqrt{m_3 + m_3'} + \sigma_3\sqrt{m_1}\right)^2 - \sigma^2}, \sqrt{2} \cdot \eta_\epsilon(\mathbb{Z}^{m_1}) \right\}.$$
   (d) Let $\sigma_2 := \sqrt{\sigma_3^2 + \sigma_4^2}$ and $\sigma_1 := \sqrt{\sigma_2^2 + \sigma^2}$.
6. Sample $\mathbf{D}_1 \xleftarrow{\$} \mathbb{Z}_q^{n' \times m_3}$, $\mathbf{D}_2 \xleftarrow{\$} \mathbb{Z}_q^{n' \times m_3'}$. Let $\mathbf{D} = [\mathbf{D}_1 | \mathbf{D}_2] \in \mathbb{Z}_q^{n' \times (m_3 + m_3')}$.
7. Sample $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n' \times m_1}$, $\mathbf{R} \xleftarrow{\$} \{-1, 0, 1\}^{m_1 \times m_2}$, $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_q^{n'}$.
8. Let $\mathbf{B} \leftarrow \mathbf{AR} \mod q \in \mathbb{Z}_q^{n' \times m_2}$

Output:

1. Return public tuple $\mathsf{pp}_{\mathsf{obl}}$ where
   $$\mathsf{pp}_{\mathsf{obl}} = \left(n', k, q, q', Q, m_1, m_2, m_3, m_3', \sigma, \sigma_1, \ldots, \sigma_4, \mathbf{A}, \mathbf{B}, \mathbf{D}_1, \mathbf{D}_2, \mathbf{G}_{\mathsf{obl}}, \mathbf{u}\right).$$
2. Give $(\mathbf{R}, F, \mathsf{ctr})$ to the signer.

Fig. 6. Algorithm $\Pi_{\mathsf{obl}}.\mathsf{Setup}(1^\lambda)$ [45].

A.8 Non-Interactive Zero-Knowledge Argument of Knowledge

In this section, we recall the definition of non-interactive zero-knowledge arguments of knowledge [15,35].

Definition 34 (Syntax). *Let $\mathcal{R} = \left\{ (s; w) \,\middle|\, \text{conditions capturing } (s, w) \right\}$ be an NP relation, $\mathcal{L}(\mathcal{R})$ the corresponding language. A Non-Interactive Zero-Knowledge Argument of Knowledge (short for NIZKAoK) of $\mathcal{R}$ is a tuple of algorithms*

$$\mathcal{NIZKAOK} = (\mathsf{NIZK.Setup}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Verify})$$

described as follows.

$\mathsf{NIZK.Setup}(1^\lambda) \to \mathsf{crs}$**:** *On input the security parameter* 1^λ*, the algorithm returns a common reference string* crs*.*

$\mathsf{NIZK.Prove}(\mathsf{crs}, x, w) \to \pi$**:** *On input* crs *from* $\mathsf{NIZK.Setup}$*,* $x \in \mathcal{L}(\mathcal{R})$ *with witness* w*, the algorithm returns a proof* π*.*

$\mathsf{NIZK.Verify}(\mathsf{crs}, x, \pi) \to \{0,1\}$**:** *On input* crs *from* $\mathsf{NIZK.Setup}$*, statement* x *and a* $\mathcal{NIZKAoK}$ *proof* π*, the algorithm returns a bit* $b \in \{0,1\}$*.*

Definition 35 (Completeness). *An* $\mathcal{NIZKAoK}$ *system is perfectly complete if* $\forall (x,w) \in \mathcal{R}$*:*

$$\Pr\left[\mathsf{NIZK.Verify}(\mathsf{crs}, x, \pi) = 1 \middle| \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda) \\ \pi \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, x, w) \end{array}\right] = 1.$$

Occasionally we allow the $\mathcal{NIZKAoK}$ *to have negligible completeness error* $\epsilon_c < \mathsf{negl}(\lambda)$*. In such cases we require that* $\mathsf{NIZK.Verify}$ *outputs 1 with probability* $> 1 - \epsilon_c$*.*

Definition 36 (Extractable Soundness). *An* $\mathcal{NIZKAoK}$ *system has extractable soundness if for all PPT adversary* $\mathcal{A}$*, there exists a PPT Extractor algorithm* $\mathsf{NIZK.Extract}(\mathsf{crs}, x, \pi)$ *such that*

$$\Pr\left[\begin{array}{l} \mathsf{NIZK.Verify}(\mathsf{crs}, x, \pi) = 1 \\ (x, w') \notin \mathcal{R} \end{array}\middle| \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda) \\ (x, \pi) \leftarrow \mathcal{A}(\mathsf{crs}) \\ w' \leftarrow \mathsf{NIZK.Extract}(\mathsf{crs}, x, \pi) \end{array}\right] < \mathsf{negl}(\lambda). \tag{4}$$

Definition 37 (Zero Knowledge). *The* $\mathcal{NIZKAoK}$ *has a simulator* $\mathsf{NIZK.Sim}$ *which takes as inputs the common reference string* crs*, a statement* $x \in \mathcal{L}(\mathcal{R})$*, an auxiliary input* aux *and produces a simulated proof* π'*. For all* $(x,w) \in \mathcal{R}$*,* (x,π) *is statistically indistinguishable from* (x,π')*, where*

$$(x,\pi) = (x, \mathsf{NIZK.Prove}(\mathsf{crs}, x, w)) \text{ where } \mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(\lambda) \text{ and}$$
$$(x,\pi') = (x, \mathsf{NIZK.Sim}(\mathsf{crs}, x, \mathsf{aux})) \text{ where } \mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(\lambda).$$

B Our Lattice-Based Instantiation of EFDGS (Detailed)

We discuss the overview of our construction of our lattice-based EFDGS in Appendix B.1. In Appendix B.2, we describe the construction. This construction requires some underlying NIZK proofs. Hence, in Appendix B.3, we discuss how to construct them. In Appendix B.4, we discuss the analysis of our construction.

B.1 Overview

We briefly discuss the construction of our lattice-based instantiation of EFDGS. Initially, we follow the strategy from [60,61,58] to organize the group as follows.

The membership of users in the group. Each user id has a pair of public and private keys $(\mathsf{pk_{id}}, \mathsf{sk_{id}})$. Let $n \in \mathbb{N}$, $k = \lfloor \log_2 q \rfloor$, and q is a sufficiently large prime. The relationship of this key pair is guaranteed through the hardness of the problem $\mathsf{SIS}^\infty_{n,2nk,q,1}$. In particular, $\mathsf{sk_{id}} \xleftarrow{\$} \{0,1\}^{2nk}$ and $\mathsf{pk_{id}} = \mathsf{bin}_k(\mathbf{T} \cdot \mathsf{sk_{id}}) \in \{0,1\}^{nk}$ where $\mathbf{T} \in \mathbb{Z}_q^{n \times 2nk}$ is a random matrix used for accumulating public keys, to be described soon. We hence use $\mathsf{reg}^\tau = (\mathsf{reg}_0^\tau, \ldots, \mathsf{reg}_{N-1}^\tau) \in (\{0,1\}^{nk})^N$ as the registry at epoch τ. Then, we use the LLNW accumulator $\mathcal{ACC}_{\mathsf{llnw}}$ (see Appendix A.3), parameterized by $\mathbf{T}$, to accumulate reg^τ into a single accumulated value $\mathbf{u}_\epsilon^\tau \in \{0,1\}^{nk}$. Recall that $\mathcal{ACC}_{\mathsf{llnw}}$ has an underlying structure in the form of a Merkle tree. Here, at epoch cur, we consider the root $\mathbf{u}_\epsilon^{\mathrm{cur}}$ of this Merkle tree as the public system information $\mathsf{info}^{\mathrm{cur}}$ of the entire system.

System authority SA *and tracing authority* TA*.* As described in Section 4, SA plays a role in enrolling and revoking users in the system. In our lattice-based EFDGS, SA has no specific key. On the other hand, TA has a pair of keys $(\mathsf{pk_{ta}}, \mathsf{sk_{ta}})$ for encrypting (by $\mathsf{pk_{ta}}$) and decrypting (by $\mathsf{sk_{ta}}$) the signer's identifier.

As the scheme requires, for a message M to be signed by a user of identifier id, this user must obtain a pre-signature for M from SA. However, to protect the privacy of M before being signed, SA is not allowed to know M, and the output pre-signature Γ does not leak any information about M. Therefore, we realize the pre-signature issuance mechanism for obtaining pre-signature by using the JRS oblivious signing protocol $\Pi_{\mathsf{obl\text{-}sign}}$ (c.f. Section 5.2 and Appendix A.7) as it allows to construct NIZK proofs on signature-massage pairs, whose the signatures are obtained from those on committed messages, without proving correct output

from random oracle heuristically realized as a hash function H, which is not a standard way for proving. We omit the details of this issue and refer to the discussion in [45]. Regarding its use in the construction, we employ steps 1 and 2 (c.f. Figure 5) for issuing pre-signatures (signatures on committed messages) while step 3 is for verifying pre-signatures and generating EFDGS signatures.

Our lattice-based EFDGS employs the following building blocks:

- LLNW accumulator $\mathcal{ACC}_{\mathsf{llnw}}$ (c.f. Section 5.1 and Appendix A.3),
- JRS oblivious signing protocol (c.f. Section 5.2 and Appendix A.7),
- Lindner-Peikert IND-CPA-secure PKE $\mathcal{PKE}_{\mathsf{lp}}$ (c.f. Appendix A.5), and
- NIZK proof systems for circuit satisfiability over $\mathbb{Z}_q$.

B.2 Description

We now describe our lattice-based EFDGS, following the syntax in Definition 1.

$\mathsf{Setup}(1^\lambda) \to (\mathsf{pp}, \mathsf{info}^{\mathrm{init}}, \mathsf{reg}^{\mathrm{init}}, \mathsf{sk}_{\mathsf{sa}}, \mathsf{sk}_{\mathsf{ta}})$**:** This algorithm works as follows:

1. Use LLNW accumulator $\mathcal{ACC}_{\mathsf{llnw}}$ (c.f. Section 5.1 and Appendix A.3) to setup initial system information as follows.
   (a) $\mathbf{T} = [\mathbf{T}_0|\mathbf{T}_1] \leftarrow \mathsf{A.Setup}(1^\lambda)$ and obtain parameters $n, k, q, N = 2^L$ for some positive integer L.
   (b) Set init $:= 0$ and cur $:=$ init.
   (c) Initialize $\mathsf{reg}^{\mathrm{init}} := (\underbrace{\mathbf{0}^{nk}, \ldots, \mathbf{0}^{nk}}_{N \text{ entries}}) \in (\{0,1\}^{nk})^N$.
   (d) Initialize the initial system information by running $\mathsf{A.Acc}_{\mathbf{T}}(\mathsf{reg}^{\mathrm{init}})$ and $\mathsf{info}^{\mathrm{init}} \leftarrow \mathsf{A.GetRoot}_{\mathbf{T}}()$.
2. For obtaining SA's public and private keys, do as follows.
   (a) Run $\Pi_{\mathsf{obl}}.\mathsf{Setup}(1^\lambda)$ (c.f. Figure 6). Obtain a public tuple

$$\mathsf{pp}_{\mathsf{obl}} = (n', k, q, q', Q, m_1, m_2, m_3, m_3', \sigma, \sigma_1, \ldots, \sigma_4, \mathbf{A}, \mathbf{B}, \mathbf{D}_1, \mathbf{D}_2, \mathbf{G}_{\mathsf{obl}}, \mathbf{u})$$

   where $\mathbf{A} \in \mathbb{Z}_q^{n' \times m_1}$, $\mathbf{B} \in \mathbb{Z}_q^{n' \times m_2}$, $\mathbf{D}_1 \in \mathbb{Z}_q^{n' \times m_3}$, $\mathbf{D}_2 \in \mathbb{Z}_q^{n' \times m_3'}$, and $\mathbf{u} \in \mathbb{Z}_q^{n'}$, while private tuple $(\mathbf{R}, F, \mathsf{ctr})$ is sent to SA, who plays the role of signer in $\Pi_{\mathsf{obl\text{-}sign}}$ (c.f. Figure 5).
   (b) Denote $(\mathsf{pk}_{\mathsf{sa}}, \mathsf{sk}_{\mathsf{sa}}) = (\mathsf{pp}_{\mathsf{obl}}, (\mathbf{R}, F, \mathsf{ctr}))$.
3. Set up encryption scheme $\mathcal{PKE}_{\mathsf{lp}}$.
   (a) $(\mathsf{pk}_{\mathsf{ta}}, \mathsf{sk}_{\mathsf{ta}}) = (\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{PKE.Gen}(1^\lambda, nk)$.
4. Return as follows.
   (a) Let $\mathsf{gpk} = (\mathbf{T}, \mathsf{pp}_{\mathsf{obl}}, \mathsf{pk}_{\mathsf{sa}}, \mathsf{pk}_{\mathsf{ta}})$. Return public $\mathsf{pp} = (\mathcal{ID}, \mathcal{T}, \mathcal{M}, \mathsf{gpk})$ where
      - $\mathcal{ID} = \{0,1\}^L$,
      - $\mathcal{T}$ is the set of non-negative integers, and
      - $\mathcal{M} = \{0,1\}^{m_3}$. ($m_3$ is in $\mathsf{pp}_{\mathsf{obl}}$.)
   (b) Return $\mathsf{sk}_{\mathsf{sa}}$ and $\mathsf{sk}_{\mathsf{ta}}$ to SA and TA, respectively.

$\mathsf{Join}\ \langle \mathsf{U}, \mathsf{SA}(\mathsf{sk}_{\mathsf{sa}}) \rangle\ (\mathsf{pp}, \mathsf{info}^{\mathrm{cur}}, \mathsf{reg}^{\mathrm{cur}}, \mathsf{id}) \to (\mathsf{info}^{\mathrm{next}}, \mathsf{reg}^{\mathrm{next}}, \mathsf{pk}_{\mathsf{id}}, \mathsf{U}(\mathsf{sk}_{\mathsf{id}}))$**:**

1. U samples $\mathsf{sk}_{\mathsf{id}} \xleftarrow{\$} \{0,1\}^{2nk}$ and computes $\mathsf{pk}_{\mathsf{id}} := h_{\mathbf{T}}(\mathsf{sk}_{\mathsf{id}})$ where $h_{\mathbf{T}}$ is defined in (1). Then, U computes and sends to SA the tuple $(\mathsf{id}, \mathsf{pk}_{\mathsf{id}}, \mathsf{NIZK}_{\mathsf{enroll}})$ where $\mathsf{NIZK}_{\mathsf{enroll}}$ is a NIZK proof for

$$\mathcal{R}_{\mathsf{enroll}} = \left\{ (\mathsf{pk}_{\mathsf{id}}; \mathsf{sk}_{\mathsf{id}}) \,\middle|\, \mathsf{pk}_{\mathsf{id}} = h_{\mathbf{T}}(\mathsf{sk}_{\mathsf{id}}) \right\}. \tag{5}$$

2. Upon receiving $(\mathsf{id}, \mathsf{pk}_{\mathsf{id}}, \mathsf{NIZK}_{\mathsf{enroll}})$ from U, if SA agrees for U to be a valid user in the system, then SA works as follows.
   (a) If $\mathsf{reg}_{\mathsf{id}}^{\mathrm{cur}} \neq \mathbf{0}^{nk}$ or $\mathsf{NIZK}_{\mathsf{enroll}}$ is invalid, then return $\perp$ and abort.
   (b) Set next $:=$ cur $+ 1$, $\mathsf{reg}^{\mathrm{next}} := \mathsf{reg}^{\mathrm{cur}}$, and $\mathsf{info}^{\mathrm{next}} := \mathsf{info}^{\mathrm{cur}}$.
   (c) Update the registry by assigning $\mathsf{reg}_{\mathsf{id}}^{\mathrm{next}} := \mathsf{pk}_{\mathsf{id}}$.
   (d) Update the system information by computing $\mathsf{A.ModLeaf}_{\mathbf{T}}(\mathsf{id}, \mathsf{pk}_{\mathsf{id}})$ and setting the system information $\mathsf{info}^{\mathrm{next}} := \mathsf{A.GetRoot}_{\mathbf{T}}()$.
   (e) Move to the next epoch by setting cur $:=$ next.

$\mathsf{Update}(\mathsf{pp}, \mathsf{info}^{\mathrm{cur}}, \mathsf{reg}^{\mathrm{cur}}, \mathsf{sk}_{\mathsf{sa}}, \mathcal{S}) \to (\mathsf{info}^{\mathrm{next}}, \mathsf{reg}^{\mathrm{next}})$: SA runs as follows.

1. Parse $\mathcal{S} = \{\mathsf{id}_1, \ldots, \mathsf{id}_S\} \subseteq [0, N)$.
2. Set next $:=$ cur $+ 1$, $\mathsf{reg}^{\mathrm{next}} := \mathsf{reg}^{\mathrm{cur}}$, and $\mathsf{info}^{\mathrm{next}} := \mathsf{info}^{\mathrm{cur}}$.
3. Update the registry by setting $\mathsf{reg}_{\mathsf{id}_i}^{\mathrm{next}} := \mathbf{0}^{nk}$ for $i \in [S]$.
4. Run $\mathsf{A.ModLeaf}_{\mathbf{T}}((\mathsf{id}_1, \ldots \mathsf{id}_S), (\mathbf{0}^{nk})^S)$ and set $\mathsf{info}^{\mathrm{next}} := \mathsf{A.GetRoot}_{\mathbf{T}}()$.

5. Move to the next epoch by setting cur := next.

$\mathsf{PreSign}\langle \mathsf{U}(\mathbf{m}), \mathsf{SA}(\mathsf{sk}_{\mathsf{sa}})\rangle(\mathsf{pp}, \mathsf{info}^{\mathrm{cur}}) \to (\Gamma, \mathsf{trans}, \mathsf{U}(\mathsf{aux}))$**:** This works as follows.

1. Both parties run steps 1 and 2 of protocol $\Pi^{\mathsf{U}(\mathbf{m}),\mathsf{SA}(\mathbf{R},F,\mathsf{ctr})}_{\mathsf{obl\text{-}sign}}(\mathsf{pp}_{\mathsf{obl}}, \mathsf{info}^{\mathrm{cur}})$ (c.f. Figure 5) to obtain $\Gamma = (\mathsf{tag}, \mathbf{v}')$ where U and SA respectively play the roles of U and S in Figure 5.
2. User U obtains $\mathsf{aux} := \mathbf{r}'$ where $\mathbf{r}'$ is the randomness generated in step 1 of $\Pi^{\mathsf{U}(\mathbf{m}),\mathsf{S}(\mathbf{R},F,\mathsf{ctr})}_{\mathsf{obl\text{-}sign}}(\mathsf{pp}_{\mathsf{obl}}, \mathsf{info}^{\mathrm{cur}})$.
3. The full transcript of the interaction is set to be $\mathsf{trans} = (\mathbf{c}, \mathsf{NIZK}_{\mathsf{obl\text{-}sign}})$ obtained in step 1 of $\Pi_{\mathsf{obl\text{-}sign}}$ (c.f. Figure 5). Then, return public (Γ, trans).
4. If U with identifier id is active with respect to $\mathsf{info}^{\mathrm{cur}}$, U may need to remember w, computed from $w \leftarrow \mathsf{A.Witness}_{\mathbf{T}}(\mathsf{reg}^{\mathrm{cur}}, \mathsf{info}^{\mathrm{cur}}, \mathsf{id})$ (see Appendix A.3), by storing w in some private storage of U, for later running Sign requiring U to provide w for U's membership at the respective system information.

$\mathsf{PreVerify}(\mathsf{pp}, \mathsf{info}^{\tau}, \mathbf{m}, \mathsf{aux}, \Gamma) \to \{0,1\}$**:** User U parses $\mathsf{aux} = \mathbf{r}'$ and does:

1. Run step 3 of protocol $\Pi^{\mathsf{U}(\mathbf{m}),\mathsf{SA}(\mathbf{R},F,\mathsf{ctr})}_{\mathsf{obl\text{-}sign}}(\mathsf{pp}_{\mathsf{obl}}, \mathsf{info}^{\tau})$ (c.f. Figure 5) with respect to inputs $\Gamma = (\mathsf{tag}, \mathbf{v}')$ and $\mathbf{r}'$.
2. If the check in step 3 of $\Pi^{\mathsf{U}(\mathbf{m}),\mathsf{SA}(\mathbf{R},F,\mathsf{ctr})}_{\mathsf{obl\text{-}sign}}(\mathsf{pp}_{\mathsf{obl}}, \mathsf{info}^{\tau})$ is invalid, return 0. Otherwise, return 1.

$\mathsf{Sign}(\mathsf{pp}, \mathsf{info}^{\tau}, \mathsf{sk}_{\mathsf{id}}, \mathbf{m}, \mathsf{aux}, \Gamma) \to \Sigma$**:** This algorithm runs by user U as follows.

1. Parse $\mathsf{pp} = (\mathcal{ID}, \mathcal{T}, \mathcal{M}, \mathsf{gpk})$, $\mathsf{gpk} = (\mathbf{T}, \mathsf{pp}_{\mathsf{obl}}, \mathsf{pk}_{\mathsf{sa}}, \mathsf{pk}_{\mathsf{ta}})$, and
$$\mathsf{pp}_{\mathsf{obl}} = (n', k, q, q', Q, m_1, m_2, m_3, m_3', \sigma, \sigma_1, \ldots, \sigma_4, \mathbf{A}, \mathbf{B}, \mathbf{D}_1, \mathbf{D}_2, \mathbf{G}_{\mathsf{obl}}, \mathbf{u}).$$
2. Parse $\mathsf{aux} = \mathbf{r}'$, $\Gamma = (\mathsf{tag}, \mathbf{v}')$, and $\mathbf{v}' = (\mathbf{v}_1' \| \mathbf{v}_2')$ where $\mathbf{v}_1' \in \mathbb{Z}_q^{m_1}$ and $\mathbf{v}_2' \in \mathbb{Z}_q^{m_2}$. Let $\mathbf{A}_{\mathsf{tag}} := [\mathbf{A} | \mathsf{tag} \cdot \mathbf{G}_{\mathsf{obl}} - \mathbf{B}]$ as in $\Pi_{\mathsf{obl\text{-}sign}}$ (c.f. Figure 5).
3. U additionally provides $w = (\mathsf{id}, (\mathbf{w}_1, \ldots, \mathbf{w}_L)) \in \{0,1\}^L \times (\{0,1\}^{nk})^L$ as a witness for U's membership with respect to info^{τ}, i.e., $\mathsf{A.Verify}_{\mathbf{T}}(\mathsf{info}^{\tau}, \mathsf{pk}_{\mathsf{id}}, w) = 1$. As noted in PreSign above, U needs to store w in advance. Otherwise, if witness for U's membership at info^{τ} is lost, U loses the ability to run this algorithm, namely, Sign with respect to system information info^{τ}.
4. Encrypt the identifier by running $\mathsf{ct} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_{\mathsf{ta}}, \mathsf{id}; \mathbf{r})$. Here $\mathbf{r} \in \{0,1\}^{\eta}$ denotes a binary string representing the encryption randomness.
5. Compute a NIZK proof $\mathsf{NIZK}_{\mathsf{sig}}$ for the relation $\mathcal{R}_{\mathsf{sig}}$ in (6). Recall that $h_{\mathbf{T}}(\mathbf{x}) = \mathsf{bin}_k(\mathbf{T} \cdot \mathbf{x}) \in \{0,1\}^{nk}$ is the hash function introduced in (1).

$$\mathcal{R}_{\mathsf{sig}} = \left\{ (\mathsf{pp}, \mathbf{m}, \mathsf{ct};\ \Gamma, \mathsf{pk}_{\mathsf{id}}, \mathsf{sk}_{\mathsf{id}}, w, \mathsf{info}^{\tau}, \mathbf{r}', \mathbf{r}, \mathsf{tag}, \mathbf{A}_{\mathsf{tag}}) \,\middle|\, (7), (8), \text{and } (9) \text{ are satisfied} \right\}, \tag{6}$$

$$\begin{cases} \mathbf{A}_{\mathsf{tag}} = [\mathbf{A} | \mathsf{tag} \cdot \mathbf{G}_{\mathsf{obl}} - \mathbf{B}], \\ \mathbf{A}_{\mathsf{tag}} \cdot \mathbf{v}' - \mathbf{A} \cdot \mathbf{r}' \equiv \mathbf{u} + \mathbf{D}_1 \cdot \mathbf{m} + \mathbf{D}_2 \cdot \mathsf{info}^{\tau} \pmod q, \\ \mathsf{tag} \in \mathbb{Z}_{q'}^{\times}, \|\mathbf{r}'\|_{\infty} \le \sigma_3 \log_2 m_1, \\ \|\mathbf{v}_1' - \mathbf{r}'\|_{\infty} \le \sigma_1 \log_2 m_1, \|\mathbf{v}_2'\|_{\infty} \le \sigma \log_2 m_2, \end{cases} \tag{7}$$

$$\begin{cases} \mathsf{pk}_{\mathsf{id}} \in \{0,1\}^{nk}, \mathsf{sk}_{\mathsf{id}} \in \{0,1\}^{2nk}, \mathsf{pk}_{\mathsf{id}} = h_{\mathbf{T}}(\mathsf{sk}_{\mathsf{id}}), \\ w = (\mathsf{id}, (\mathbf{w}_1, \ldots, \mathbf{w}_L)) \in \{0,1\}^L \times (\{0,1\}^{nk})^L, \\ \mathsf{A.Verify}\,(\mathsf{info}^{\tau}, \mathsf{pk}_{\mathsf{id}}, w) = 1, \end{cases} \tag{8}$$

$$\mathbf{r} \in \{0,1\}^{\eta}, \mathsf{ct} = \mathsf{PKE.Enc}(\mathsf{pk}_{\mathsf{ta}}, \mathsf{id}; \mathbf{r}) \text{ where } \mathsf{id} \text{ is in } w. \tag{9}$$

We briefly explain the constraints in (6) as follows. (7) is the pre-verification check, via PreVerify, to ascertain the correct use of info^{τ} authenticated for $\mathbf{m}$ in ExtPreSig. (8) is for ensuring membership in the system. (9) is for the correct encryption of the identifier.

6. Output the final signature $\Sigma := (\mathsf{ct}, \mathsf{NIZK}_{\mathsf{sig}})$.

$\mathsf{Verify}(\mathsf{pp}, \mathbf{m}, \Sigma) \to \{0,1\}$**:** This is run by any public verifier. It works as follows.

1. Parse $\Sigma = (\mathsf{ct}, \mathsf{NIZK}_{\mathsf{sig}})$.
2. Verify $\mathsf{NIZK}_{\mathsf{sig}}$ to check whether $(\mathsf{pp}, \mathbf{m}, \mathsf{ct}) \in \mathcal{L}(\mathcal{R}_{\mathsf{sig}})$. Return 1 if the verification passes. Otherwise, return 0.

$\mathsf{Trace}(\mathsf{pp}, \mathsf{sk}_{\mathsf{ta}}, \mathbf{m}, \Sigma) \to (\mathcal{ID} \cup \{\perp\}, \Xi)$**:** This algorithm is run by the tracing authority TA as follows.

1. First, run $\mathsf{Verify}(\mathsf{pp}, \mathbf{m}, \Sigma)$. If verification fails, abort and output $\perp$.
2. Parse $\Sigma = (\mathsf{ct}, \mathsf{NIZK}_{\mathsf{sig}})$ and compute $\mathsf{id} \leftarrow \mathsf{PKE.Dec}(\mathsf{sk}_{\mathsf{ta}}, \mathsf{ct})$.
3. Compute a NIZK proof $\mathsf{NIZK}_{\mathsf{trace}}$ for the relation

$$\mathcal{R}_{\mathsf{trace}} = \left\{ (\mathsf{ct}, \mathsf{id}; \mathsf{sk}_{\mathsf{ta}}) \,\middle|\, \mathsf{id} = \mathsf{PKE.Dec}(\mathsf{sk}_{\mathsf{ta}}, \mathsf{ct}) \right\}. \tag{10}$$

4. Return (id, Ξ) where $\Xi = \mathsf{NIZK}_{\mathsf{trace}}$.

$\mathsf{Judge}(\mathsf{pp}, \mathbf{m}, \Sigma, \mathsf{id}, \Xi) \to \{0, 1, \perp\}$**:** This is run by any public verifier as follows.

1. First run $\mathsf{Verify}(\mathsf{pp}, \mathbf{m}, \Sigma)$. If verification fails, abort and output $\perp$.
2. Parse $\Xi = \mathsf{NIZK}_{\mathsf{trace}}$ and verify $\mathsf{NIZK}_{\mathsf{trace}}$ to check whether $(\mathsf{ct}, \mathsf{id}) \in \mathcal{L}(\mathcal{R}_{\mathsf{trace}})$. Return 1 if the verification passes. Otherwise, return 0.

B.3 Underlying NIZK Proofs

In this appendix, we discuss the instantiations of NIZK proofs appearing in the description of our lattice-based construction of EFDGS in Appendix B.2. The construction requires the following four NIZK proofs. (i) In Join, we need $\mathsf{NIZK}_{\mathsf{enroll}}$ for relation $\mathcal{R}_{\mathsf{enroll}}$ in (5). (ii) In $\mathsf{PreSign}$, we need $\mathsf{NIZK}_{\mathsf{obl\text{-}sign}}$ for relation $\mathcal{R}_{\mathsf{obl\text{-}sign}}$ in (2). (iii) In Sign, we need $\mathsf{NIZK}_{\mathsf{sig}}$ for relation $\mathcal{R}_{\mathsf{sig}}$ in (6). (iv) In Judge, we need $\mathsf{NIZK}_{\mathsf{trace}}$ for relation $\mathcal{R}_{\mathsf{trace}}$ in (10). These NIZK proofs can be instantiated by any NIZK proof system for circuit satisfiability. Specifically, the constraints specified in $\mathcal{R}_{\mathsf{enroll}}$, $\mathcal{R}_{\mathsf{obl\text{-}sign}}$, $\mathcal{R}_{\mathsf{sig}}$, and $\mathcal{R}_{\mathsf{trace}}$ can be captured by different arithmetic circuits by appropriately transforming those constraints into the ones suitable for arithmetic circuits with the following cases.

1. *Linear Constraints.* These constraints include multiplications between (public or private) matrices and vectors. Their constraints are well-known to be transformed into arithmetic circuits straightforwardly.
2. *Binary Constraints.* To show that $x \in \{0, 1\}$, we simply apply [41,34] to equivalently show that $x \cdot (1-x) = 0$.
3. *Inequality Constraints.* In the relations mentioned above, we need to prove inequalities of the form $0 \leq x < v$ for some public $v \in \mathbb{N}$ and private $x \in \mathbb{N}$. To this end, we apply Lemma 11 (c.f. Appendix A.1), which determines $k_v \in \mathbb{N}$ and $B_1, \ldots, B_{k_v} \in \mathbb{N}$ from v as follows. We can find $b_1, \ldots, b_{k_v}$ such that we can prove $x \in [0, v)$ by equivalently showing that $(b_1, \ldots, b_{k_v}) \in \{0, 1\}^{k_v}$ and $x = \sum_{i \in [k_v]} b_i \cdot B_i$. Proving $(b_1, \ldots, b_{k_v}) \in \{0, 1\}^{k_v}$ can be done by proving each b_i is binary, which is discussed in the case of binary constraints.
4. *Infinity Norm Bound Constraints.* Let $\mathbf{x} = (x_1, \ldots, x_\ell) \in \mathbb{Z}_q^\ell$ for some $\ell \in \mathbb{N}$. Assume that we would like to prove $\|\mathbf{x}\|_\infty \leq v$ for some $v \in \mathbb{N}$. Specifically, we would like to prove that $x_i \in [-v, v]$ for $i \in [\ell]$. This is done by equivalently proving that $x_i + v \in [0, 2v + 1)$ by applying the case of inequality constraints.
5. *Valid Accumulation in LLNW Accumulator.* Recall the algorithm $\mathsf{A.Verify}$ in Appendix A.3 that we need to run as follows to verify a correct accumulation. For i from $L - 1$ down to 0, compute

$$\mathbf{v}_i := \begin{cases} h_{\mathbf{T}}(\mathbf{v}_{i+1}, \mathbf{w}_{i+1}) = \mathsf{bin}_k(\mathbf{T} \cdot (\mathbf{v}_{i+1} \| \mathbf{w}_{i+1})) & \text{if } j_{i+1} = 0, \\ h_{\mathbf{T}}(\mathbf{w}_{i+1}, \mathbf{v}_{i+1}) = \mathsf{bin}_k(\mathbf{T} \cdot (\mathbf{w}_{i+1} \| \mathbf{v}_{i+1})) & \text{if } j_{i+1} = 1. \end{cases}$$

   Then, return $b = 1$ if $\mathbf{v}_0 = \mathbf{u}_\epsilon$. Otherwise, return $b = 0$. As suggested from [51,52], the above computation can be transformed into

$$\mathbf{G}_{\mathsf{acc}} \cdot \mathbf{v}_i = \mathbf{T} \cdot ((1 - j_{i+1}) \cdot (\mathbf{v}_{i+1} \| \mathbf{w}_{i+1}) + j_{i+1} \cdot \ (\mathbf{w}_{i+1} \| \mathbf{v}_{i+1})) \,.$$

   Hence, we can prove correct accumulation by proving correct computation of each $\mathbf{v}_i$ according to the above form and $j_i \in \{0, 1\}$ for $i \in [L]$.
6. *Correct Decryption of $\mathcal{PKE}_{\mathsf{lp}}$.* This is required by algorithm Trace to run $\mathsf{NIZK}_{\mathsf{trace}}$ (c.f. (10)) which requires proving correct decryption of PKE $\mathcal{PKE}_{\mathsf{lp}}$ (c.f. Appendix A.5). Notice that, in Appendix A.5, to decrypt, we need to run the decoding decode which maps an element b_q in $\left[-\left\lfloor \frac{q}{4} \right\rfloor, \left\lfloor \frac{q}{4} \right\rfloor\right)$ to $b = 0$ and remaining elements in $\mathbb{Z}_q$ to $b = 1$. Hence, we see that if and only if

$$b = 0 \iff b_q + \left\lfloor \frac{q}{4} \right\rfloor \in \left[0, 2 \cdot \left\lfloor \frac{q}{4} \right\rfloor\right), \text{ and} \tag{11}$$

$$b = 1 \iff b_q - \left\lfloor \frac{q}{4} \right\rfloor \in \left[0, q - 1 - 2 \cdot \left\lfloor \frac{q}{4} \right\rfloor\right). \tag{12}$$

   For case (11), by applying the discussion for inequality constraints above, we can construct a circuit $\mathcal{C}_0(b_q, w_0)$, where w_0 is some supporting witness, e.g., for parsing into binary, certifying whether b is decoded correctly from b_q. If the output of $\mathcal{C}_0$ is 1, it implies that $b = 0$ is decoded from b_q. Similarly, we can construct $\mathcal{C}_1(b_q, w_1)$ with a similar meaning for case (12). Then, we unify both cases by proving $(1 - b) \cdot \mathcal{C}_0(b_q, w_0) + b \cdot \mathcal{C}_1(b_q, w_1) = 1$.

B.4 Analysis

Theorem 38 (Correctness and Security of Lattice-Based EFDGS). *Our lattice-based EFDGS is correct, unforgeable, and private according to Definitions 5, 10, and 8, respectively, assuming that the underlying building blocks, namely, NIZK proofs, PKE $\mathcal{PKE}_{\mathsf{lp}}$, the JRS oblivious signing interactive protocol $\Pi_{\mathsf{obl\text{-}sign}}$ are complete and unforgeable, and the Ajtai commitment (in step 1 of $\Pi_{\mathsf{obl\text{-}sign}}$) is hiding and binding.*

Proof. The proof follows Lemmas 39, 40, and 43.

In brief, the correctness is straightforward. For privacy, we employ the obliviousness of $\Pi_{\mathsf{obl\text{-}sign}}$ to guarantee system information unlinkability. Anonymity is not compromised due to the IND-CPA of $\mathcal{PKE}_{\mathsf{lp}}$ and zero knowledge of NIZK proofs. For unforgeability, we first need to extract the witnesses behind the NIZK proofs (by using extractors or the extraction trapdoor in the simulated setup). Then, we can argue the unforgeability based on the security of the employed building blocks. □

Correctness. The correctness of our lattice-based EFDGS (c.f. Appendix B.2) follows the following Lemma 39.

Lemma 39. *Assume* SA *is honest. Then, our lattice-based EFDGS is correct according to Definition 5 assuming the completeness of the underlying NIZK proofs, the correctness of PKE $\mathcal{PKE}_{\mathsf{lp}}$ (c.f. Lemma 24), and the completeness of the JRS oblivious signing interactive protocol $\Pi_{\mathsf{obl\text{-}sign}}$.*

Proof. The proof is straightforward for both properties in Definition 5. In particular, protocol $\mathsf{PreSign}$ runs $\Pi_{\mathsf{obl\text{-}sign}}$ and steps 1 and 2 of $\mathsf{NIZK}_{\mathsf{obl\text{-}sign}}$, algorithm Sign runs $\mathsf{PKE.Enc}$ and $\mathsf{NIZK}_{\mathsf{sig}}$, and algorithm Trace runs $\mathsf{PKE.Dec}$ and $\mathsf{NIZK}_{\mathsf{trace}}$ as sub-routines. Hence, when running $\mathsf{PreVerify}$, Verify, and Judge, we require running verifications of the corresponding NIZK proofs, verification in step 3 of $\Pi_{\mathsf{obl\text{-}sign}}$, and the decryption of PKE $\mathcal{PKE}_{\mathsf{lp}}$. Therefore, when honestly following the computations as instructed, the algorithms $\mathsf{PreVerify}$, Verify, and Judge always return 1 while $\mathsf{id}' = \mathsf{id}$ is due to the correctness of the $\mathcal{PKE}_{\mathsf{lp}}$. □

Privacy. The privacy of our lattice-based EFDGS (c.f. Appendix B.2) follows the following Lemma 40.

Lemma 40 (Privacy of Lattice-Based EFDGS). *Our lattice-based EFDGS (c.f. Appendix B.2) is private (c.f. Definition 8) assuming that the Ajtai commitments (in step 1 of $\Pi_{\mathsf{obl\text{-}sign}}$) are hiding, the underlying NIZK proofs are zero-knowledge, and PKE $\mathcal{PKE}_{\mathsf{lp}}$ is IND-CPA (c.f. Lemma 25).*

Proof. The proof follows Lemmas 41 and 42. □

Lemma 41 (Message Hiding and System Information Unlinkability). *Our lattice-based EFDGS (c.f. Appendix B.2) is message-hiding and system-information-unlinkable with respect to oracle $\mathsf{OPreChal}_b$ (c.f. Definition 7), assuming that the Ajtai commitments (in step 1 of $\Pi_{\mathsf{obl\text{-}sign}}$) are (statistically) hiding, the underlying NIZK proofs, i.e $\mathsf{NIZK}_{\mathsf{obl\text{-}sign}}$ and $\mathsf{NIZK}_{\mathsf{sig}}$, are (statistically) zero-knowledge.*

Proof. Throughout we assume the adversary $\mathcal{A}$ only provides valid inputs (so that $\mathsf{OPreChal}_b$ would not return $\perp$). By following the specification of oracle $\mathsf{OPreChal}_b$ in Definition 7, for convenience purpose, we parse the following components as

$$(\Gamma^{(0)}, \mathsf{trans}^{(0)}) = (\Gamma^{(0)}, (\mathbf{c}^{(0)}, \mathsf{NIZK}^{(0)}_{\mathsf{obl\text{-}sign}})), \quad \Sigma^{(0)} = (\mathsf{ct}^{(0)}, \mathsf{NIZK}^{(0)}_{\mathsf{sig}}),$$
$$(\Gamma^{(1)}, \mathsf{trans}^{(1)}) = (\Gamma^{(1)}, (\mathbf{c}^{(1)}, \mathsf{NIZK}^{(1)}_{\mathsf{obl\text{-}sign}})), \quad \Sigma^{(1)} = (\mathsf{ct}^{(1)}, \mathsf{NIZK}^{(1)}_{\mathsf{sig}}).$$

Define the following sequence of games and denote by W_i the output of $\mathcal{A}$ in game G_i.

Game G_0: This is the original experiment $\mathsf{E}^{\mathsf{pre\text{-}unlk\text{-}0}}_{\mathcal{A}}(\lambda)$ from Figure 3 for $b = 0$. Hence,

$$\Pr[W_0 = 1] = \Pr[\mathsf{E}^{\mathsf{pre\text{-}unlk\text{-}0}}_{\mathcal{A}}(\lambda) = 1].$$

Notice that, in this game, the NIZK proofs are as follows.

$\mathsf{NIZK}^{(0)}_{\mathsf{obl\text{-}sign}}$ is a real proof for $(\mathbf{A}, \mathbf{D}_1, m_1, m_3, \sigma_3, \mathbf{c}^{(0)}; \mathbf{m}^{(0)}, \mathbf{r}'^{(0)}) \in \mathcal{R}_{\mathsf{obl\text{-}sign}}$,

$\mathsf{NIZK}^{(1)}_{\mathsf{obl\text{-}sign}}$ is a real proof for $(\mathbf{A}, \mathbf{D}_1, m_1, m_3, \sigma_3, \mathbf{c}^{(1)}; \mathbf{m}^{(1)}, \mathbf{r}'^{(1)}) \in \mathcal{R}_{\mathsf{obl\text{-}sign}}$,

$\mathsf{NIZK}^{(0)}_{\mathsf{sig}}$ is a real proof for

$(\mathsf{pp}, \mathbf{m}^{(0)}, \mathsf{ct}^{(0)}; \Gamma^{(0)}, \mathsf{pk}_{\mathsf{id}^{(0)}}, \mathsf{sk}_{\mathsf{id}^{(0)}}, w^{(0)}, \mathsf{info}^{\tau^{(0)}}, \mathbf{r}'^{(0)}, \mathbf{r}^{(0)}, \mathsf{tag}^{(0)}, \mathbf{A}_{\mathsf{tag}^{(0)}}) \in \mathcal{R}_{\mathsf{sig}}$,

$\mathsf{NIZK}^{(1)}_{\mathsf{sig}}$ is a real proof for

$(\mathsf{pp}, \mathbf{m}^{(1)}, \mathsf{ct}^{(1)}; \Gamma^{(1)}, \mathsf{pk}_{\mathsf{id}^{(1)}}, \mathsf{sk}_{\mathsf{id}^{(1)}}, w^{(1)}, \mathsf{info}^{\tau^{(1)}}, \mathbf{r}'^{(1)}, \mathbf{r}^{(1)}, \mathsf{tag}^{(1)}, \mathbf{A}_{\mathsf{tag}^{(1)}}) \in \mathcal{R}_{\mathsf{sig}}$.

Game G_1: This is the same as game G_0, except that the NIZK proof $\mathsf{NIZK}^{(0)}_{\mathsf{obl\text{-}sign}}$ is replaced by the simulated proof $\widetilde{\mathsf{NIZK}}^{(0)}_{\mathsf{obl\text{-}sign}}$ by running the respective simulator. Zero knowledge of NIZK proofs ensures that, except with negligible probability, $\mathcal{A}$ cannot distinguish G_1 from G_0, namely,

$$|\Pr[W_1 = 1] - \Pr[W_0 = 1]| \leq \mathsf{negl}(\lambda).$$

Notice that, in this game, the NIZK proofs are as follows.

$\widetilde{\mathsf{NIZK}}^{(0)}_{\mathsf{obl\text{-}sign}}$ is a simulated proof for $(\mathbf{A}, \mathbf{D}_1, m_1, m_3, \sigma_3, \mathbf{c}^{(0)}) \in \mathcal{L}(\mathcal{R}_{\mathsf{obl\text{-}sign}})$,
$\mathsf{NIZK}^{(1)}_{\mathsf{obl\text{-}sign}}$ is a real proof for $(\mathbf{A}, \mathbf{D}_1, m_1, m_3, \sigma_3, \mathbf{c}^{(1)}; \mathbf{m}^{(1)}, \mathbf{r}'^{(1)}) \in \mathcal{R}_{\mathsf{obl\text{-}sign}}$,
$\mathsf{NIZK}^{(0)}_{\mathsf{sig}}$ is a real proof for
$(\mathsf{pp}, \mathbf{m}^{(0)}, \mathsf{ct}^{(0)}; \Gamma^{(0)}, \mathsf{pk}_{\mathsf{id}^{(0)}}, \mathsf{sk}_{\mathsf{id}^{(0)}}, w^{(0)}, \mathsf{info}^{\tau^{(0)}}, \mathbf{r}'^{(0)}, \mathbf{r}^{(0)}, \mathsf{tag}^{(0)}, \mathbf{A}_{\mathsf{tag}^{(0)}}) \in \mathcal{R}_{\mathsf{sig}}$,
$\mathsf{NIZK}^{(1)}_{\mathsf{sig}}$ is a real proof for
$(\mathsf{pp}, \mathbf{m}^{(1)}, \mathsf{ct}^{(1)}; \Gamma^{(1)}, \mathsf{pk}_{\mathsf{id}^{(1)}}, \mathsf{sk}_{\mathsf{id}^{(1)}}, w^{(1)}, \mathsf{info}^{\tau^{(1)}}, \mathbf{r}'^{(1)}, \mathbf{r}^{(1)}, \mathsf{tag}^{(1)}, \mathbf{A}_{\mathsf{tag}^{(1)}}) \in \mathcal{R}_{\mathsf{sig}}$.

Game G_2: This is the same as game G_1, except that the NIZK proof $\mathsf{NIZK}^{(1)}_{\mathsf{obl\text{-}sign}}$ is replaced by the simulated proof $\widetilde{\mathsf{NIZK}}^{(1)}_{\mathsf{obl\text{-}sign}}$ by running the respective simulator. Zero knowledge of NIZK proofs ensures that, except with negligible probability, $\mathcal{A}$ cannot distinguish G_2 from G_1, namely,

$$|\Pr[W_2 = 1] - \Pr[W_1 = 1]| \leq \mathsf{negl}(\lambda).$$

Notice that, in this game, the NIZK proofs are as follows.

$\widetilde{\mathsf{NIZK}}^{(0)}_{\mathsf{obl\text{-}sign}}$ is a simulated proof for $(\mathbf{A}, \mathbf{D}_1, m_1, m_3, \sigma_3, \mathbf{c}^{(0)}) \in \mathcal{L}(\mathcal{R}_{\mathsf{obl\text{-}sign}})$,
$\widetilde{\mathsf{NIZK}}^{(1)}_{\mathsf{obl\text{-}sign}}$ is a simulated proof for $(\mathbf{A}, \mathbf{D}_1, m_1, m_3, \sigma_3, \mathbf{c}^{(1)}) \in \mathcal{L}(\mathcal{R}_{\mathsf{obl\text{-}sign}})$,
$\mathsf{NIZK}^{(0)}_{\mathsf{sig}}$ is a real proof for
$(\mathsf{pp}, \mathbf{m}^{(0)}, \mathsf{ct}^{(0)}; \Gamma^{(0)}, \mathsf{pk}_{\mathsf{id}^{(0)}}, \mathsf{sk}_{\mathsf{id}^{(0)}}, w^{(0)}, \mathsf{info}^{\tau^{(0)}}, \mathbf{r}'^{(0)}, \mathbf{r}^{(0)}, \mathsf{tag}^{(0)}, \mathbf{A}_{\mathsf{tag}^{(0)}}) \in \mathcal{R}_{\mathsf{sig}}$,
$\mathsf{NIZK}^{(1)}_{\mathsf{sig}}$ is a real proof for
$(\mathsf{pp}, \mathbf{m}^{(1)}, \mathsf{ct}^{(1)}; \Gamma^{(1)}, \mathsf{pk}_{\mathsf{id}^{(1)}}, \mathsf{sk}_{\mathsf{id}^{(1)}}, w^{(1)}, \mathsf{info}^{\tau^{(1)}}, \mathbf{r}'^{(1)}, \mathbf{r}^{(1)}, \mathsf{tag}^{(1)}, \mathbf{A}_{\mathsf{tag}^{(1)}}) \in \mathcal{R}_{\mathsf{sig}}$.

Game G_3: This is the same as game G_2, except that the NIZK proof $\mathsf{NIZK}^{(0)}_{\mathsf{sig}}$ is replaced by the simulated proof $\widetilde{\mathsf{NIZK}}^{(0)}_{\mathsf{sig}}$ by running the respective simulator. Zero knowledge of NIZK proofs ensures that, except with negligible probability, $\mathcal{A}$ cannot distinguish G_3 from G_2, namely,

$$|\Pr[W_3 = 1] - \Pr[W_2 = 1]| \leq \mathsf{negl}(\lambda).$$

Notice that, in this game, the NIZK proofs are as follows.

$\widetilde{\mathsf{NIZK}}^{(0)}_{\mathsf{obl\text{-}sign}}$ is a simulated proof for $(\mathbf{A}, \mathbf{D}_1, m_1, m_3, \sigma_3, \mathbf{c}^{(0)}) \in \mathcal{L}(\mathcal{R}_{\mathsf{obl\text{-}sign}})$,
$\widetilde{\mathsf{NIZK}}^{(1)}_{\mathsf{obl\text{-}sign}}$ is a simulated proof for $(\mathbf{A}, \mathbf{D}_1, m_1, m_3, \sigma_3, \mathbf{c}^{(1)}) \in \mathcal{L}(\mathcal{R}_{\mathsf{obl\text{-}sign}})$,
$\widetilde{\mathsf{NIZK}}^{(0)}_{\mathsf{sig}}$ is a simulated proof for $(\mathsf{pp}, \mathbf{m}^{(0)}, \mathsf{ct}^{(0)}) \in \mathcal{L}(\mathcal{R}_{\mathsf{sig}})$,
$\mathsf{NIZK}^{(1)}_{\mathsf{sig}}$ is a real proof for
$(\mathsf{pp}, \mathbf{m}^{(1)}, \mathsf{ct}^{(1)}; \Gamma^{(1)}, \mathsf{pk}_{\mathsf{id}^{(1)}}, \mathsf{sk}_{\mathsf{id}^{(1)}}, w^{(1)}, \mathsf{info}^{\tau^{(1)}}, \mathbf{r}'^{(1)}, \mathbf{r}^{(1)}, \mathsf{tag}^{(1)}, \mathbf{A}_{\mathsf{tag}^{(1)}}) \in \mathcal{R}_{\mathsf{sig}}$.

Game G_4: This is the same as game G_3, except that the NIZK proof $\mathsf{NIZK}^{(1)}_{\mathsf{sig}}$ is replaced by the simulated proof $\widetilde{\mathsf{NIZK}}^{(1)}_{\mathsf{sig}}$ by running the respective simulator. Zero knowledge of NIZK proofs ensures that, except with negligible probability, $\mathcal{A}$ cannot distinguish G_4 from G_3, namely,

$$|\Pr[W_4 = 1] - \Pr[W_3 = 1]| \leq \mathsf{negl}(\lambda).$$

Notice that, in this game, the NIZK proofs are as follows.

$$\widetilde{\mathsf{NIZK}}^{(0)}_{\mathsf{obl\text{-}sign}} \text{ is a simulated proof for } (\mathbf{A}, \mathbf{D}_1, m_1, m_3, \sigma_3, \mathbf{c}^{(0)}) \in \mathcal{L}(\mathcal{R}_{\mathsf{obl\text{-}sign}}),$$
$$\widetilde{\mathsf{NIZK}}^{(1)}_{\mathsf{obl\text{-}sign}} \text{ is a simulated proof for } (\mathbf{A}, \mathbf{D}_1, m_1, m_3, \sigma_3, \mathbf{c}^{(1)}) \in \mathcal{L}(\mathcal{R}_{\mathsf{obl\text{-}sign}}),$$
$$\widetilde{\mathsf{NIZK}}^{(0)}_{\mathsf{sig}} \text{ is a simulated proof for } (\mathsf{pp}, \mathbf{m}^{(0)}, \mathsf{ct}^{(0)}) \in \mathcal{L}(\mathcal{R}_{\mathsf{sig}}),$$
$$\widetilde{\mathsf{NIZK}}^{(1)}_{\mathsf{sig}} \text{ is a simulated proof for } (\mathsf{pp}, \mathbf{m}^{(1)}, \mathsf{ct}^{(1)}) \in \mathcal{L}(\mathcal{R}_{\mathsf{sig}}).$$

Game G_5: This is the same as game G_4, except that the commitment $\mathbf{c}^{(0)}$ is a commitment to $\mathbf{m}^{(1)}$, in place of $\mathbf{m}^{(0)}$, and the simulated proof $\widetilde{\mathsf{NIZK}}^{(0)}_{\mathsf{obl\text{-}sign}}$ is computed accordingly. We assume that this commitment employs a fresh randomness $\mathbf{r}'^{(1)}$. Hiding of commitments ensures that, except with negligible probability, $\mathcal{A}$ cannot distinguish G_5 from G_4, namely,

$$|\Pr[W_5 = 1] - \Pr[W_4 = 1]| \leq \mathsf{negl}(\lambda).$$

Game G_6: This is the same as game G_5, except that the commitment $\mathbf{c}^{(1)}$ is a commitment to $\mathbf{m}^{(0)}$, in place of $\mathbf{m}^{(1)}$, and the simulated proof $\widetilde{\mathsf{NIZK}}^{(1)}_{\mathsf{obl\text{-}sign}}$ is computed accordingly. We assume that this commitment employs a fresh randomness $\mathbf{r}'^{(0)}$. Hiding of commitments ensures that, except with negligible probability, $\mathcal{A}$ cannot distinguish G_6 from G_5, namely,

$$|\Pr[W_6 = 1] - \Pr[W_5 = 1]| \leq \mathsf{negl}(\lambda).$$

Game G_7: This is the same as game G_6, except that the simulated proof $\widetilde{\mathsf{NIZK}}^{(0)}_{\mathsf{obl\text{-}sign}}$ is replaced by a real NIZK proof $\mathsf{NIZK}^{(0)}_{\mathsf{obl\text{-}sign}}$ for

$$(\mathbf{A}, \mathbf{D}_1, m_1, m_3, \sigma_3, \mathbf{c}^{(0)}; \mathbf{m}^{(1)}, \mathbf{r}'^{(1)}) \in \mathcal{R}_{\mathsf{obl\text{-}sign}}$$

(c.f. (2)). Zero knowledge of NIZK proofs ensures that, except with negligible probability, $\mathcal{A}$ cannot distinguish G_7 from G_6, namely,

$$|\Pr[W_7 = 1] - \Pr[W_6 = 1]| \leq \mathsf{negl}(\lambda).$$

Notice that, in this game, the NIZK proofs are as follows.

$$\mathsf{NIZK}^{(0)}_{\mathsf{obl\text{-}sign}} \text{ is a real proof for } (\mathbf{A}, \mathbf{D}_1, m_1, m_3, \sigma_3, \mathbf{c}^{(0)}; \mathbf{m}^{(1)}, \mathbf{r}'^{(1)}) \in \mathcal{R}_{\mathsf{obl\text{-}sign}},$$
$$\widetilde{\mathsf{NIZK}}^{(1)}_{\mathsf{obl\text{-}sign}} \text{ is a simulated proof for } (\mathbf{A}, \mathbf{D}_1, m_1, m_3, \sigma_3, \mathbf{c}^{(1)}) \in \mathcal{L}(\mathcal{R}_{\mathsf{obl\text{-}sign}}),$$
$$\widetilde{\mathsf{NIZK}}^{(0)}_{\mathsf{sig}} \text{ is a simulated proof for } (\mathsf{pp}, \mathbf{m}^{(0)}, \mathsf{ct}^{(0)}) \in \mathcal{L}(\mathcal{R}_{\mathsf{sig}}),$$
$$\widetilde{\mathsf{NIZK}}^{(1)}_{\mathsf{sig}} \text{ is a simulated proof for } (\mathsf{pp}, \mathbf{m}^{(1)}, \mathsf{ct}^{(1)}) \in \mathcal{L}(\mathcal{R}_{\mathsf{sig}}).$$

Game G_8: This is the same as game G_7, except that the simulated proof $\widetilde{\mathsf{NIZK}}^{(1)}_{\mathsf{obl\text{-}sign}}$ is replaced by a real NIZK proof $\mathsf{NIZK}^{(1)}_{\mathsf{obl\text{-}sign}}$ for

$$(\mathbf{A}, \mathbf{D}_1, m_1, m_3, \sigma_3, \mathbf{c}^{(1)}; \mathbf{m}^{(0)}, \mathbf{r}'^{(0)}) \in \mathcal{R}_{\mathsf{obl\text{-}sign}}$$

(c.f. (2)). Zero knowledge of NIZK proofs ensures that, except with negligible probability, $\mathcal{A}$ cannot distinguish G_8 from G_7, namely,

$$|\Pr[W_8 = 1] - \Pr[W_7 = 1]| \leq \mathsf{negl}(\lambda).$$

Notice that, in this game, the NIZK proofs are as follows.

$$\mathsf{NIZK}^{(0)}_{\mathsf{obl\text{-}sign}} \text{ is a real proof for } (\mathbf{A}, \mathbf{D}_1, m_1, m_3, \sigma_3, \mathbf{c}^{(0)}; \mathbf{m}^{(1)}, \mathbf{r}'^{(1)}) \in \mathcal{R}_{\mathsf{obl\text{-}sign}},$$
$$\mathsf{NIZK}^{(1)}_{\mathsf{obl\text{-}sign}} \text{ is a real proof for } (\mathbf{A}, \mathbf{D}_1, m_1, m_3, \sigma_3, \mathbf{c}^{(1)}; \mathbf{m}^{(0)}, \mathbf{r}'^{(0)}) \in \mathcal{R}_{\mathsf{obl\text{-}sign}},$$
$$\widetilde{\mathsf{NIZK}}^{(0)}_{\mathsf{sig}} \text{ is a simulated proof for } (\mathsf{pp}, \mathbf{m}^{(0)}, \mathsf{ct}^{(0)}) \in \mathcal{L}(\mathcal{R}_{\mathsf{sig}}),$$
$$\widetilde{\mathsf{NIZK}}^{(1)}_{\mathsf{sig}} \text{ is a simulated proof for } (\mathsf{pp}, \mathbf{m}^{(1)}, \mathsf{ct}^{(1)}) \in \mathcal{L}(\mathcal{R}_{\mathsf{sig}}).$$

Game G_9: This is the same as game G_8, except that the simulated proof $\widetilde{\mathsf{NIZK}}_{\mathsf{sig}}^{(0)}$ is replaced by a real NIZK proof $\mathsf{NIZK}_{\mathsf{sig}}^{(0)}$ for

$$(\mathsf{pp}, \mathbf{m}^{(0)}, \mathsf{ct}^{(0)}; \Gamma^{(1)}, \mathsf{pk}_{\mathsf{id}^{(0)}}, \mathsf{sk}_{\mathsf{id}^{(0)}}, w^{(0)}, \mathsf{info}^{\tau^{(1)}}, \mathbf{r}'^{(0)}, \mathbf{r}^{(0)}, \mathsf{tag}^{(1)}, \mathbf{A}_{\mathsf{tag}^{(1)}}) \in \mathcal{R}_{\mathsf{sig}}$$

(c.f. (6)). Zero knowledge of NIZK proofs ensures that, except with negligible probability, $\mathcal{A}$ cannot distinguish G_9 from G_8, namely,

$$|\Pr[W_9 = 1] - \Pr[W_8 = 1]| \leq \mathsf{negl}(\lambda).$$

Notice that, in this game, the NIZK proofs are as follows.

$\mathsf{NIZK}_{\mathsf{obl\text{-}sign}}^{(0)}$ is a real proof for $(\mathbf{A}, \mathbf{D}_1, m_1, m_3, \sigma_3, \mathbf{c}^{(0)}; \mathbf{m}^{(1)}, \mathbf{r}'^{(1)}) \in \mathcal{R}_{\mathsf{obl\text{-}sign}}$,
$\mathsf{NIZK}_{\mathsf{obl\text{-}sign}}^{(1)}$ is a real proof for $(\mathbf{A}, \mathbf{D}_1, m_1, m_3, \sigma_3, \mathbf{c}^{(1)}; \mathbf{m}^{(0)}, \mathbf{r}'^{(0)}) \in \mathcal{R}_{\mathsf{obl\text{-}sign}}$,
$\mathsf{NIZK}_{\mathsf{sig}}^{(0)}$ is a real proof for
$(\mathsf{pp}, \mathbf{m}^{(0)}, \mathsf{ct}^{(0)}; \Gamma^{(1)}, \mathsf{pk}_{\mathsf{id}^{(0)}}, \mathsf{sk}_{\mathsf{id}^{(0)}}, w^{(0)}, \mathsf{info}^{\tau^{(1)}}, \mathbf{r}'^{(0)}, \mathbf{r}^{(0)}, \mathsf{tag}^{(1)}, \mathbf{A}_{\mathsf{tag}^{(1)}}) \in \mathcal{R}_{\mathsf{sig}}$,
$\widetilde{\mathsf{NIZK}}_{\mathsf{sig}}^{(1)}$ is a simulated proof for $(\mathsf{pp}, \mathbf{m}^{(1)}, \mathsf{ct}^{(1)}) \in \mathcal{L}(\mathcal{R}_{\mathsf{sig}})$.

Game G_{10}: This is the same as game G_9, except that the simulated proof $\widetilde{\mathsf{NIZK}}_{\mathsf{sig}}^{(1)}$ is replaced by a real NIZK proof $\mathsf{NIZK}_{\mathsf{sig}}^{(1)}$ for

$$(\mathsf{pp}, \mathbf{m}^{(1)}, \mathsf{ct}^{(1)}; \Gamma^{(0)}, \mathsf{pk}_{\mathsf{id}^{(1)}}, \mathsf{sk}_{\mathsf{id}^{(1)}}, w^{(1)}, \mathsf{info}^{\tau^{(0)}}, \mathbf{r}'^{(1)}, \mathbf{r}^{(1)}, \mathsf{tag}^{(0)}, \mathbf{A}_{\mathsf{tag}^{(0)}}) \in \mathcal{R}_{\mathsf{sig}}$$

(c.f. (6)). Zero knowledge of NIZK proofs ensures that, except with negligible probability, $\mathcal{A}$ cannot distinguish G_{10} from G_9 namely,

$$|\Pr[W_{10} = 1] - \Pr[W_9 = 1]| \leq \mathsf{negl}(\lambda).$$

Notice that, in this game, the NIZK proofs are as follows.

$\mathsf{NIZK}_{\mathsf{obl\text{-}sign}}^{(0)}$ is a real proof for $(\mathbf{A}, \mathbf{D}_1, m_1, m_3, \sigma_3, \mathbf{c}^{(0)}; \mathbf{m}^{(1)}, \mathbf{r}'^{(1)}) \in \mathcal{R}_{\mathsf{obl\text{-}sign}}$,
$\mathsf{NIZK}_{\mathsf{obl\text{-}sign}}^{(1)}$ is a real proof for $(\mathbf{A}, \mathbf{D}_1, m_1, m_3, \sigma_3, \mathbf{c}^{(1)}; \mathbf{m}^{(0)}, \mathbf{r}'^{(0)}) \in \mathcal{R}_{\mathsf{obl\text{-}sign}}$,
$\mathsf{NIZK}_{\mathsf{sig}}^{(0)}$ is a real proof for
$(\mathsf{pp}, \mathbf{m}^{(0)}, \mathsf{ct}^{(0)}; \Gamma^{(1)}, \mathsf{pk}_{\mathsf{id}^{(0)}}, \mathsf{sk}_{\mathsf{id}^{(0)}}, w^{(0)}, \mathsf{info}^{\tau^{(1)}}, \mathbf{r}'^{(0)}, \mathbf{r}^{(0)}, \mathsf{tag}^{(1)}, \mathbf{A}_{\mathsf{tag}^{(1)}}) \in \mathcal{R}_{\mathsf{sig}}$,
$\mathsf{NIZK}_{\mathsf{sig}}^{(1)}$ is a real proof for
$(\mathsf{pp}, \mathbf{m}^{(1)}, \mathsf{ct}^{(1)}; \Gamma^{(0)}, \mathsf{pk}_{\mathsf{id}^{(1)}}, \mathsf{sk}_{\mathsf{id}^{(1)}}, w^{(1)}, \mathsf{info}^{\tau^{(0)}}, \mathbf{r}'^{(1)}, \mathbf{r}^{(1)}, \mathsf{tag}^{(0)}, \mathbf{A}_{\mathsf{tag}^{(0)}}) \in \mathcal{R}_{\mathsf{sig}}$.

Notice that game G_{10} is the exactly the original experiment $\mathsf{E}_{\mathcal{A}}^{\mathsf{pre\text{-}unlk\text{-}1}}(\lambda)$ in Figure 3 for $b = 1$. Hence,

$$\Pr[W_{10} = 1] = \Pr[\mathsf{E}_{\mathcal{A}}^{\mathsf{pre\text{-}unlk\text{-}1}}(\lambda) = 1].$$

Thus,

$$\left|\Pr[\mathsf{E}_{\mathcal{A}}^{\mathsf{pre\text{-}unlk\text{-}0}}(\lambda) = 1] - \Pr[\mathsf{E}_{\mathcal{A}}^{\mathsf{pre\text{-}unlk\text{-}1}}(\lambda) = 1]\right| \leq \mathsf{negl}(\lambda).$$

We remark that if the Ajtai commitments are statistically hiding and the NIZK proofs are statistically zero-knowledge, then our lattice-based EFDGS is secure against any computationally unbounded adversary $\mathcal{A}$ in the case of message hiding and system information unlinkability.

□

Lemma 42 (Anonymity). *Our lattice-based EFDGS (c.f. Appendix B.2) is anonymous with respect to oracle* $\mathsf{OSigChal}_b$ *(c.f. Definition 7), assuming that the underlying NIZK proofs are zero-knowledge, and* $\mathcal{PKE}_{\mathsf{lp}}$ *is IND-CPA (c.f. Lemma 25).*

Proof. We assume the adversary only provides valid inputs (so that $\mathsf{OSigChal}_b$ would not return $\perp$). By following the specification of oracle $\mathsf{OSigChal}_b$ in Definition 7, for convenience purpose, we parse

$$\Sigma = (\mathsf{ct}, \mathsf{NIZK}_{\mathsf{sig}}).$$

Define the following sequence of games and denote by W_i the output of $\mathcal{A}$ in game G_i.

Game G_0: This is the original experiment $\mathsf{E}_{\mathcal{A}}^{\mathsf{anon}\text{-}0}(\lambda)$ from Figure 3 for $b = 0$. Hence,

$$\Pr[W_0 = 1] = \Pr[\mathsf{E}_{\mathcal{A}}^{\mathsf{anon}\text{-}0}(\lambda) = 1].$$

Notice that, in this game, the NIZK proofs are as follows.

$$\mathsf{NIZK}_{\mathsf{sig}} \text{ is a real proof for}$$
$$(\mathsf{pp}, \mathbf{m}, \mathsf{ct}; \Gamma^{(0)}, \mathsf{pk}_{\mathsf{id}^{(0)}}, \mathsf{sk}_{\mathsf{id}^{(0)}}, w^{(0)}, \mathsf{info}^{\tau^{(0)}}, \mathbf{r}'^{(0)}, \mathbf{r}^{(0)}, \mathsf{tag}^{(0)}, \mathbf{A}_{\mathsf{tag}^{(0)}}) \in \mathcal{R}_{\mathsf{sig}}.$$

Game G_1: This is the same as game G_0, except that the NIZK proof $\mathsf{NIZK}_{\mathsf{sig}}$ is replaced by the simulated proof $\widetilde{\mathsf{NIZK}}_{\mathsf{sig}}$ by running the respective simulator. Zero knowledge of NIZK proofs ensures that, except with negligible probability, $\mathcal{A}$ cannot distinguish G_1 from G_0, namely,

$$|\Pr[W_1 = 1] - \Pr[W_0 = 1]| \leq \mathsf{negl}(\lambda).$$

Notice that, in this game, the NIZK proofs are as follows.

$$\widetilde{\mathsf{NIZK}}_{\mathsf{sig}} \text{ is a simulated proof for } (\mathsf{pp}, \mathbf{m}, \mathsf{ct}) \in \mathcal{L}(\mathcal{R}_{\mathsf{sig}}).$$

Game G_2: This is the same as game G_1, except that the ciphertext ct encrypts $\mathsf{id}^{(1)}$, in place of $\mathsf{id}^{(0)}$, and the simulated proof $\widetilde{\mathsf{NIZK}}_{\mathsf{sig}}$ is computed accordingly. We assume that this ciphertext employs a fresh randomness $\mathbf{r}$. IND-CPA of PKE ensures that, except with negligible probability, $\mathcal{A}$ cannot distinguish G_2 from G_1, namely,

$$|\Pr[W_2 = 1] - \Pr[W_1 = 1]| \leq \mathsf{negl}(\lambda).$$

Game G_3: This is the same as game G_2, except that the simulated proof $\widetilde{\mathsf{NIZK}}_{\mathsf{sig}}$ is replaced by a real NIZK proof $\mathsf{NIZK}_{\mathsf{sig}}$ for

$$(\mathsf{pp}, \mathbf{m}, \mathsf{ct}; \Gamma^{(1)}, \mathsf{pk}_{\mathsf{id}^{(1)}}, \mathsf{sk}_{\mathsf{id}^{(1)}}, w^{(1)}, \mathsf{info}^{\tau^{(1)}}, \mathbf{r}'^{(1)}, \mathbf{r}^{(1)}, \mathsf{tag}^{(1)}, \mathbf{A}_{\mathsf{tag}^{(1)}}) \in \mathcal{R}_{\mathsf{sig}}$$

(c.f. (6)). Zero knowledge of NIZK proofs ensures that, except with negligible probability, $\mathcal{A}$ cannot distinguish G_3 from G_2, namely,

$$|\Pr[W_3 = 1] - \Pr[W_2 = 1]| \leq \mathsf{negl}(\lambda).$$

Notice that game G_3 is the exactly the original experiment $\mathsf{E}_{\mathcal{A}}^{\mathsf{anon}\text{-}1}(\lambda)$ in Figure 3 for $b = 1$. Hence,

$$\Pr[W_3 = 1] = \Pr[\mathsf{E}_{\mathcal{A}}^{\mathsf{anon}\text{-}1}(\lambda) = 1].$$

Thus,

$$\left|\Pr[\mathsf{E}_{\mathcal{A}}^{\mathsf{anon}\text{-}0}(\lambda) = 1] - \Pr[\mathsf{E}_{\mathcal{A}}^{\mathsf{anon}\text{-}1}(\lambda) = 1]\right| \leq \mathsf{negl}(\lambda)$$

which concludes the proof. □

Unforgeability. The unforgeability of our lattice-based EFDGS (c.f. Appendix B.2) follows the following Lemma 43.

Lemma 43 (Unforgeability of Lattice-Based EFDGS). *Our lattice EFDGS (c.f. Appendix B.2) is unforgeable (c.f. Definition 10) assuming the underlying NIZK proofs, i.e., $\mathsf{NIZK}_{\mathsf{obl\text{-}sign}}$ and $\mathsf{NIZK}_{\mathsf{sig}}$, are sound and mode-indistinguishable (if these NIZK proofs are dual-mode), hash function $h_{\mathbf{T}}$ is collision resistance, $\Pi_{\mathsf{obl\text{-}sign}}$ is unforgeable (c.f. Lemma 32), and $\mathcal{PKE}_{\mathsf{lp}}$ is correct (c.f. Lemma 24).*

Proof. We split the proof of this lemma into two parts, namely,

– *Extractions.* In this part, we describe how to construct the additional algorithms, in the spirit of Definition 9 for sim-ext, as well as showing how the simulated setup satisfies the extra sub-properties, including correctness (for sim-ext) and setup indistinguishability.
– *Arguing Unforgeability.* With the additional algorithms in sim-ext, we argue the unforgeability according to Definition 10.

We now proceed with the two mentioned parts.

Extractions. Recall that unforgeability of EFDGS (c.f. Section 4.2) requires the ability to construct the algorithms $\mathsf{SimSetup}$, $\mathsf{ExtPreSig}$, and ExtSig as required in Definition 9. Constructing these algorithms crucially relies on the extractors of NIZK proofs $\mathsf{NIZK}_{\mathsf{obl\text{-}sign}}$ and $\mathsf{NIZK}_{\mathsf{sig}}$ for relations $\mathcal{R}_{\mathsf{obl\text{-}sign}}$ and $\mathcal{R}_{\mathsf{sig}}$, respectively, in algorithms $\mathsf{PreSign}$ and Sign, respectively. Moreover, when setting up the system by using $\mathsf{SimSetup}$, according to the security definition of EFDGS in Section 4.2, we need to guarantee that the system satisfies two additional properties.

– Correctness with respect to the simulated setup, and
– Setup indistinguishability.

Hence, following [68], in this result, we consider the two following types of zero-knowledge proofs with extraction strategies discussed as follows.

– *Dual-Mode NIZK Proofs [42].* In this type of proof, there are two setups allowing perfect ZK in the hiding mode and witness extractability in the binding mode (whose setup provides an additional trapdoor for extracting the witness). Hence, in the real setup Setup of our lattice-based EFDGS, we use the hiding mode while, in the simulated setup $\mathsf{SimSetup}$, we use the binding mode of the respective dual-mode NIZK proof. As noted from [68], [42] only provides extraction of group elements (in pairings) and does not provide extraction of $\mathbb{Z}_q$-elements. Since our NIZK proofs, namely, $\mathsf{NIZK}_{\mathsf{obl\text{-}sign}}$ and $\mathsf{NIZK}_{\mathsf{sig}}$, only prove constraints over $\mathbb{Z}_q$, we hence follow [68] to commit the $\mathbb{Z}_q$-elements by committing to their binary representation instead. By mapping the bits into the group elements, i.e., 0 and 1 are respectively mapped to the identity and the generator of the respective group (in pairings), we can extract the bits by simply extracting the group elements. Hence, the extracted $\mathbb{Z}_q$-element is 0 (respectively, 1) if the extracted group element is the identity (respectively, the generator). We can recover the $\mathbb{Z}_q$-elements from the extracted bits as desired.
– *NIZKAoKs from (Multi-Round) Interactive Protocols.* These proofs are usually constructed by transforming a multi-round interactive protocol (3-round ones are notoriously called Σ-protocols) into a respective non-interactive version by using the Fiat-Shamir transform [37]. These non-interactive protocols are non-interactive zero-knowledge arguments of knowledge (NIZKAoK, c.f. Appendix A.8 for a preliminary of NIZKAoKs) if the respective interactive protocols are ZKAoKs. However, there are various results for this type of NIZK proofs for arithmetic circuits, including [40,85,6,63,10] just to name a few. Extracting witnesses from these NIZK proofs usually relies on the forking lemma [77], and the extraction strategies depend on the designs of the proofs. Here, we omit the details of how to extract from the proofs. Besides, we refer to the following theoretical results [7,8] regarding extracting non-interactive NIZK proofs from the Fiat-Shamir transform. Hence, when using this type of NIZK proof, the simulated setup is designed to allow the extractor in the corresponding system to run and extract the witnesses.

We now describe the additional algorithms, with respect to the above discussion, as follows.

$\mathsf{SimSetup}(1^\lambda)$**:** If the employed NIZK proofs are dual-mode NIZK proofs, return the trapdoor in the binding mode, allowing to extract witnesses. Otherwise, if they are from (multi-round) interactive proofs, we use the setting allowing an extractor to extract the witnesses described above. Here, the extraction trapdoor can be assumed to be empty.

$\mathsf{ExtPreSig}(\mathsf{pp}, \mathsf{td}_{\mathsf{ext}}, \mathsf{info}^\tau, \Gamma, \mathsf{trans}) \to (\widetilde{\mathbf{m}}, \widetilde{\mathsf{aux}})$**:** Recall that $\mathsf{PreSign}\ \langle \mathsf{U}(\mathbf{m}), \mathsf{SA}(\mathsf{sk}_{\mathsf{sa}})\rangle\ (\mathsf{pp}, \mathsf{info}^\tau)$ (c.f. Appendix B.2) was invoked at epoch τ by running steps 1 and 2 of protocol $\Pi_{\mathsf{obl\text{-}sign}}^{\mathsf{U}(\mathbf{m}),\mathsf{SA}(\mathbf{R},F,\mathsf{ctr})}(\mathsf{pp}_{\mathsf{obl}}, \mathsf{info}^\tau)$ to obtain the transcript $\mathsf{trans} = (\mathbf{c}, \mathsf{NIZK}_{\mathsf{obl\text{-}sign}})$ and $\Gamma = (\mathsf{tag}, \mathbf{v}')$ where $\mathsf{NIZK}_{\mathsf{obl\text{-}sign}}$ is a NIZK proof for relation $\mathcal{R}_{\mathsf{obl\text{-}sign}}$ (c.f. (2)). By using extraction trapdoor $\mathsf{td}_{\mathsf{ext}}$, we can extract $(\widetilde{\mathbf{m}}, \widetilde{\mathbf{r}}')$ from $\mathsf{NIZK}_{\mathsf{obl\text{-}sign}}$ satisfying $\mathcal{R}_{\mathsf{obl\text{-}sign}}$. Hence, as from Appendix A.7, we can obtain a valid signature for $(\widetilde{\mathbf{m}} \| \mathsf{info}^\tau)$, hence guaranteeing that $\mathsf{PreVerify}(\mathsf{pp}, \mathsf{info}^\tau, \widetilde{\mathbf{m}}, \widetilde{\mathsf{aux}}, \Gamma) = 1$. This algorithm then returns $(\widetilde{\mathbf{m}}, \widetilde{\mathsf{aux}}) = (\widetilde{\mathbf{m}}, \widetilde{\mathbf{r}}')$.

$\mathsf{ExtSig}(\mathsf{pp}, \mathsf{td}_{\mathsf{ext}}, \mathbf{m}, \Sigma) \to (\widetilde{\mathsf{id}}, \widetilde{\mathsf{info}^\tau})$**:** Parse $\Sigma = (\mathsf{ct}, \mathsf{NIZK}_{\mathsf{sig}})$. Since we use a NIZK proof $\mathsf{NIZK}_{\mathsf{sig}}$ for $\mathcal{R}_{\mathsf{sig}}$, we can extract

$$(\widetilde{\Gamma}, \widetilde{\mathsf{pk}_{\mathsf{id}}}, \widetilde{\mathsf{sk}_{\mathsf{id}}}, \widetilde{w}, \widetilde{\mathsf{info}^\tau}, \widetilde{\mathbf{r}}', \widetilde{\mathbf{r}}, \widetilde{\mathsf{tag}}, \widetilde{\mathbf{A}_{\mathsf{tag}}})$$

such that $(\mathsf{pp}, \mathbf{m}, \mathsf{ct}; \widetilde{\Gamma}, \widetilde{\mathsf{pk}_{\mathsf{id}}}, \widetilde{\mathsf{sk}_{\mathsf{id}}}, \widetilde{w}, \widetilde{\mathsf{info}}^{\tau}, \widetilde{\mathbf{r}}', \widetilde{\mathbf{r}}, \widetilde{\mathsf{tag}}, \widetilde{\mathbf{A}}_{\widetilde{\mathsf{tag}}}) \in \mathcal{R}_{\mathsf{sig}}$. Parse $\widetilde{w} = (\widetilde{\mathsf{id}}, (\widetilde{\mathbf{w}}_1, \ldots, \widetilde{\mathbf{w}}_L))$. Then, return $(\widetilde{\mathsf{id}}, \widetilde{\mathsf{info}}^{\tau})$.

We now consider the additional properties, namely, correctness and setup indistinguishability, when instantiated with respect to the above types of proofs. Regarding setup indistinguishability, both setups are indistinguishable. Specifically, for dual-mode NIZK proofs, the binding and hiding modes are computationally indistinguishable, which implies the indistinguishability of both setups. For NIZKAoKs from (multi-round) interactive protocols, the setup indistinguishability trivially holds. Regarding correctness in the simulated setup, the proof is straightforward.

Arguing Unforgeability. We now prove the required sub-properties of unforgeability (see discussion in Section 4.2). Regarding unforgeability with respect to experiment $\mathsf{E}_{\mathcal{A}}^{\mathsf{unf\text{-}wo\text{-}tracesnd}}$, assume that PPT adversary $\mathcal{A}$ is able to produce $(\mathbf{m}, \Sigma)$ such that $(\mathbf{m}, \Sigma) \notin \mathsf{SIGS}$ and $\mathsf{Verify}(\mathsf{pp}, \mathbf{m}, \Sigma) = 1$, as required to win the unforgeability experiment $\mathsf{E}_{\mathcal{A}}^{\mathsf{unf\text{-}wo\text{-}tracesnd}}$ (c.f. Figure 4). We prove extractability, system information unforgeability, traceability, and non-frameability as follows.

- *Extractability.* In this sub-property, $\mathcal{A}$ is considered to win $\mathsf{E}_{\mathcal{A}}^{\mathsf{unf\text{-}wo\text{-}tracesnd}}(\lambda)$ if $\mathsf{Judge}(\mathsf{pp}, \mathbf{m}, \Sigma, \mathsf{id}', \Xi) = 1$, the traced identifier id' belongs to $\mathcal{ID}$ and the extracted identifier $\widetilde{\mathsf{id}}$ is not equal to id'. As ExtSig extracts $\widetilde{\mathsf{id}}$ and $\widetilde{\mathbf{r}}$ from $\mathsf{NIZK}_{\mathsf{sig}}$ for relation $\mathcal{R}_{\mathsf{sig}}$ (c.f. (6)), we should note that, by soundness of $\mathsf{NIZK}_{\mathsf{sig}}$, $\mathsf{ct} = \mathsf{PKE.Enc}(\mathsf{pk}_{\mathsf{ta}}, \widetilde{\mathsf{id}}; \widetilde{\mathbf{r}})$, i.e., ct is the correct encryption of $\widetilde{\mathsf{id}}$. On the other hand, since $\mathsf{Judge}(\mathsf{pp}, \mathbf{m}, \Sigma, \mathsf{id}', \Xi) = 1$, we understand that id' is the correct decryption of ct, i.e., $\mathsf{id}' \leftarrow \mathsf{PKE.Dec}(\mathsf{sk}_{\mathsf{ta}}, \mathsf{ct})$ due to the soundness of $\mathsf{NIZK}_{\mathsf{trace}}$, for relation $\mathcal{R}_{\mathsf{trace}}$ (c.f. (10)). Since $\mathsf{id}' \neq \widetilde{\mathsf{id}}$, this implies a decryption error that violates the correctness of the respective PKE $\mathcal{PKE}_{\mathsf{lp}}$. Hence, the extractability of our lattice-based EFDGS is guaranteed by the soundness of $\mathsf{NIZK}_{\mathsf{sig}}$ and correctness of $\mathcal{PKE}_{\mathsf{lp}}$.
- *System Information Unforgeability.* In this sub-property, $\mathcal{A}$ is considered to win $\mathsf{E}_{\mathcal{A}}^{\mathsf{unf\text{-}wo\text{-}tracesnd}}(\lambda)$ if the extracted pair $(\mathbf{m}, \widetilde{\mathsf{info}}^{\tau}) \notin \mathsf{EPL}$, where $\widetilde{\mathsf{info}}^{\tau}$ is extracted via ExtSig, and $\mathcal{A}$ has never accessed to $\mathsf{sk}_{\mathsf{sa}}$. Notice that the extraction of $\widetilde{\mathsf{info}}^{\tau}$ is proceeded by extracting from $\mathsf{NIZK}_{\mathsf{sig}}$ for relation $\mathcal{R}_{\mathsf{sig}}$ (c.f. (6)). In $\mathcal{R}_{\mathsf{sig}}$, the set of constraints (i) implies a valid original signature, namely, $(\mathsf{tag}, \mathbf{v}' - (\mathbf{r}' \| \mathbf{0}^{m_2}))$, which is verifiable by $\mathsf{pk}_{\mathsf{sa}}$ and message $(\mathbf{m} \| \widetilde{\mathsf{info}}^{\tau})$ (see Remark 29). Since $\mathcal{A}$ has no access to $\mathsf{sk}_{\mathsf{sa}}$ and can create a valid signature for a new message $(\mathbf{m} \| \widetilde{\mathsf{info}}^{\tau})$, which has not been requested previously via either $\mathsf{PreSign}$ or $\mathsf{PreSign}_{\mathsf{aug}}$, i.e., $(\mathbf{m}, \widetilde{\mathsf{info}}^{\tau}) \notin \mathsf{EPL}$, this implies that $\mathcal{A}$ can forge a pre-signature verifiable by $\mathsf{pk}_{\mathsf{sa}}$. By Lemma 32, this only happens with negligible probability.
- *Traceability.* In this sub-property, $\mathcal{A}$ is considered to win $\mathsf{E}_{\mathcal{A}}^{\mathsf{unf\text{-}wo\text{-}tracesnd}}(\lambda)$ if the traced identifier id' is not in the corrupt list CL and $\mathcal{A}$ has never accessed to $\mathsf{sk}_{\mathsf{sa}}$. Since $\mathsf{Verify}(\mathsf{pp}, \mathbf{m}, \Sigma) = 1$, we know that the underlying NIZK proof $\mathsf{NIZK}_{\mathsf{sig}}$, for relation $\mathcal{R}_{\mathsf{sig}}$ (c.f. (6)), is verified successfully. However, by the set of constraints (ii) in relation $\mathcal{R}_{\mathsf{sig}}$ and by extractability proved above, it implies that id' is a valid identity at the respective extracted system information $\widetilde{\mathsf{info}}^{\tau}$ (via ExtSig). Hence, there are three possibilities for $\mathcal{A}$ as follows.
  (i) Identifier id' is in the system at the system information $\widetilde{\mathsf{info}}^{\tau}$ and $\mathcal{A}$ has a valid secret mapping to $\mathsf{pk}_{\mathsf{id}'}$, implying that $\mathcal{A}$ is able to invert $h_{\mathbf{T}}$ to some valid pre-image. In this case, with probability at least 1/2 [62, Lemma 8], the valid pre-image that $\mathcal{A}$ founds is different from the actual $\mathsf{sk}_{\mathsf{id}'}$, hence breaking the collision resistance of the hash function $h_{\mathbf{T}}$.
  (ii) $\mathcal{A}$ does not have a valid path of accumulating following the Merkle tree structure underlying $\mathcal{ACC}_{\mathsf{llnw}}$. In this case, $\mathcal{A}$ is considered to successfully break the collision resistance of $h_{\mathbf{T}}$.
  (iii) $\mathcal{A}$ is able to break the soundness of $\mathsf{NIZK}_{\mathsf{sig}}$.
  Hence, breaking traceability only happens with negligible probability due to the collision resistance of $h_{\mathbf{T}}$ and the soundness of $\mathsf{NIZK}_{\mathsf{sig}}$.
- *Non-Frameability.* With a similar argument as for the case traceability, framing EFDGS signatures to an honest user whose traced identifier id' in the set HL is impossible because $\mathcal{A}$ does not possess $\mathsf{sk}_{\mathsf{id}'}$. Hence, breaking this property implies breaking either the collision resistance of $h_{\mathbf{T}}$ or the soundness of $\mathsf{NIZK}_{\mathsf{sig}}$, which only happens with negligible probability.

Regarding the case of tracing soundness with respect to Figure 4, by the correctness of $\mathcal{PKE}_{\mathsf{lp}}$ and the soundness of $\mathsf{NIZK}_{\mathsf{sig}}$ which captures correct encryption of identifier, $\mathcal{A}$ has negligibly advantage to provide the distinct identifiers with associated proofs of tracing.

We thus conclude the proof. □