Group Encryption: Full Dynamicity, Message Filtering and Code-Based Instantiation

Khoa Nguyen[1], Reihaneh Safavi-Naini[2], Willy Susilo[3], Huaxiong Wang[1], Yanhong Xu[2], and Neng Zeng[4]

[1] School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore

[2] Department of Computer Science, University of Calgary, Calgary, Canada

[3] Institute of Cybersecurity and Cryptology, School of Computing and Information Technology, University of Wollongong, Wollongong NSW, Australia

[4] Information Systems Technology and Design, Singapore University of Technology and Design, Singapore

Abstract. Group encryption (GE), introduced by Kiayias, Tsiounis and Yung (Asiacrypt'07), is the encryption analogue of group signatures. It allows to send verifiably encrypted messages satisfying certain requirements to certified members of a group, while keeping the anonymity of the receivers. Similar to the tracing mechanism in group signatures, the receiver of any ciphertext can be identified by an opening authority - should the needs arise. The primitive of GE is motivated by a number of interesting privacy-preserving applications, including the filtering of encrypted emails sent to certified members of an organization.

This paper aims to improve the state-of-affairs of GE systems. Our first contribution is the formalization of fully dynamic group encryption (FDGE) - a GE system simultaneously supporting dynamic user enrolments and user revocations. The latter functionality for GE has not been considered so far. As a second contribution, we realize the message filtering feature for GE based on a list of t-bit keywords and 2 commonly used policies: "permissive" - accept the message if it contains at least one of the keywords as a substring; "prohibitive" - accept the message if all of its t-bit substrings are at Hamming distance at least d from all keywords, for $d \geq 1$. This feature so far has not been substantially addressed in existing instantiations of GE based on DCR, DDH, pairing-based and lattice-based assumptions. Our third contribution is the first instantiation of GE under code-based assumptions. The scheme is more efficient than the lattice-based construction of Libert et al. (Asiacrypt'16) - which, prior to our work, is the only known instantiation of GE under post-quantum assumptions. Our scheme supports the 2 suggested policies for message filtering, and in the random oracle model, it satisfies the stringent security notions for FDGE that we put forward.

Keywords. group encryption, full dynamicity, revocation, anonymity, zero-knowledge, spam filtering, code-based protocols, string matching

1 Introduction

The study of group encryption - the encryption analogue of group signatures [22] - was initiated by Kiayias, Tsiounis and Yung (KTY) [34] in 2007. While group signatures allow the signers to hide their identities within a set of certified senders, group encryption protects the anonymity of the decryptors within a set of legitimate receivers. To keep users accountable for their actions, signatures/ciphertexts can be de-anonymized in cases of disputes, using a secret key possessed by an opening authority.

In a group encryption scheme, the sender of a ciphertext can generate publicly verifiable proofs that: (i) The ciphertext is well-formed and can be decrypted by some registered group member; (ii) The opening authority can identify the intended receiver should the needs arise; (iii) The plaintext satisfies certain requirements, such as being a witness for some public relation.

Group encryption (GE) schemes are motivated by a number of appealing privacy-preserving applications. A natural application is for encrypted email filtering, where GE allows a firewall to accept only those incoming emails that are intended for some certified organization user. If accepted, the encrypted messages are guaranteed to satisfy some prescribed requirements, such as the absence of spammy/unethical keywords or the presence of keywords that are of the organization's interests.

As pointed out in [34] and subsequent work [20,1,41,37], GE can also find interesting applications in the contexts of anonymous trusted third parties, oblivious retriever storage systems or asynchronous transfers of encrypted datasets. For instance, it allows to archive on remote servers encrypted datasets intended for some anonymous client who paid a subscription to the storage provider. Furthermore, the recipient can be identified by a judge if a misbehaving server is found guilty of hosting suspicious transaction records or any other illegal content.

From the theoretical point of view, one can build a secure GE scheme based on anonymous CCA2-secure public key encryption schemes, digital signatures, commitments and zero-knowledge proofs. The designs of GE are typically more sophisticated than for group signatures, due to the need of proving well-formedness of ciphertexts encrypted via hidden-but-certified users' public keys. In particular, as noted by Kiayias et al. [34], GE implies hierarchical group signatures [59] - a proper generalization of group signatures [7,8].

In their pioneering work, Kiayias et al. instantiated GE based on the Decisional Composite Residuosity (DCR) and the Decisional Diffie Hellman (DDH) assumptions. The zero-knowledge proof of ciphertext well-formedness in their scheme is interactive, but can be made non-interactive in the random oracle model using the Fiat-Shamir transformation [26]. Cathalo et al. [20] subsequently proposed a non-interactive realization based on pairings in the standard model. El Aimani and Joye [1] then suggested various efficiency improvements for pairing-based GE. The first construction of GE from lattice assumptions was later presented by Libert et al. [37].

Libert et al. [41] enriched the KTY model of GE by introducing a refined tracing mechanism inspired by that of traceable signatures [33]. In this setting,

the opening authority can release a user-specific trapdoor that enables public tracing of ciphertexts sent to that specific users without violating other users' privacy. Izabachène et al. [31] suggested mediated traceable anonymous encryption - a related primitive that addresses the problem of eliminating subliminal channels.

Current Limitations of GE. To date, GE has been much less well-studied than group signatures [22], even though they are functionally dual to each other. The group signature primitive has a longer history of development, and serves as a primary case study for privacy-preserving authentication systems. Meanwhile, GE was introduced close to the rises of powerful encryption systems such as attribute-based [30], functional [12] and fully-homomorphic [27] encryption, and has not gained much traction. Nevertheless, given its compelling features and the nice applications it can potentially offer, we argue that GE deserves more attention from the community. In this work, we thus aim to contribute to the development of GE. To start with, we identify several limitations of existing GE systems.

First, the problem of user revocation, which is a prominent issue in multi-user cryptosystems, has not been formally addressed. The KTY model [34], while allowing dynamic enrolments of new users to the group, does not provide any mechanism to prevent revoked users (e.g., those who were expelled for misbehaviours, stopped subscribing to the services or retired from the organizations) from decrypting new ciphertexts intended for them (unless the whole system is re-initiated). We next observe that, although it was not discussed by authors of [41], their refined tracing method might pave the way for a mechanism akin to verifier-local revocation [13], in which verifiers test incoming ciphertexts using the trapdoors corresponding to *all* revoked users. Beside incurring complexity linear in the number of revoked users, such a mechanism is known to only provide a weak notion of anonymity (called selfless-anonymity) for non-revoked users. A formal treatment of fully dynamic GE (i.e., which supports both dynamic enrolments and revocations of users) with strong security requirements is therefore highly desirable.

The second limitation is about the usefulness of existing GE schemes in the context of email filtering - which is arguably the most natural application of the primitive. Recall that such filtering functionality is supposed to be done by defining a relation $R = \{(x, w)\}$ and accepting only messages w such that $(x, w) \in R$, for a publicly given x. However, in all known instantiations of GE, the relations for messages are defined according to the computationally hard problems used in other system components. More precisely, the KTY scheme [34] employs the discrete-log relation, i.e., it only accepts w if $g^w = h$ for given (g, h). Subsequent works follow this pattern: pairing-based relations are used in [20,1,41] and a Short-Integer-Solution relation is used in [37] for message filtering. While such treatment does comply the definitions of GE, it seems too limited to be useful for filtering spams. Designing GE schemes with expressive policies that capture real-life spam filtering methods is hence an interesting open question.

Third, regarding the diversity of concrete computational assumptions used in building GE, among all existing schemes, the only one that is not known to be vulnerable against quantum computers [57] is the lattice-based construction from [37]. This raises the question of realizing GE based on alternative quantum-resistant assumptions, e.g., those from codes, multivariates and isogenies. In terms of privacy-preserving cryptographic protocols, other post-quantum candidates are much less developed compared to lattice-based constructions, and it would be tempting to catch up in the scope of GE.

Our Contributions and Techniques. This work addresses all the 3 limitations of existing GE that we discussed above. Our first contribution is a formal model for fully dynamic group encryption (FDGE), with carefully defined syntax and robust security notions. Our model empowers the KTY model with the user revocation functionality and paves the way for new instantiations of GE in which enrolling new users and revoking existing users can be done simultaneously and efficiently. As a second contribution, we suggest to realize message filtering for GE not based on computationally hard problems, but a list of keywords and how these keywords match with substrings of the encrypted messages. To this end, we define 2 policies for accepting "good" messages and rejecting "bad" ones, that capture the spirit of the String Matching problem and the Approximate String Matching problem that are widely used in contemporary spam filtering techniques. Our third contribution is the first code-based GE scheme that follows our FDGE model and that supports both of the 2 message filtering policies we propose. We provide more technical details in the following.

Group Encryption with Full Dynamicity. We formalize the primitive of FDGE as the encryption analogue of fully dynamic group signatures [14]. Beyond the usual joining algorithm of the KTY model [34], FDGE makes it possible to update the group periodically to reflect user revocations. Our model is defined in a way such that it captures the 2 most commonly used approaches for handling user revocations in group signatures, based on revocation lists [17] and accumulators [19]. As noted in [14], there is an attack inherently to group signature schemes following the revocation-list-based approach. When translated into the GE context, such attack would permit group users to decrypt ciphertexts sent to them even before they join the group. Our FDGE model does not allow such attack, and we view this as a preventative measure in case a revocation-list-based revocable GE will be proposed in the future.

Regarding security requirements, we define the notions of message secrecy, anonymity, and soundness that are inline with and carefully extended from the KTY model [34]. We consider adversaries with strong capabilities, including the ability to corrupt the group manager (GM) and/or the opening authority (OA) to a large extent. Specifically, not only do we permit full corruption [5] of the GM and/or OA when defining message secrecy and anonymity, but we also tolerate maliciously generated keys for the fully corrupted authorities. In terms of

[5] Full corruption means that the adversary entirely controls the authority - who may no longer follow its program.

soundness, only partial corruption [6] of OA is allowed. Note that the assumption on partially corrupted OA is minimal, since otherwise a fully corrupted OA could simply refuse to open ciphertexts.

Message Filtering. Spamming and spam filtering are complicated areas, and currently there is no single filtering solution that can address all the clever tricks of spammers. In the present work, we do not attempt to invent such a solution. Our goal is to equip GE schemes with some basic, yet commonly used policies for filtering. More precisely, we suggest to employ a public list $S = \{\mathbf{s}_1, \ldots, \mathbf{s}_k\}$ of k binary keywords, each of which has bit-length t, to test against length t substrings of the encrypted message $\mathbf{w} \in \{0,1\}^p$. This list can be regularly updated by the GM, depending on the interests and needs of the organization. The keywords $\mathbf{s}_i$ could either be "good" ones that all legitimate messages are expected to contain, or be "bad" ones that should be far - in terms of Hamming distance - from all substrings of $\mathbf{w}$. Respectively, we consider the following 2 policies.

1. **"Permissive":** $\mathbf{w}$ is a legitimate message if and only if there exists $i \in [1, k]$ such that $\mathbf{s}_i$ is a substring of $\mathbf{w}$. This policy captures the String Matching problem and can be applied when the current interests of the group are reflected by the keywords $\mathbf{s}_i$'s, and all messages that do not contain any of these keywords are rejected.
2. **"Prohibitive":** $\mathbf{w}$ is a legitimate message if and only if for every length-t substring $\mathbf{y}$ of $\mathbf{w}$ and every $\mathbf{s}_i \in S$, their Hamming distance is at least d. This policy is related to the Approximate String Matching problem. Here, the keywords $\mathbf{s}_i$'s could correspond to topics that are unethical, illegal, adultery, or simply out of the group's interests. The requirement on minimum Hamming distance d is to address spammers who might slightly alter $\mathbf{s}_i$ so that it passes the filtering while still being somewhat readable.

Having defined the policies, our next step is to derive methods for proving in zero-knowledge that the secret message $\mathbf{w}$ satisfies each of the policies, which will be used by the message sender when proving the well-formedness of the ciphertext. Let us discuss the high-level ideas.

Regarding the permissive policy, our observation is that if we form matrix $\mathbf{W} \in \{0,1\}^{t \times (p-t+1)}$ whose columns are length-t substrings of $\mathbf{w}$, and matrix $\mathbf{S} \in \{0,1\}^{t \times k}$ whose columns are the keywords $\mathbf{s}_i$, then $\mathbf{w}$ is legitimate if and only if there exist weight-1 vectors $\mathbf{g} \in \{0,1\}^{p-t+1}$ and $\mathbf{h} \in \{0,1\}^k$ such that $\mathbf{W} \cdot \mathbf{g} = \mathbf{S} \cdot \mathbf{h}$. Then, to handle this relation, we employ Stern's permuting technique [58] to prove knowledge of such $\mathbf{g}, \mathbf{h}$ and we adapt Libert et al.'s technique [37] for proving the well-formedness of the quadratic term $\mathbf{W} \cdot \mathbf{g}$.

As for the prohibitive policy, we consider all the $(p-t+1) \cdot k$ sums $\mathbf{z}_{i,j} \in \{0,1\}^t$ over $\mathbb{Z}_2$ of substrings of $\mathbf{w}$ and keywords in S. Then, $\mathbf{w}$ is legitimate if and only if all these sums have Hamming weight at least d. To prove these statements, we perform the following extension trick, inspired by [42].

[6] Partial corruption means that the adversary only knows the secret key of the authority who still follows its prescribed program.

We append $(t-d)$ coordinates to $\mathbf{z}_{i,j} \in \{0,1\}^t$ to get $\mathbf{z}^*_{i,j} \in \{0,1\}^{2t-d}$ with Hamming weight exactly t. Such an extension is always possible if $\mathbf{z}_{i,j}$ has weight at least d. Furthermore, the converse also holds: if $\mathbf{z}^*_{i,j}$ has weight t, then the original $\mathbf{z}_{i,j}$ must have weight at least $t-(t-d)=d$. At this point, it suffices to use Stern's permuting technique [58] for proving knowledge of fixed-weight binary vectors.

The techniques sketched above can be smoothly integrated into our code-based instantiation of FDGE.

Code-based instantiation. To design a scheme satisfying our model of FDGE, we would need: (1) An anonymous CCA2-secure public-key encryption to encrypt messages under a group user's public key and to encrypt the user's public key under the OA's public key; (2) A secure digital signature to certify public keys of group members; and (3) Zero-knowledge proofs compatible with the encryption and signature layers, as well as with the message filtering layer.

In the code-based setting, the first ingredient can be obtained from the randomized McEliece encryption scheme [54] that satisfies CPA-security and the Naor-Yung transformation [52]. The second ingredient seems not readily available, as code-based signatures for which there are efficient zero-knowledge proofs of knowledge of message/signature pairs are not known to date. To tackle this issue, we adapt the strategy of Ling et al. in their construction of lattice-based fully dynamic group signatures [44]. This amounts to replacing the signature scheme by an accumulator scheme [9] equipped with zero-knowledge arguments of membership. We hence can make use of the code-based realization of Merkle-tree acummulators recently proposed by Nguyen et al. [53].

The main idea is to use Merkle-tree accumulators to certify users' public key. Let $N=2^\ell$ be the maximum expected number of group users. Let $\mathsf{pk}=(\mathbf{G}_0,\mathbf{G}_1)$ be a user public key, where $\mathbf{G}_0,\mathbf{G}_1$ are 2 McEliece encryption matrices (recall that we employ the Naor-Yung double encryption technique). Then pk is hashed to a vector $\mathbf{d}\neq\mathbf{0}$, which is placed at the tree leaf corresponding to the identity $j\in\{0,1\}^\ell$ of the user in the group. A tree root is then computed based on all the 2^ℓ leaves. The user's certificate, which is made available to message senders, consists of pk, j and hash values in the path from her leaf to the root.

When sending a message $\mathbf{w}$ satisfying "permissive" or "prohibitive" policy to user j, the sender uses pk to encrypt $\mathbf{w}$ as $\mathbf{c}_w$, and uses the OA's public key to encrypt j as $\mathbf{c}_{\mathsf{oa}}$, so that OA can recover j if necessary. As for well-formedness of ciphertext, sender proves in zero-knowledge that:

1. $\mathbf{w}$ satisfies the given policy. This can be done using the discussed techniques.
2. $\mathbf{c}_{\mathsf{oa}}$ is an honestly computed ciphertext of j. This part is quite straightforward to realize via techniques for Stern's protocol.
3. $\mathbf{c}_w$ is a correct ciphertext of the $\mathbf{w}$ from (1.), computed under some hidden public key pk, whose hash value $\mathbf{d}\neq\mathbf{0}$ is at the tree leaf corresponding to the j from (2.). This is indeed the most sophisticated portion of our scheme. It requires to demonstrate: (i) membership of $\mathbf{d}$ in the tree and $\mathbf{d}\neq\mathbf{0}$ is the

hash of value of pk; (ii) $\mathbf{c}_w$ has the form $\mathbf{c}_w = \mathsf{pk} \cdot \begin{bmatrix} \mathbf{r} \\ \mathbf{w} \end{bmatrix} + \mathbf{e}$, where $(\mathbf{r}, \mathbf{e})$ is the encryption randomness.

While statement (i) can be handled using the techniques from [53,42], (ii) would require to prove an Learning-Parity-with-Noise-like relation with hidden-but-certified matrix pk. We then tackle this problem by adapting the techniques for Learning-with-Errors relations [55] from [37] into the binary setting.

Having discussed the main technical ingredients of the scheme, let us now explain how user revocations and dynamic user enrolments can be done in a simple manner based on Merkle trees. The ideas, first suggested in [44], are as follows. At the setup phase, all leaves in the tree are set as $\mathbf{0}$. When a new user joins the group, as mentioned, $\mathbf{0}$ is changed to $\mathbf{d} \neq \mathbf{0}$. If the user is later revoked from the group, the value is set back to $\mathbf{0}$. For each change, the GM can efficiently update the tree by re-computing the path in time $\mathcal{O}(\log N)$. Note that in the zero-knowledge layer above, the sender in part proves that $\mathbf{d}$ is non-zero - which is interpreted as "the sender is indeed an active group user".

Putting everything together, we obtain the first construction of code-based (fully dynamic) GE. In the random oracle model, we prove that the scheme satisfies all the stringent security notions of FDGE, namely, message secrecy, anonymity and soundness, based on the security of the code-based technical ingredients we employ.

The scheme, however, should only be viewed as a proof-of-concept, as it is not practical - due to the involvement of heavy zero-knowledge arguments. However, in comparison with [37] the only known GE scheme from post-quantum assumptions, ours is more efficient. The main reason is that ours uses a Merkle tree - which can be viewed as a weak form of signatures, while theirs relies on a standard-model lattice-based signature scheme, whose supported zero-knowledge arguments incurred an overhead factor of $\log^2 q$, where $q > 2^{30}$. We estimate that, for 128 bits of security, our argument size is about 2 orders of magnitude smaller than theirs. In other words, our scheme is more efficient than [37], but is still not practical. We leave the problem of obtaining practically usable FDGE schemes from post-quantum assumptions as an interesting open question.

Other related Work. Enabling efficient user revocations in advanced privacy-preserving cryptographic constructions is generally a challenging problem, since one has to ensure that revoked users are no longer able to act as active users, and the workloads of other parties (managers, non-revoked users, verifiers) do not significantly increase in the meantime. In the context of group signatures, several different approaches have been suggested [17,19,13] to address this problem, and efficient pairing-based constructions supporting both dynamic joining and efficient revocation were given in [51,40,39]. Bootle et al. [14] pointed out a few shortcomings of previous models, and put forward robust security notions for fully dynamic group signatures. Here, we adapt the [14] model to provide the first formal treatment of user revocation in the context of GE.

The major tools for building those privacy-preserving constructions are zero-knowledge (ZK) proof [29] and argument [28,16] systems that allow to prove the

truth of a statement while revealing no additional information. Almost all known zero-knowledge proof/argument systems used in code-based cryptography follow Stern's framework [58]. Variants of Stern's protocol have been employed to design privacy-preserving constructions, such as proofs of plaintext knowledge [50], linear-size ring signatures [48,23,49,15], linear-size and sublinear-size group signatures [25,2], proofs of valid openings for commitments and proofs for general relations [32]. Recently, Nguyen et al. [53] proposed a number of new code-based privacy-preserving protocols, including accumulators, range proofs, logarithmic-size ring signatures and group signatures. However, prior to our work, no construction of code-based GE was known.

Organization. The rest of the paper is organized as follows. Section 2 introduces the model and security requirements of FDGE. In Section 3, we recall some necessary background on code-based cryptography and techniques for Stern-like zero-knowledge protocols. In Section 4, we present our ZK argument for a quadratic relations. This is crucial for proving the permissive relation in Section 5 - where we also present the strategies for proving the prohibitive relation. Next, we describe our code-based instantiation of FDGE in Section 6. Then 2 zero-knowledge argument systems supporting the scheme are presented in Section 7. Finally, we provide security proofs for the scheme in Section 8.

2 Fully Dynamic Group Encryption: Model and Security Requirements

In this section, we first present the model of fully dynamic group encryption FDGE that offers both dynamic join and revocation, which is developed from the one proposed by Kiayias et al. [34]. Our model is analogous to the fully dynamic group signature one proposed by Bootle et al. [14]. In a FDGE scheme, the parties involved are the sender, the verifier, the group manager GM who manages the group of receivers, and the opening authority OA who is capable of identifying the recipients of ciphertexts. $\mathcal{R}$ is a public relation for which a FDGE should be verifiable. Receivers can join and leave the group at the choice of the GM. We assume that the GM will publish group information info_τ at the beginning of each time epoch τ. The information depicts changes to the group such as the existing group members or the revoked members at current epoch τ. It is required that anyone can verify the authenticity and well-formedness of the group information. In addition, by comparing the current group information with the previous one, it is possible to recover the list of members revoked from the group at the current epoch. We also assume that the epoch maintains the order in which the group information was published, i.e., info_{τ_1} precedes info_{τ_2} if $\tau_1 < \tau_2$.

Compared to [34], our model enables the GM to remove some users from the group through a group updating algorithm GUpdate. Another difference is that we avoid interaction by employing a non-interactive zero-knowledge (NIZK) proof, which has already been considered by Cathalo, Libert and Yung [20]. As highlighted by the authors, non-interaction is highly desirable as the sender, who

might be required to repeat the proof with many verifiers, needs to maintain a state and remember all the random coins used to generate the ciphertext.

2.1 Syntax

Formally, a FDGE that is verifiable for a public relation $\mathcal{R}$ consists of the following polynomial-time algorithms.

$\mathsf{Setup}_{\mathsf{init}}(1^\lambda)$ The algorithm takes as input the security parameter 1^λ and outputs a set of public parameters pp.

$\mathsf{Setup}_{\mathsf{OA}}(\mathsf{pp})$ This algorithm is run by the opening authority OA. It takes as input pp and outputs a key pair $(\mathsf{pk}_{\mathsf{OA}}, \mathsf{sk}_{\mathsf{OA}})$.

$\mathsf{Setup}_{\mathsf{GM}}(\mathsf{pp})$ This algorithm is run by the group manger GM. It takes as input the public parameters pp and outputs a key pair $(\mathsf{pk}_{\mathsf{GM}}, \mathsf{sk}_{\mathsf{GM}})$. Meanwhile, GM initializes the group information info and a public registration directory **reg**.

$\mathsf{G}_{\mathcal{R}}(1^\lambda)$ This randomized algorithm takes as input the security parameter λ and outputs public and secret parameters $(\mathsf{pk}_{\mathcal{R}}, \mathsf{sk}_{\mathcal{R}})$ for the relation $\mathcal{R}$. Note that $\mathsf{sk}_{\mathcal{R}}$ is an empty string if a publicly samplable relation $\mathcal{R}$ is considered.

$\mathsf{Sample}_{\mathcal{R}}(\mathsf{pk}_{\mathcal{R}}, \mathsf{sk}_{\mathcal{R}})$ This probabilistic algorithm takes $(\mathsf{pk}_{\mathcal{R}}, \mathsf{sk}_{\mathcal{R}})$ as input and outputs a statement and witness pair (x, w).

$\mathcal{R}(\mathsf{pk}_{\mathcal{R}}, x, w)$ The polynomial-time testing algorithm takes as input $(\mathsf{pk}_{\mathcal{R}}, x, w)$ and returns 1 if and only if (x, w) is in the relation based on the public parameter $\mathsf{pk}_{\mathcal{R}}$.

$\langle \mathsf{Join}, \mathsf{Issue}(\mathsf{sk}_{\mathsf{GM}}) \rangle (\mathsf{pk}_{\mathsf{GM}}, \mathsf{info}_{\tau_{\text{current}}})$ This is an interactive protocol securely run between a user and the GM. Both the Join and Issue algorithms takes as inputs $\mathsf{pk}_{\mathsf{GM}}$ and $\mathsf{info}_{\tau_{\text{current}}}$ at current time epoch τ_{current} while the the latter algorithm takes $\mathsf{sk}_{\mathsf{GM}}$ as an additional input. Upon successful completion, the algorithm Join outputs a user key pair $(\mathsf{pk}, \mathsf{sk})$ while Issue adds a new record in the directory **reg**. Note that GM may update group information and that **reg** may store information like user identifier or user public key that may be used by GM and OA for later updating and opening.

$\mathsf{GUpdate}(\mathsf{sk}_{\mathsf{GM}}, \mathcal{S}, \mathsf{info}_{\tau_{\text{current}}}, \mathbf{reg})$ This algorithm is run by the GM who will advance the epoch and update the group information. Given the secret key $\mathsf{sk}_{\mathsf{GM}}$, a set $\mathcal{S}$ of active users to be deleted from the group, current group information $\mathsf{info}_{\tau_{\text{current}}}$, and the directory **reg**, the GM computes new group information $\mathsf{info}_{\tau_{\text{new}}}$ and may update the directory **reg** as well. If there is no change to the group information or $\mathcal{S}$ contains inactive users (who has never joined the group yet or who has been revoked from the group), this algorithm aborts.

$\mathsf{Enc}(\mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathsf{OA}}, \mathsf{info}_\tau, w, \mathsf{pk}, L)$ This randomized encryption algorithm is run by the sender who wishes to encrypt a witness w for its chosen user pk. It returns a ciphertext ψ with a certain label L. As in [34], L is a public string bound to the ciphertext that may contain some transaction related data or be empty. If pk is not an active user at current time epoch τ or $\mathcal{R}(\mathsf{pk}_{\mathcal{R}}, x, w) = 0$, this algorithm aborts. Let coins_ψ be the random coins used to generate ψ.

$\mathcal{P}(\mathsf{pp}, \mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathsf{OA}}, \mathsf{info}_\tau, \mathsf{pk}_{\mathcal{R}}, x, \psi, L, w, \mathsf{pk}, \mathsf{coins}_\psi)$ This randomized proof algorithm is run by the sender who acts as a prover and demonstrates the honest computation of ciphertext ψ. Given all the inputs, it outputs a proof π_ψ. The proof ensures that there exists a certified and active group member at time τ, who is able to decrypt ψ and obtain w' such that $\mathcal{R}(\mathsf{pk}_{\mathcal{R}}, x, w') = 1$, and whose public key is encrypted under $\mathsf{pk}_{\mathsf{OA}}$ and can be later revealed using the OA's secret key $\mathsf{sk}_{\mathsf{OA}}$.

$\mathcal{V}((\mathsf{pp}, \mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathsf{OA}}, \mathsf{info}_\tau, \mathsf{pk}_{\mathcal{R}}, x, \psi, L), \pi_\psi)$ This verification algorithm is run by any verifier who on input the tuple $(\mathsf{pp}, \mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathsf{OA}}, \mathsf{info}_\tau, \mathsf{pk}_{\mathcal{R}}, x, \psi, L)$ and a corresponding proof π_ψ outputs bit 1 or 0. If the output is 1, we say the proof π_ψ is valid.

$\mathsf{Dec}(\mathsf{info}_\tau, \mathsf{sk}, \psi, L)$ This decryption algorithm is run by the user in possession of the secret key sk. Given all the inputs, it outputs w' such that $\mathcal{R}(\mathsf{pk}_{\mathcal{R}}, x, w') = 1$ or $\perp$ otherwise.

$\mathsf{Open}(\mathsf{info}_\tau, \mathsf{sk}_{\mathsf{OA}}, \psi, L)$ This opening algorithm is run by the OA who holds the key $\mathsf{sk}_{\mathsf{OA}}$. Given the inputs, it returns an *active* user public key pk or $\perp$ to indicate opening failure.

To ease the notations, we additionally use the following algorithms in the security experiments.

$\mathsf{IsActive}(\mathsf{info}_\tau, \mathsf{pk})$ This algorithm returns 1 if user pk is an active user at time τ and 0 otherwise.

Correctness. Informally, correctness of a GE scheme requires that an honest proof of correct encryption is always valid, that the designated receiver can always recover the encrypted message, and that the GM is capable of identifying the receiver. We model this requirement in the experiment $\mathbf{Expt}^{\mathsf{corr}}_{\mathcal{A}}(1^\lambda)$. Below, we first define some oracles that are accessible to the adversary.

$\mathsf{AddU}(\mathsf{sk}_{\mathsf{GM}})$ This oracle adds an honest user to the group at current time τ_{current}. It simulates the interactive protocol $\langle \mathsf{Join}, \mathsf{Issue}(\mathsf{sk}_{\mathsf{GM}}) \rangle (\mathsf{pk}_{\mathsf{GM}}, \mathsf{info}_{\tau_{\text{current}}})$ and maintains an honest user list HUL. Let the output of Join be $(\mathsf{pk}, \mathsf{sk})$. It then adds pk to HUL.

$\mathsf{GUp}(\cdot)$ This oracle allows the adversary to remove a set of active users from the group at current time epoch τ_{current}. When a set $\mathcal{S}$ is queried, it advances the time epoch to τ_{new} and updates the group information to $\mathsf{info}_{\tau_{\text{new}}}$ by executing the algorithm $\mathsf{GUpdate}(\mathsf{sk}_{\mathsf{GM}}, \mathcal{S}, \mathsf{info}_{\tau_{\text{current}}}, \mathbf{reg})$. As the algorithm GUpdate, it may update the $\mathbf{reg}$.

Definition 1. *Define* $\mathbf{Adv}^{\mathsf{corr}}_{\mathcal{A}}(1^\lambda) = \Pr[\mathbf{Expt}^{\mathsf{corr}}_{\mathcal{A}}(1^\lambda) = 1]$ *as the advantage of an adversary* $\mathcal{A}$ *against correctness in the experiment* $\mathbf{Expt}^{\mathsf{corr}}_{\mathcal{A}}(1^\lambda)$. *A* FDGE *is correct if, for any PPT adversary* $\mathcal{A}$*, the advantage of* $\mathcal{A}$ *is negligible in* λ.

Experiment $\mathbf{Expt}^{\mathsf{corr}}_{\mathcal{A}}(1^\lambda)$
$\mathsf{pp} \leftarrow \mathsf{Setup}_{\mathsf{init}}(1^\lambda)$; $(\mathsf{pk}_{\mathsf{OA}}, \mathsf{sk}_{\mathsf{OA}}) \leftarrow \mathsf{Setup}_{\mathsf{OA}}(\mathsf{pp})$; $(\mathsf{pk}_{\mathsf{GM}}, \mathsf{sk}_{\mathsf{GM}}) \leftarrow \mathsf{Setup}_{\mathsf{GM}}(\mathsf{pp})$.
$(\mathsf{pk}_{\mathcal{R}}, \mathsf{sk}_{\mathcal{R}}) \leftarrow \mathsf{G}_{\mathcal{R}}(1^\lambda)$; $\mathsf{HUL} \leftarrow \emptyset$.
$(\mathsf{pk}, \tau, x, w, L) \leftarrow \mathcal{A}^{\mathsf{AddU}, \mathsf{GUp}}(\mathsf{pp}, \mathsf{pk}_{\mathsf{OA}}, \mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathcal{R}})$.

If $\mathsf{pk} \notin \mathsf{HUL}$ or $\mathsf{info}_\tau = \perp$ or $\mathsf{IsActive}(\mathsf{info}_\tau, \mathsf{pk}) = 0$ or $\mathcal{R}(\mathsf{pk}_\mathcal{R}, x, w) = 0$, return 0.
$\psi \leftarrow \mathsf{Enc}(\mathsf{pk}_\mathsf{GM}, \mathsf{pk}_\mathsf{OA}, \mathsf{info}_\tau, w, \mathsf{pk}, L)$.
$\pi_\psi \leftarrow \mathcal{P}\big(\mathsf{pp}, \mathsf{pk}_\mathsf{GM}, \mathsf{pk}_\mathsf{OA}, \mathsf{info}_\tau, \mathsf{pk}_\mathcal{R}, x, \psi, L, w, \mathsf{pk}, \mathsf{coins}_\psi\big)$.
$w' \leftarrow \mathsf{Dec}(\mathsf{info}_\tau, \mathsf{sk}, \psi, L)$; $\mathsf{pk}' \leftarrow \mathsf{Open}(\mathsf{info}_\tau, \mathsf{sk}_\mathsf{OA}, \psi, L)$.
If $\mathcal{V}\big((\mathsf{pp}, \mathsf{pk}_\mathsf{GM}, \mathsf{pk}_\mathsf{OA}, \mathsf{info}_\tau, \mathsf{pk}_\mathcal{R}, x, \psi, L), \pi_\psi\big) = 0$ or $w' \neq w$ or $\mathsf{pk}' \neq \mathsf{pk}$, return 1 otherwise return 0.

2.2 Formulation of the Security Requirements

We now present three security requirements: message secrecy, anonymity, and soundness for FDGE, which are carefully adapted from the dynamic case. We formulate those requirements through experiments that are run between a challenger and an adversary. As mentioned earlier, the adversary is empowered with attack capability to the maximum extent possible. Specifically, in the definition of message secrecy and anonymity, it fully corrupts GM and/or OA and generates keys arbitrarily on behalf of them. Regarding soundness, only partial corruption of the OA whose key is still honestly generated is allowed. Details of the security requirements are described below.

Message Secrecy. This security notion protects the appointed receiver from a malicious adversary who tries to extract the information about the encrypted message. It requires that the adversary cannot distinguish a ciphertext that is an encryption of a real witness or encryption of a randomly chosen one even though it could fully corrupt the GM, the OA, and all group members except one that is chosen as the receiver. We model this requirement using $\mathbf{Expt}^{\mathsf{sec}-b}_\mathcal{A}(1^\lambda)$ for $b \in \{0, 1\}$. In the following, we define some oracles that will be used in the experiment.

- $\mathsf{USER}()$ This oracle simulates the algorithm Join, when interacted with adversary $\mathcal{A}$ who plays the role of GM, to introduce an honest user to the group at current time τ_{current}. it maintains an honest user list HUL. Let the output of this oracle be $(\mathsf{pk}, \mathsf{sk})$ and add pk to HUL.
- $\mathsf{RevealU}(\cdot)$ This oracle allows the adversary to learn an honest user secret key. It maintains a bad user list BUL. When a user public key pk is queried, it returns the corresponding secret key sk and adds pk to BUL if $\mathsf{pk} \notin \mathsf{BUL}$, and aborts otherwise.
- $\mathsf{CH}^b_{\mathsf{ror}}(\tau, \mathsf{pk}, w, L)$ This is a real-or-random challenge oracle which is only called once. It returns $(\psi, \mathsf{coins}_\psi)$ such that $\psi \leftarrow \mathsf{Enc}(\mathsf{pk}_\mathsf{GM}, \mathsf{pk}_\mathsf{OA}, \mathsf{info}_\tau, w, \mathsf{pk}, L)$ if $b = 1$, whereas if $b = 0$ $\psi \leftarrow \mathsf{Enc}(\mathsf{pk}_\mathsf{GM}, \mathsf{pk}_\mathsf{OA}, \mathsf{info}_\tau, w', \mathsf{pk}, L)$ where w' is sampled uniformly from the space of all possible plaintexts. In both cases, coins_ψ are the random coins used for the computation of the challenged ciphertext ψ.
- $\mathsf{DEC}(\mathsf{sk}, \cdot)$ This is an oracle for the decryption function Dec. When (ψ, τ, L) is queried to this oracle, it returns the output of $\mathsf{Dec}(\mathsf{info}_\tau, \mathsf{sk}, \psi, L)$. When a tuple $(\mathsf{pk}, \psi, \tau, L)$ should be rejected by this oracle, we write $\mathsf{DEC}^{\neg(\psi,\tau,L,)}(\cdot)$.

$\mathsf{PROVE}^b_{\mathcal{P},\mathcal{P}'}(\mathsf{pk}, \tau, \mathsf{pk}_{\mathcal{R}}, x, w, L, \psi, \mathsf{coins}_\psi)$ This oracle can be invoked a polynomial number times. It generates proofs of validity of the challenged ciphertext. If $b = 1$, let $\pi_\psi \leftarrow \mathcal{P}\big(\mathsf{pp}, \mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathsf{OA}}, \mathsf{info}_\tau, \mathsf{pk}_{\mathcal{R}}, x, \psi, L, w, \mathsf{pk}, \mathsf{coins}_\psi\big)$ and return the output π_ψ. If $b = 0$, it runs a simulator $\mathcal{P}'$ that takes the same inputs as $\mathcal{P}$ except (w, coins_ψ) and returns whatever $\mathcal{P}'$ outputs.

In the experiment $\mathbf{Expt}^{\mathsf{sec}-b}_{\mathcal{A}}(1^\lambda)$, the adversary $\mathcal{A}$ fully controls the GM and the OA, and enrolls honest users to the group by interacting with the oracle USER. It is entitled to corrupt at most all but one honest users by querying the $\mathsf{RevealU}$ oracle and to update the group information, insofar as info and $\mathbf{reg}$ are well-formed. At some point, the adversary chooses a targeted receiver pk^* and has access to the DEC oracle with respect to pk^*. It then specifies a certain epoch τ^*, a label L^* together with the relation $\mathsf{pk}^*_{\mathcal{R}}$ and the statement witness pair (x^*, w^*). Afterwards, the challenger encrypts the witness w^* if $b = 1$ or a random message if $b = 0$ to the receiver pk^*, and sends the resultant ciphertext ψ^* to $\mathcal{A}$. After receiving it, $\mathcal{A}$ is allowed to query the PROVE oracle for proofs of its validity and still has access to the DEC oracle with respect to pk^* with the natural restriction that (ψ^*, τ^*, L^*) is forbidden. Finally, $\mathcal{A}$ is asked to guess the challenger's choice.

Definition 2. *Let the advantage of an adversary $\mathcal{A}$ against message secrecy be $\mathbf{Adv}^{\mathsf{sec}-b}_{\mathcal{A}}(1^\lambda) = |\Pr[\mathbf{Expt}^{\mathsf{sec}-1}_{\mathcal{A}}(1^\lambda) = 1] - \Pr[\mathbf{Expt}^{\mathsf{sec}-0}_{\mathcal{A}}(1^\lambda) = 1]|$. A FDGE satisfies message secrecy if, for any PPT adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ is negligible in λ.*

Experiment $\mathbf{Expt}^{\mathsf{sec}-b}_{\mathcal{A}}(1^\lambda)$
$\mathsf{pp} \leftarrow \mathsf{Setup}_{\mathsf{init}}(1^\lambda)$; $(\mathsf{aux}, \mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathsf{OA}}) \leftarrow \mathcal{A}(\mathsf{pp})$; $\mathsf{HUL} \leftarrow \emptyset$, $\mathsf{BUL} \leftarrow \emptyset$.
Throughout the experiment, if info or $\mathbf{reg}$ is not well-formed, return 0.
$(\mathsf{pk}^*, \mathsf{aux}) \leftarrow \mathcal{A}^{\mathsf{USER},\mathsf{RevealU}}(\mathsf{aux})$; if $\mathsf{pk}^* \notin \mathsf{HUL} \setminus \mathsf{BUL}$, return 0.
Let sk^* be the corresponding secret key of pk^*.
$(\tau^*, \mathsf{pk}^*_{\mathcal{R}}, x^*, w^*, L^*) \leftarrow \mathcal{A}^{\mathsf{DEC}(\mathsf{sk}^*,\cdot)}(\mathsf{aux})$.
If $\mathsf{IsActive}(\mathsf{info}_{\tau^*}, \mathsf{pk}^*) = 0$ or $\mathcal{R}(\mathsf{pk}^*_{\mathcal{R}}, x^*, w^*) = 0$ return 0.
$(\psi^*, \mathsf{coins}_{\psi^*}) \leftarrow \mathsf{CH}^b_{\mathsf{ror}}(\tau^*, \mathsf{pk}^*, w^*, L^*)$.
Let $\eth^* = (\mathsf{pk}^*, \tau^*, \mathsf{pk}^*_{\mathcal{R}}, x^*, w^*, L^*, \psi^*, \mathsf{coins}_{\psi^*})$.
$b' \leftarrow \mathcal{A}^{\mathsf{PRVOE}^b_{\mathcal{P},\mathcal{P}'}(\eth^*),\mathsf{DEC}^{\neg(\psi^*,\tau^*,L^*)}(\mathsf{sk}^*,\cdot)}(\mathsf{aux}, \psi^*)$.
Return b'.

Anonymity. This notion aims to prevent the adversary from learning information about the identity of the receiver of a ciphertext. It requires that an adversary without possession of the secret key of OA is not capable of distinguishing which one of two group members of its choice is the recipient of a ciphertext. Note that the adversary is forbidden from corrupting these two challenged members since they know whether a ciphertext is intended for them by simply decrypting it. We model this requirement in $\mathbf{Expt}^{\mathsf{anon}-b}_{\mathcal{A}}(1^\lambda)$ for $b \in \{0, 1\}$, which will utilize the following challenge oracle $\mathsf{CH}^b_{\mathsf{anon}}$ and opening oracle OPEN.

$\mathsf{CH}^b_{\mathsf{anon}}(\tau, \mathsf{pk}_0, \mathsf{pk}_1, w, L)$ This is a challenge oracle that can be called only once. It returns $(\psi, \mathsf{coins}_\psi)$ such that $\psi \leftarrow \mathsf{Enc}(\mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathsf{OA}}, \mathsf{info}_\tau, w, \mathsf{pk}_b, L)$.

$\mathsf{OPEN}(\mathsf{sk}_{\mathsf{OA}}, \cdot)$ This is an oracle for the opening algorithm Open. When decryption of a tuple (ψ, τ, L) is requested, it returns $\mathsf{Open}(\mathsf{info}_\tau, \mathsf{sk}_{\mathsf{OA}}, \psi, L)$. When a tuple (ψ, τ, L) is forbidden, we write $\mathsf{OPEN}^{\neg(\psi,\tau,L)}(\mathsf{sk}_{\mathsf{OA}}, \cdot)$.

In the experiment, the adversary $\mathcal{A}$ can fully corrupt the GM. By interacting with the oracles $\mathsf{USER}, \mathsf{RevealU}$, it can also introduce honest users to the group and learn up to all but two secret keys at a later point. As in the $\mathbf{Expt}_{\mathcal{A}}^{\mathsf{sec}-b}(1^\lambda)$, $\mathcal{A}$ is allowed to update the group at its will, provided that the group information info and $\mathbf{reg}$ are well-formed. Moreover, $\mathcal{A}$ has access to the $\mathsf{OPEN}(\mathsf{sk}_{\mathsf{OA}}, \cdot)$ oracle. At some point, $\mathcal{A}$ specifies two targeted receivers $\mathsf{pk}_0^*, \mathsf{pk}_1^*$ and is granted access to the DEC oracle with respect to both recipients. Next, it outputs a specific epoch τ^* and $(\mathsf{pk}_{\mathcal{R}}^*, x^*, w^*)$ to the challenger, who will encrypt the witness to receiver pk_b^*. Thereafter, the challenger sends the challenge ciphertext ψ^* to $\mathcal{A}$. The latter is further allowed to query the proof of validity of ψ^* and accessible to oracles $\mathsf{DEC}(\mathsf{sk}_0^*, \cdot), \mathsf{DEC}(\mathsf{sk}_1^*, \cdot), \mathsf{OPEN}(\mathsf{sk}_{\mathsf{OA}}, \cdot)$ with the constraint that the tuple (ψ^*, τ^*, L^*) is not queried to any of the oracles. Lastly, $\mathcal{A}$ is asked to guess which one of the two users is the challenger's choice.

Definition 3. *Define the advantage of an adversary $\mathcal{A}$ against anonymity as $\mathbf{Adv}_{\mathcal{A}}^{\mathsf{anon}}(1^\lambda) = |\Pr[\mathbf{Expt}_{\mathcal{A}}^{\mathsf{anon}-1}(1^\lambda) = 1] - \Pr[\mathbf{Expt}_{\mathcal{A}}^{\mathsf{anon}-0}(1^\lambda) = 1]|$. A FDGE satisfies anonymity if, for any PPT adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ is negligible in λ.*

Experiment $\mathbf{Expt}_{\mathcal{A}}^{\mathsf{anon}-b}(1^\lambda)$
$\mathsf{pp} \leftarrow \mathsf{Setup}_{\mathsf{init}}(1^\lambda)$; $(\mathsf{pk}_{\mathsf{OA}}, \mathsf{sk}_{\mathsf{OA}}) \leftarrow \mathsf{Setup}_{\mathsf{OA}}(\mathsf{pp})$; $(\mathsf{aux}, \mathsf{pk}_{\mathsf{GM}}) \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{pk}_{\mathsf{OA}})$.
$\mathsf{HUL} \leftarrow \emptyset$, $\mathsf{BUL} \leftarrow \emptyset$.
Throughout the experiment, if info or $\mathbf{reg}$ is not well-formed, return 0.
$(\mathsf{pk}_0^*, \mathsf{pk}_1^*, \mathsf{aux}) \leftarrow \mathcal{A}^{\mathsf{USER},\mathsf{RevealU},\mathsf{OPEN}(\mathsf{sk}_{\mathsf{OA}},\cdot)}(\mathsf{aux})$.
If $\mathsf{pk}_0^* \notin \mathsf{HUL} \setminus \mathsf{BUL}$ or $\mathsf{pk}_1^* \notin \mathsf{HUL} \setminus \mathsf{BUL}$, return 0.
Let sk_0^* and sk_1^* be the secret keys of pk_0^* and pk_1^*, respectively.
$(\tau^*, \mathsf{pk}_{\mathcal{R}}^*, x^*, w^*, L^*, \mathsf{aux}) \leftarrow \mathcal{A}^{\mathsf{DEC}(\mathsf{sk}_0^*,\cdot),\mathsf{DEC}(\mathsf{sk}_1^*,\cdot),\mathsf{OPEN}(\mathsf{sk}_{\mathsf{OA}},\cdot)}(\mathsf{aux})$.
If $\mathsf{IsActive}(\mathsf{info}_{\tau^*}, \mathsf{pk}_0^*) = 0$ or $\mathsf{IsActive}(\mathsf{info}_{\tau^*}, \mathsf{pk}_1^*) = 0$ or $\mathcal{R}(\mathsf{pk}_{\mathcal{R}}^*, x^*, w^*) = 0$ return 0.
$(\psi^*, \mathsf{coins}_{\psi^*}) \leftarrow \mathsf{CH}_{\mathsf{anon}}^b(\tau^*, \mathsf{pk}_0^*, \mathsf{pk}_1^*, w^*, L^*)$.
Let $\mathfrak{F}^* = (\mathsf{pp}, \mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathsf{OA}}, \mathsf{info}_{\tau^*}, \mathsf{pk}_{\mathcal{R}}^*, x^*, \psi^*, L^*, w^*, \mathsf{pk}_b^*, \mathsf{coins}_{\psi^*})$.
Let $t^* = (\psi^*, \tau^*, L^*)$.
$b' \leftarrow \mathcal{A}^{\mathcal{P}(\mathfrak{F}^*),\mathsf{DEC}^{\neg t^*}(\mathsf{sk}_0^*,\cdot),\mathsf{DEC}^{\neg t^*}(\mathsf{sk}_1^*,\cdot),\mathsf{OPEN}^{\neg t^*}(\mathsf{sk}_{\mathsf{OA}},\cdot)}(\mathsf{aux}, \psi^*)$.
Return b'.

Soundness. This notion requires that the adversary cannot generate a ciphertext with a valid proof associated with time epoch τ such that (1) the opening of the ciphertext is a public key that does not belong to any active group member at time τ, (2) the revealed public key is not in the language $\mathcal{L}_{\mathsf{pk}}^{\mathsf{pp}}$ of valid public keys, (3) the ciphertext is not in the space $\mathcal{L}_{\mathsf{ciphertext}}^{(\mathsf{pk}_{\mathsf{GM}},\mathsf{pk}_{\mathsf{OA}},\tau,\mathsf{pk}_{\mathcal{R}},x,L,\mathsf{pk})}$ of valid ciphertexts. Note that $\mathcal{L}_{\mathsf{pk}}^{\mathsf{pp}} = \{\mathsf{pk} : \exists\ \mathsf{sk}$ such that $(\mathsf{pk}, \mathsf{sk})$ is a valid user key pair$\}$ and

that

$$\mathcal{L}_{\mathsf{ciphertext}}^{(\mathsf{pk_{GM}},\mathsf{pk_{OA}},\tau,\mathsf{pk}_{\mathcal{R}},x,L,\mathsf{pk})} = \{\psi : \exists\ w \text{ such that } \psi = \mathsf{Enc}(\mathsf{pk_{GM}}, \mathsf{pk_{OA}}, \mathsf{info}_\tau, w, \mathsf{pk}, L), \\ \mathcal{R}(\mathsf{pk}_{\mathcal{R}}, x, w) = 1, \text{ and } \mathsf{IsActive}(\mathsf{info}_\tau, \mathsf{pk}) = 1\}.$$

We model this requirement in the experiment $\mathbf{Expt}_{\mathcal{A}}^{\mathsf{sound}}(1^\lambda)$. The adversary is given the secret key of OA and is permitted to adaptively register users to the group through oracle queries $\mathsf{REG}(\mathsf{sk_{GM}})$, as defined below. In addition, it can remove some users from the group by querying the oracle $\mathsf{GUp}(\cdot)$.

$\mathsf{REG}(\mathsf{sk_{GM}})$ This oracle simulates the GM and runs the algorithm Issue. When queried by adversary $\mathcal{A}$ who plays the role of a user, it interacts with $\mathcal{A}$ and if successful registers an adversarially controlled user to the group at current time τ_{current}. As the algorithm Issue, it maintains a public directory **reg** and may update the group information as well.

Definition 4. *Define* $\mathbf{Adv}_{\mathcal{A}}^{\mathsf{sound}}(1^\lambda) = \Pr[\mathbf{Expt}_{\mathcal{A}}^{\mathsf{sound}}(1^\lambda) = 1]$ *as the advantage of an adversary* $\mathcal{A}$ *against soundness in the experiment* $\mathbf{Expt}_{\mathcal{A}}^{\mathsf{sound}}(1^\lambda)$. *A* FDGE *satisfies soundness if, for any PPT adversary* $\mathcal{A}$*, the advantage of* $\mathcal{A}$ *is negligible in* λ.

Experiment $\mathbf{Expt}_{\mathcal{A}}^{\mathsf{sound}}(1^\lambda)$
$\mathsf{pp} \leftarrow \mathsf{Setup}_{\mathsf{init}}(1^\lambda)$; $(\mathsf{pk_{OA}}, \mathsf{sk_{OA}}) \leftarrow \mathsf{Setup_{OA}}(\mathsf{pp})$; $(\mathsf{pk_{GM}}, \mathsf{sk_{GM}}) \leftarrow \mathsf{Setup_{GM}}(\mathsf{pp})$.
$(\tau, \mathsf{pk}_{\mathcal{R}}, x, \psi, L, \pi_\psi, \mathsf{aux}) \leftarrow \mathcal{A}^{\mathsf{REG},\mathsf{GUp}}(\mathsf{pp}, \mathsf{pk_{GM}}, \mathsf{pk_{OA}}, \mathsf{sk_{OA}})$.
If $\mathcal{V}\big((\mathsf{pp}, \mathsf{pk_{GM}}, \mathsf{pk_{OA}}, \mathsf{info}_\tau, \mathsf{pk}_{\mathcal{R}}, x, \psi, L), \pi_\psi\big) = 0$, return 0.
$\mathsf{pk} \leftarrow \mathsf{Open}(\mathsf{info}_\tau, \mathsf{sk_{OA}}, \psi, L)$.
If $\mathsf{IsActive}(\mathsf{info}_\tau, \mathsf{pk}) = 0$ or $\mathsf{pk} \notin \mathcal{L}_{\mathsf{pk}}^{\mathsf{pp}}$ or $\psi \notin \mathcal{L}_{\mathsf{ciphertext}}^{(\mathsf{pk_{GM}},\mathsf{pk_{OA}},\mathsf{pk}_{\mathcal{R}},x,L,\mathsf{pk})}$, return 1 else return 0.

3 Some Background on Code-Based Cryptography and Stern-like Zero-Knowledge Protocols

NOTATIONS. Let $a, b \in \mathbb{Z}$. Denote $[a, b]$ as the set $\{a, \ldots, b\}$. We simply write $[b]$ when $a = 1$. Let $\oplus$ denote the bit-wise addition operation modulo 2. If S is a finite set, then $x \xleftarrow{\$} S$ means that x is chosen uniformly at random from S. Throughout this paper, all vectors are column vectors. When concatenating vectors $\mathbf{x} \in \{0,1\}^m$ and $\mathbf{y} \in \{0,1\}^k$, for simplicity, we use the notation $(\mathbf{x}\|\mathbf{y}) \in \{0,1\}^{m+k}$ instead of $(\mathbf{x}^\top\|\mathbf{y}^\top)^\top$. The Hamming weight of vector $\mathbf{x} \in \{0,1\}^m$ is denoted by $wt(\mathbf{x})$. The Hamming distance between vectors $\mathbf{x}, \mathbf{y} \in \{0,1\}^m$ is denoted by $d_H(\mathbf{x}, \mathbf{y})$, and is equal to $wt(\mathbf{x} \oplus \mathbf{y})$. Denote by $\mathsf{B}(n, \omega)$ the set of all binary vectors of length n with Hamming weight ω, and by S_n the symmetric group of all permutations of n elements.

Let $\mathbb{Z}^+$ be the set consisting of all positive integers. For $c \in \mathbb{Z}^+$ and k divisible by c, define the following.

$\mathsf{Regular}(k,c)$ is the set containing all vectors of the form $\mathbf{w} = (\mathbf{w}_1 \| \ldots \| \mathbf{w}_{\frac{k}{c}}) \in \{0,1\}^{2^c \cdot \frac{k}{c}}$ that consists of $\frac{k}{c}$ blocks, each of which is an element of $\mathsf{B}(2^c, 1)$. We call $\mathbf{w}$ *regular word* if $\mathbf{w} \in \mathsf{Regular}(k,c)$ for some k, c.

2-$\mathsf{Regular}(k,c)$ is the set of all vectors $\mathbf{x} \in \{0,1\}^{2^c \cdot \frac{k}{c}}$ such that there exist regular words $\mathbf{v}, \mathbf{w} \in \mathsf{Regular}(k,c)$ satisfying $\mathbf{x} = \mathbf{v} \oplus \mathbf{w}$. We call $\mathbf{w}$ 2-*regular word* if $\mathbf{w} \in$ 2-$\mathsf{Regular}(k,c)$ for some k, c.

$\mathsf{RE} : \{0,1\}^k \to \{0,1\}^{2^c \cdot \frac{k}{c}}$ is a regular encoding function that maps $\mathbf{x} \in \{0,1\}^k$ to a vector $\mathsf{RE}(\mathbf{x}) \in \{0,1\}^{2^c \cdot \frac{k}{c}}$ as follows. Let $\mathbf{x} = (\mathbf{x}_1 \| \ldots \| \mathbf{x}_{\frac{k}{c}})$, where $\mathbf{x}_j = (x_{j,1}, \ldots, x_{j,c})^\top$ for $j \in [\frac{k}{c}]$. Compute $t_j = \sum_{i=1}^{c} 2^{c-i} \cdot x_{j,i}$. Let $\Delta_{2^c}(\mathbf{x}_j) \in \mathsf{B}(2^c, 1)$ whose sole 1 entry is at the t_j-th position for $t_j \in [0, 2^c - 1]$. $\mathsf{RE}(\mathbf{x})$ is then defined as $(\Delta_{2^c}(\mathbf{x}_1) \| \Delta_{2^c}(\mathbf{x}_2) \| \cdots \| \Delta_{2^c}(\mathbf{x}_{\frac{k}{c}})) \in \mathsf{Regular}(k,c)$.

We now recall the 2-$\mathsf{RNSD}_{n,k,c}$ problem, introduced by Augot, Finiasz and Sendrier (AFS) [4,5]. The problem asks to find low-weight 2-regular codewords in random binary linear codes, and is closely related to the Small Codeword Problem [45] and binary Shortest Vector Problem [3], with an additional constraint that the solution codeword must be 2-regular.

Definition 5. *The* 2-$\mathsf{RNSD}_{n,k,c}$ *Problem: given a uniformly random matrix* $\mathbf{B} \in \mathbb{Z}_2^{n \times m}$*, where* $m = 2^c \cdot k/c$*, find a non-zero vector* $\mathbf{z} \in$ *2-*$\mathsf{Regular}(k,c) \subseteq \{0,1\}^m$ *such that* $\mathbf{B} \cdot \mathbf{z} = \mathbf{0}$.

The problem is shown to be NP-complete in the worst case [4]. All the previous algorithms require exponential times in the security parameter in terms of practical use and a comprehensive discussion of known attacks and parameter settings was given in [11].

The Modified AFS Hash Function Nguyen et al. [53] recently modified the AFS hash function family [4,5] so that it takes 2 inputs (instead of just 1) and hence is suitable for building Merkle hash trees. The definition is given below.

Definition 6. *Let* $m = 2 \cdot 2^c \cdot n/c$*. The function family* $\mathcal{H}$ *mapping* $\{0,1\}^n \times \{0,1\}^n$ *to* $\{0,1\}^n$ *is defined as* $\mathcal{H} = \{h_{\mathbf{B}} \mid \mathbf{B} \in \mathbb{Z}_2^{n \times m}\}$*, where for* $\mathbf{B} = [\mathbf{B}_0 | \mathbf{B}_1]$ *with* $\mathbf{B}_0, \mathbf{B}_1 \in \mathbb{Z}_2^{n \times m/2}$*, and for any* $(\mathbf{u}_0, \mathbf{u}_1) \in \{0,1\}^n \times \{0,1\}^n$*, we have:*

$$h_{\mathbf{B}}(\mathbf{u}_0, \mathbf{u}_1) = \mathbf{B}_0 \cdot \mathsf{RE}(\mathbf{u}_0) \oplus \mathbf{B}_1 \cdot \mathsf{RE}(\mathbf{u}_1) \in \{0,1\}^n.$$

The collision resistance of the hash function family relies on the hardness of the 2-RNSD problem.

Lemma 1 ([53]). *The function family* $\mathcal{H}$*, defined in Definition 6 is collision-resistant, assuming the hardness of the* 2-$\mathsf{RNSD}_{n,2n,c}$ *problem.*

3.1 Code-Based Merkle-tree Accumulators

We now recall the code-based Merkle-tree accumulators suggested in [53].

$\mathsf{Setup}_{\mathsf{Acc}}(\lambda)$. Given $n = \mathcal{O}(\lambda)$, $c = \mathcal{O}(1)$ and $m = 2 \cdot 2^c \cdot n/c$. Sample $\mathbf{B} \xleftarrow{\$} \mathbb{Z}_2^{n \times m}$, and output the public parameter $pp = \mathbf{B}$.

$\mathsf{Accu}_{\mathbf{B}}(R = \{\mathbf{d}_0, \ldots, \mathbf{d}_{N-1}\} \subseteq (\{0,1\}^n)^N)$. Assume $N = 2^\ell$ without loss of generality. Re-write $\mathbf{d}_j$ as $\mathbf{u}_{\ell,j}$ and call $\mathbf{d}_j$ the leaf value of the leaf node $\mathsf{bin}(j)$ for $j \in [0, N-1]$. Build a binary tree upon N leaves $\mathbf{u}_{\ell,0}, \ldots, \mathbf{u}_{\ell,2^\ell-1}$ in the following way. For $k \in \{\ell-1, \ell-2, \ldots, 1, 0\}$ and $i \in [0, 2^k - 1]$, compute $\mathbf{u}_{k,i} = h_{\mathbf{B}}(\mathbf{u}_{k+1,2i}, \mathbf{u}_{k+1,2i+1})$. Output the accumulated value $\mathbf{u} = \mathbf{u}_{0,0}$.

$\mathsf{WitGen}_{\mathbf{B}}(R, \mathbf{d})$. If $\mathbf{d} \notin R$, the algorithm outputs $\perp$. Otherwise, it outputs the witness w for $\mathbf{d}$ as follows.

1. Set $\mathbf{d} = \mathbf{d}_j$ for some $j \in [0, N-1]$. Re-write $\mathbf{d}_j = \mathbf{u}_{\ell,j}$. Let $\mathsf{bin}(j) = [j_1 | \ldots | j_\ell]^\top \in \{0,1\}^\ell$ be the binary representation of j.
2. Consider the path from $\mathbf{u}_{\ell,j}$ to the root $\mathbf{u}$, the witness w then consists of $\mathsf{bin}(j)$ as well as all the sibling nodes of the path. Let $w = \big(\mathsf{bin}(j), (\mathbf{w}_\ell, \ldots, \mathbf{w}_1)\big) \in \mathbb{Z}_2^\ell \times \big(\mathbb{Z}_2^n\big)^\ell$. We give an example in Figure 1.

$\mathsf{Verify}_{\mathbf{B}}\big(\mathbf{u}, \mathbf{d}, w\big)$. Let w be of the following form:

$$w = \big([j_1| \cdots |j_\ell]^\top, (\mathbf{w}_\ell, \ldots, \mathbf{w}_1)\big).$$

This algorithm then computes $\mathbf{v}_\ell, \ldots, \mathbf{v}_0$. Let $\mathbf{v}_\ell = \mathbf{d}$ and

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

Output 1 if $\mathbf{v}_0 = \mathbf{u}$ or 0 otherwise.

Lemma 2 ([53]). *Assume that the* $\mathsf{2\text{-}RNSD}_{n,2n,c}$ *problem is hard, then the given accumulator scheme is secure.*

3.2 An Efficient Updating Algorithm

To construct our fully dynamic group encryption, it is required to update the accumulated value $\mathbf{u}_{0,0}$ whenever a user joins or is revoked from the group. It is thus desirable that one can update $\mathbf{u}_{0,0}$ without having to reconstruct the whole tree. We follow the approach of Ling et al. [44], who constructed their dynamic group signature by introducing an efficient updating algorithm to the static (lattice-based) Merkle tree accumulator [38]. Moreover, their updating algorithm $\mathsf{TUpdate}_{\mathbf{B}}$ runs in time $\mathcal{O}(\ell)$: they simply update the path from a leaf node to the root if that leaf node value needs to be modified. Specifically, given input a bit-string $\mathsf{bin}(j) \in \{0,1\}^\ell$ and a value $\mathbf{d}^* \in \{0,1\}^n$, the algorithm $\mathsf{TUpdate}_{\mathbf{B}}$ proceeds as follows.

$\mathsf{TUpdate}_{\mathbf{B}}(\mathsf{bin}(j), \mathbf{d}^*)$ Let $\mathbf{d}_j$ be the existing leaf value of the leaf node $\mathsf{bin}(j)$. It executes the algorithm $\mathsf{WitGen}_{\mathbf{B}}(R, \mathbf{d}_j)$, obtaining $w = (\mathsf{bin}(j), (\mathbf{w}_\ell, \ldots, \mathbf{w}_1))$. It then sets $\mathbf{v}_\ell = \mathbf{d}^*$ and recursively computes $\mathbf{v}_{\ell-1}, \ldots, \mathbf{v}_0$ as in (1). Finally, for $i \in [0, \ell]$, it sets $\mathbf{u}_{i, \lfloor \frac{j}{2^{\ell-i}} \rfloor} = \mathbf{v}_i$.

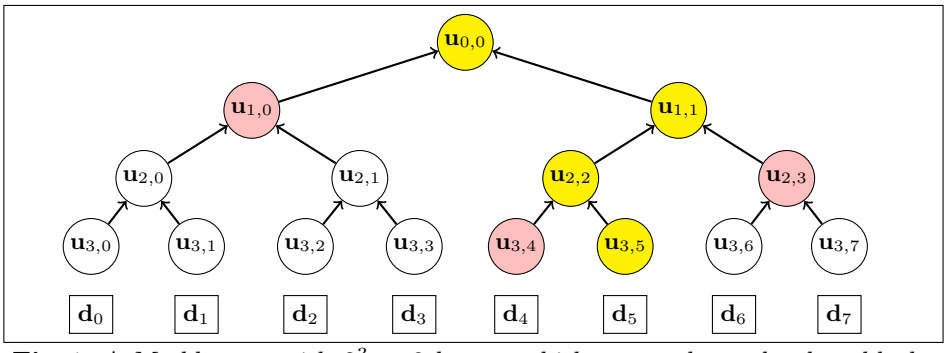

Fig. 1: A Merkle tree with $2^3 = 8$ leaves, which accumulates the data blocks $\mathbf{d}_0, \ldots, \mathbf{d}_7$ into the value $\mathbf{u} = \mathbf{u}_{0,0}$ at the root. Let $j = 5$. Then the path from $\mathbf{u}_{3,5} = \mathbf{d}_5$ to the root consists of the yellow nodes, whose siblings are the pink nodes. Hence, $\mathsf{bin}(j)$ as well as the pink nodes form a witness to the fact that $\mathbf{d}_5$ is accumulated in $\mathbf{u}$. If one needs to update $\mathbf{d}_5$ to $\mathbf{d}^*$, it suffices to update the yellow nodes.

3.3 The Randomized McEliece Encryption Scheme

Now we recall the original McEliece [47] encryption scheme and its randomized version proposed in [54]. The following describes the randomized one.

- $\mathsf{Setup}_{\mathsf{ME}}(1^\lambda)$: Let $n_e = n_e(\lambda), k_e = k_e(\lambda), t_e = t_e(\lambda)$ be the parameters for a binary $[n_e, k_e, 2t_e + 1]$ Goppa code. Choose k_1, $k_2 \in \mathbb{Z}$ such that $k_e = k_1 + k_2$. Let $\mathbb{Z}_2^{k_2}$ be the plaintext space.
- $\mathsf{KeyGen}_{\mathsf{ME}}(n_\mathsf{e}, k_e, t_\mathsf{e})$: This algorithm outputs the encryption key and decryption key for the randomized McEliece encryption scheme. It works as follows:
  1. Choose a generator matrix $\mathbf{G}' \in \mathbb{Z}_2^{n_\mathsf{e} \times k_e}$ of a randomly selected $[n_\mathsf{e}, k_e, 2t_\mathsf{e} + 1]$ Goppa code. Let $\mathbf{S} \in \mathbb{Z}_2^{k_e \times k_e}$ be a random invertible matrix and $\mathbf{P} \in \mathbb{Z}_2^{n_\mathsf{e} \times n_\mathsf{e}}$ be a random permutation matrix, then compute $\mathbf{G} = \mathbf{P}\mathbf{G}'\mathbf{S} \in \mathbb{Z}_2^{n_\mathsf{e} \times k_e}$. Output encryption key $\mathsf{pk}_\mathsf{ME} = \mathbf{G}$ and decryption key $\mathsf{sk}_\mathsf{ME} = (\mathbf{S}, \mathbf{G}', \mathbf{P})$.
- $\mathsf{Enc}_{\mathsf{ME}}(\mathsf{pk}_\mathsf{ME}, \mathbf{m})$: On input a message $\mathbf{m} \in \mathbb{Z}_2^{k_2}$ and pk_ME, sample $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_2^{k_1}$ and $\mathbf{e} \xleftarrow{\$} \mathsf{B}(n_\mathsf{e}, t_\mathsf{e})$, and then output the ciphertext $\mathbf{c} = \mathbf{G} \cdot \begin{pmatrix} \mathbf{r} \\ \mathbf{m} \end{pmatrix} \oplus \mathbf{e} \in \mathbb{Z}_2^{n_\mathsf{e}}$.
- $\mathsf{Dec}_{\mathsf{ME}}(\mathsf{sk}_\mathsf{ME}, \mathbf{c})$: On input the ciphertext $\mathbf{c}$ and decryption key sk_ME, it works as follows:
  1. Multiply $\mathbf{P}^{-1}$ to the left of the ciphertext $\mathbf{c}$, then apply an error-correcting algorithm. Obtain $\mathbf{m}'' = \mathit{Decode}_{\mathbf{G}'}(\mathbf{P}^{-1} \cdot \mathbf{c})$ where Decode is an error-correcting algorithm with respect to $\mathbf{G}'$. Returns $\perp$ if Decode fails.
  2. Multiply $\mathbf{S}^{-1}$ to the left of the ciphertext $\mathbf{m}''$, then $\mathbf{m}' = \mathbf{S}^{-1} \cdot \mathbf{m}''$, parse $\mathbf{m}' = \begin{pmatrix} \mathbf{r} \\ \mathbf{m} \end{pmatrix}$, where $\mathbf{r} \in \mathbb{Z}_2^{k_1}$ and $\mathbf{m} \in \mathbb{Z}_2^{k_2}$, and return $\mathbf{m}$.

Regarding the original McEliece scheme, it encrypts a message $\mathbf{m} \in \mathbb{Z}_2^{k_e}$ as $\mathbf{c} = \mathbf{G} \cdot \mathbf{m} \oplus \mathbf{e}$ while the decryption algorithm simply outputs $\mathbf{m}' \in \mathbb{Z}_2^{k_e}$. Looking ahead, we employ the original scheme to ensure that a potential user who requests to join the group knows the underlying McEliece decryption key. This is achieved by encrypting a random message under the encryption key chosen by the user and asks the user to output the correct message. As pointed out in [47], the running time of the user who does not know the underlying decryption key can be at least as $k_e^3 \cdot \binom{n_e}{k_e} / \binom{n_e - t_e}{k_e}$[7].

Note that the original McEliece scheme is not even CPA-secure. Fortunately, the above randomized one has pseudorandom ciphertexts which implies CPA-security, under the Decisional McEliece problem $\mathsf{DMcE}(n_e, k_e, t_e)$ and the Decisional Learning Parity with (fixed-weight) Noise problem $\mathsf{DLPN}(n_e, k_1, \mathsf{B}(n_e, t_e))$ in the standard model. The two problems are recalled below.

Definition 7. *The* $\mathsf{DMcE}(n, k, t)$ *problem: given a matrix* $\mathbf{G} \in \mathbb{Z}_2^{n \times k}$*, distinguish whether it is uniformly chosen from* $\mathbb{Z}_2^{n \times k}$ *or is generated by algorithm* $\mathsf{ME.KeyGen}(n, k, t)$ *described above.*

When $n = n(\lambda), k = k(\lambda), t = t(\lambda)$, we say that the $\mathsf{DMcE}(n, k, t)$ problem is hard, if the success probability of any PPT distinguisher is at most $1/2 + \mathsf{negl}(\lambda)$.

Definition 8. *The* $\mathsf{DLPN}(n, k, \mathsf{B}(n, t))$ *problem: given a pair* $(\mathbf{A}, \mathbf{c}) \in \mathbb{Z}_2^{n \times k} \times \mathbb{Z}_2^n$*, distinguish whether it is uniformly chosen from* $\mathbb{Z}_2^{n \times k} \times \mathbb{Z}_2^n$ *or is obtained by choosing* $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_2^{n \times k}$*,* $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_2^k$*,* $\mathbf{e} \xleftarrow{\$} \mathsf{B}(n, t)$ *and outputting* $(\mathbf{A}, \mathbf{A} \cdot \mathbf{r} \oplus \mathbf{e})$*.*

When $n = n(\lambda), k = k(\lambda), t = t(\lambda)$, we say that the $\mathsf{DLPN}(n, k, \mathsf{B}(n, t))$ problem is hard, if the success probability of any PPT distinguisher is at most $1/2 + \mathsf{negl}(\lambda)$.

3.4 Zero-Knowledge Arguments and Stern-like Protocols

We will work with statistical zero-knowledge argument systems, namely, interactive protocols where the zero-knowledge property holds against *any* cheating verifier, while the soundness property only holds against *computationally bounded* cheating provers. More formally, let the set of statements-witnesses $\mathrm{R} = \{(x, w)\} \in \{0,1\}^* \times \{0,1\}^*$ be an NP relation. A two-party game $\langle \mathcal{P}, \mathcal{V} \rangle$ is called an interactive argument system for the relation R with soundness error e if the following conditions hold:

- $\mathsf{Completeness}$. If $(x, w) \in \mathrm{R}$ then $\Pr\big[\langle \mathcal{P}(x, w), \mathcal{V}(y) \rangle = 1\big] = 1$.
- $\mathsf{Soundness}$. If $(x, w) \notin \mathrm{R}$, then $\forall$ PPT $\widehat{\mathcal{P}}$: $\Pr[\langle \widehat{\mathcal{P}}(x, w), \mathcal{V}(x) \rangle = 1] \leq e$.

[7] The estimated running time in [47] is actually $k_e^3 \cdot n_e^{k_e} / (n_e - t_e)^{k_e}$, which is a further estimation of ours. The factor k_e^3 is the amount of work involved in solving k_e simultaneous equations with k_e unknown.

An argument system is called statistical zero-knowledge if there exists a PPT simulator $\mathcal{S}(x)$ having oracle access to any $\widehat{\mathcal{V}}(x)$ and producing a simulated transcript that is statistically close to the one of the real interaction between $\mathcal{P}(x,w)$ and $\widehat{\mathcal{V}}(x)$. A related notion is argument of knowledge, which, for three-move protocols (commitment-challenge-response), requires the existence of a PPT extractor taking as input a set of valid transcripts w.r.t. all possible values of the "challenge" to the same "commitment", and outputs w' such that $(x,w') \in \mathrm{R}$.

The statistical zero-knowledge arguments of knowledge presented in this work are Stern-like [58] protocols. In particular, they are Σ-protocols in the generalized sense defined in [32,10] (where 3 valid transcripts are needed for extraction, instead of just 2). The basic protocol consists of 3 moves: commitment, challenge, response. If a statistically hiding and computationally binding string commitment is employed in the first move, then one obtains a statistical zero-knowledge argument of knowledge (ZKAoK) with perfect completeness, constant soundness error 2/3. In many applications, the protocol is repeated a sufficient number of times to make the soundness error negligibly small. For instance, to achieve soundness error 2^{-80}, it suffices to repeat the basic protocol 137 times.

An abstraction of Stern's protocols. We recall an abstraction, adapted from [36], which captures the sufficient conditions to run a Stern-like protocol. Looking ahead, this abstraction will be helpful for us in presenting our ZK argument systems: we will reduce the relations we need to prove to instances of the abstract protocol, using our specific techniques. Let K, L be positive integers, where $L \geq K$, and let VALID be a subset of $\{0,1\}^L$. Suppose that $\mathcal{S}$ is a finite set such that one can associate every $\phi \in \mathcal{S}$ with a permutation Γ_ϕ of L elements, satisfying the following conditions:

$$\begin{cases} \mathbf{w} \in \mathsf{VALID} \iff \Gamma_\phi(\mathbf{w}) \in \mathsf{VALID}, \\ \text{If } \mathbf{w} \in \mathsf{VALID} \text{ and } \phi \text{ is uniform in } \mathcal{S}, \text{ then } \Gamma_\phi(\mathbf{w}) \text{ is uniform in } \mathsf{VALID}. \end{cases} \tag{2}$$

We aim to construct a statistical ZKAoK for the following abstract relation:

$$\mathrm{R}_{\text{abstract}} = \big\{ \big((\mathbf{M}, \mathbf{v}); \mathbf{w} \big) \in \mathbb{Z}_2^{K \times L} \times \mathbb{Z}_2^K \times \mathsf{VALID} : \mathbf{M} \cdot \mathbf{w} = \mathbf{v} \big\}.$$

The conditions in (2) play a crucial role in proving in ZK that $\mathbf{w} \in \mathsf{VALID}$: To do so, the prover samples $\phi \xleftarrow{\$} \mathcal{S}$ and lets the verifier check that $\Gamma_\phi(\mathbf{w}) \in \mathsf{VALID}$, while the latter cannot learn any additional information about $\mathbf{w}$ thanks to the randomness of ϕ. Furthermore, to prove in ZK that the linear equation holds, the prover samples a masking vector $\mathbf{r}_w \xleftarrow{\$} \mathbb{Z}_2^L$, and convinces the verifier instead that $\mathbf{M} \cdot (\mathbf{w} \oplus \mathbf{r}_w) = \mathbf{M} \cdot \mathbf{r}_w \oplus \mathbf{v}$.

The interaction between prover $\mathcal{P}$ and verifier $\mathcal{V}$ is described in Figure 2. The protocol employs a statistically hiding and computationally binding string commitment scheme COM, for example, the one proposed in [53, Section 3.1].

The properties of the protocols are summarized in Theorem 1.

Theorem 1 ([36,53]). *Assume that* COM *is a statistically hiding and computationally binding string commitment scheme. Then, the protocol in Figure 2 is*

1. **Commitment:** Prover samples $\mathbf{r}_w \xleftarrow{\$} \mathbb{Z}_2^L$, $\phi \xleftarrow{\$} \mathcal{S}$ and randomness ρ_1, ρ_2, ρ_3 for COM. Then he sends $\mathsf{CMT} = \big(C_1, C_2, C_3\big)$ to the verifier, where

$$C_1 = \mathsf{COM}(\phi, \mathbf{M} \cdot \mathbf{r}_w; \rho_1), \;\; C_2 = \mathsf{COM}(\Gamma_\phi(\mathbf{r}_w); \rho_2),$$
$$C_3 = \mathsf{COM}(\Gamma_\phi(\mathbf{w} \oplus \mathbf{r}_w); \rho_3).$$

2. **Challenge:** The verifier sends a challenge $Ch \xleftarrow{\$} \{1, 2, 3\}$ to the prover.
3. **Response:** Depending on Ch, the prover sends RSP computed as follows:
   - $Ch = 1$: Let $\mathbf{t}_w = \Gamma_\phi(\mathbf{w})$, $\mathbf{t}_r = \Gamma_\phi(\mathbf{r}_w)$, and RSP $= (\mathbf{t}_w, \mathbf{t}_r, \rho_2, \rho_3)$.
   - $Ch = 2$: Let $\phi_2 = \phi$, $\mathbf{w}_2 = \mathbf{w} \oplus \mathbf{r}_w$, and RSP $= (\phi_2, \mathbf{w}_2, \rho_1, \rho_3)$.
   - $Ch = 3$: Let $\phi_3 = \phi$, $\mathbf{w}_3 = \mathbf{r}_w$, and RSP $= (\phi_3, \mathbf{w}_3, \rho_1, \rho_2)$.

Verification: Receiving RSP, the verifier proceeds as follows:

- $Ch = 1$: Check that $\mathbf{t}_w \in \mathsf{VALID}$, $C_2 = \mathsf{COM}(\mathbf{t}_r; \rho_2)$, $C_3 = \mathsf{COM}(\mathbf{t}_w \oplus \mathbf{t}_r; \rho_3)$.
- $Ch = 2$: Check that $C_1 = \mathsf{COM}(\phi_2, \mathbf{M} \cdot \mathbf{w}_2 \oplus \mathbf{v}; \rho_1)$, $C_3 = \mathsf{COM}(\Gamma_{\phi_2}(\mathbf{w}_2); \rho_3)$.
- $Ch = 3$: Check that $C_1 = \mathsf{COM}(\phi_3, \mathbf{M} \cdot \mathbf{w}_3; \rho_1)$, $C_2 = \mathsf{COM}(\Gamma_{\phi_3}(\mathbf{w}_3); \rho_2)$.

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

Fig. 2: Stern-like ZKAoK for the relation $\mathrm{R}_{\mathrm{abstract}}$.

a statistical ZKAoK *with perfect completeness, soundness error* $2/3$*, and communication cost* $\mathcal{O}(L)$*. In particular:*

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

3.5 Previous Extension and Permutation Techniques

This section reviews several supporting techniques for Stern-like protocols, that were proposed in previous works and that will be useful for designing the zero-knowledge arguments of the present paper. We first recall the permutation technique that is designed to prove knowledge of a binary vector of fixed hamming weight, which originates from Stern [58].

Technique for handling binary vector with fixed hamming weight. For any $\mathbf{e} \in \mathsf{B}(n, \omega)$ and $\sigma \in \mathsf{S}_n$, it is easy to see that the following equivalence holds [8].

$$\mathbf{e} \in \mathsf{B}(n, \omega) \Longleftrightarrow \sigma(\mathbf{e}) \in \mathsf{B}(n, \omega), \tag{3}$$

[8] Note that for $\mathbf{e} = [e_1 | \cdots | e_m]^\top$, $\sigma(\mathbf{e})$ is defined as $\sigma(e_i) = e_{\sigma(i)}$ for $i \in [n]$.

To show that the vector $\mathbf{e}$ has hamming weight ω, the prover samples a uniformly random permutation $\sigma \in \mathsf{S}_n$ and shows the verifier that $\sigma(\mathbf{e}) \in \mathsf{B}(n, \omega)$. Due to the above equivalence (3), the verifier should be convinced that $\mathbf{e} \in \mathsf{B}(n, \omega)$. Furthermore, $\sigma(\mathbf{e})$ reveal no information about $\mathbf{e}$ due to the uniformity of σ.

The above technique was later developed to prove various forms of secret vectors. We now review the extension and permutation techniques for proving the knowledge of arbitrary binary vectors, non-zero binary vector and well-formed regular words, which were presented in [38], [42] and [53], respectively.

Let $\oplus$ denote the bit-wise addition operation modulo 2. For any bit $b \in \{0, 1\}$, denote by $\overline{b}$ the bit $b = b \oplus 1$. Note that, for any $b, d \in \{0, 1\}$, we have $\overline{b \oplus d} = b \oplus d \oplus 1 = \overline{b} \oplus d$.

Techniques for handling arbitrary binary vectors. To prove the knowledge of a binary vector $\mathbf{x} \in \{0, 1\}^n$, define the extension process and permutation as follows.

- For a binary vector $\mathbf{x} = [\, x_1 \,|\, \ldots \,|\, x_n \,]^\top \in \{0, 1\}^n$, where $n \in \mathbb{Z}^+$, denote by $\mathsf{Encode}(\mathbf{x})$ the vector $[\, \overline{x}_1 \,|\, x_1 \,|\, \ldots \,|\, \overline{x}_n \,|\, x_n \,]^\top \in \{0, 1\}^{2n}$.
- Let $\mathbf{I}_n^* \in \mathbb{Z}_2^{n \times 2n}$ be an extension of the identity matrix $\mathbf{I}_n$, obtained by inserting a zero-column $\mathbf{0}^n$ right before each column of $\mathbf{I}_n$. We have for $\mathbf{x} \in \{0, 1\}^n$,

$$\mathbf{x} = \mathbf{I}_n^* \cdot \mathsf{Encode}(\mathbf{x}). \tag{4}$$

- For $\mathbf{b} = [\, b_1 \,|\, \ldots \,|\, b_n \,]^\top \in \{0, 1\}^n$, define the permutation $F_{\mathbf{b}}$ that transforms vector $\mathbf{z} = [\, z_{1,0} \,|\, z_{1,1} \,|\, \ldots \,|\, z_{n,0} \,|\, z_{n,1} \,]^\top \in \{0, 1\}^{2n}$ into:

$$F_{\mathbf{b}}(\mathbf{z}) = [\, z_{1,b_1} \,|\, z_{1,\overline{b}_1} \,|\, \ldots \,|\, z_{n,b_n} \,|\, z_{n,\overline{b}_n} \,]^\top.$$

  Note that, for any $\mathbf{b}, \mathbf{x} \in \{0, 1\}^n$, we have:

$$\mathbf{z} = \mathsf{Encode}(\mathbf{x}) \iff F_{\mathbf{b}}(\mathbf{z}) = \mathsf{Encode}(\mathbf{x} \oplus \mathbf{b}). \tag{5}$$

The above equivalence (5) is useful in the Stern's framework [58] for proving knowledge of binary witness-vectors. Towards the goal, one encodes $\mathbf{x}$ to $\mathbf{z} = \mathsf{Encode}(\mathbf{x})$, samples a random binary vector $\mathbf{b}$ and permutes $\mathbf{z}$ using $F_{\mathbf{b}}$. Then one demonstrates to the verifier that the permuted vector $F_{\mathbf{b}}(\mathbf{z})$ is of the correct form $\mathsf{Encode}(\mathbf{x} \oplus \mathbf{b})$. Due to (5), the verifier should be convinced that $\mathbf{z}$ is well formed, which further implies the knowledge of a binary vector $\mathbf{x}$. Meanwhile, vector $\mathbf{b}$ serves as a "one-time pad" that perfects hides $\mathbf{x}$. In addition, if we have to show that $\mathbf{x}$ appears somewhere else, we can use the same $\mathbf{b}$ at those places.

Techniques for handling non-zero binary vector. To prove the knowledge of a non-zero binary vector $\mathbf{x} \in \{0, 1\}^n \setminus \{\mathbf{0}^n\}$, we note that applying the above Encode function is not sufficient. It does not allow any way to show that $\mathbf{x}$ is non-zero. Fortunately, Ling et al. [42] suggested a technique that allows proving knowledge of a non-zero binary vector[9]. It first concatenates $n - 1$ "dummy"

[9] In fact, they also proposed a technique on proving knowledge of an arbitrary binary vector $\mathbf{x} \in \{0, 1\}^n$ by first appending n dummy entries to it to obtain $\mathbf{x}^* \in \mathsf{B}(2n, n)$

entries to $\mathbf{x}$ to get $\mathbf{x}^* \in \mathsf{B}(2n-1, n)$ and then convinces the verifier that $\pi(\mathbf{x}^*) \in \mathsf{B}(2n-1, n)$ for $\pi \xleftarrow{\$} \mathsf{S}_{2n-1}$. Let $\mathbf{P}_n^* = [\mathbf{I}_n | \mathbf{0}^{n\times(n-1)}] \in \mathbb{Z}_2^{n\times 2n-1}$. Then for any $\mathbf{x} \in \{0,1\}^n \setminus \{\mathbf{0}^n\}$, we have

$$\mathbf{x} = \mathbf{P}_n^* \cdot \mathbf{x}^*. \tag{6}$$

It is worth mentioning that it is impossible to extend a zero vector to an element of the set $\mathsf{B}(2n-1, n)$. For any $\pi \xleftarrow{\$} \mathsf{S}_n$, the following equivalence holds.

$$\mathbf{x} \in \{0,1\}^n \setminus \{\mathbf{0}^n\} \iff \mathbf{x}^* \in \mathsf{B}(2n-1, n) \iff \pi(\mathbf{x}^*) \in \mathsf{B}(2n-1, n). \tag{7}$$

Techniques for handling regular words. To prove knowledge of a vector $\mathbf{z} \in \{0,1\}^{2^c \cdot \frac{n}{c}}$ such that there exists $\mathbf{x} \in \{0,1\}^n$ and $\mathbf{z} = \mathsf{RE}(\mathbf{x}) \in \mathsf{Regular}(n, c)$, the authors in [53] introduces the following permutations.

- For every $\mathbf{b} = [\, b_1 \,|\, \cdots \,|\, b_c \,] \in \{0,1\}^c$, define the permutation $E_{\mathbf{b}}$ that transforms vector $\mathbf{z} = [\, z_{0,0,\ldots,0} \,|\, \cdots \,|\, z_{i_1,\ldots,i_c} \,|\, \cdots \,|\, z_{1,1,\ldots,1} \,] \in \{0,1\}^{2^c}$ into vector $E_{\mathbf{b}}(\mathbf{z}) = [\, z'_{0,0,\ldots,0} \,|\, \cdots \,|\, z'_{1,1,\ldots,1} \,]$, where for each $(i_1, \ldots, i_c) \in \{0,1\}^c$, we have $z_{i_1,\ldots,i_c} = z'_{i_1 \oplus b_1, \ldots, i_c \oplus b_c}$. It is verifiable that for any $\mathbf{x}, \mathbf{b} \in \{0,1\}^c$, we have:

$$\mathbf{z} = \Delta_{2^c}(\mathbf{x}) \iff E_{\mathbf{b}}(\mathbf{z}) = \Delta_{2^c}(\mathbf{x} \oplus \mathbf{b}). \tag{8}$$

- For $\mathbf{b} = (\mathbf{b}_1 \| \cdots \| \mathbf{b}_{\frac{n}{c}}) \in \{0,1\}^n$ consisting of $\frac{n}{c}$ blocks of length c, define the permutation $E'_{\mathbf{b}}$ that transforms vector $\mathbf{z} = (\mathbf{z}_1 \| \cdots \| \mathbf{z}_{\frac{n}{c}}) \in \{0,1\}^{2^c \cdot \frac{n}{c}}$ consisting of $\frac{n}{c}$ blocks of length 2^c into vector of the following form

$$E'_{\mathbf{b}}(\mathbf{z}) = \big(E_{\mathbf{b}_1}(\mathbf{z}_1) \,\|\, \cdots \,\|\, E_{\mathbf{b}_{\frac{n}{c}}}(\mathbf{z}_{\frac{n}{c}})\big).$$

  For any $\mathbf{x}, \mathbf{b} \in \{0,1\}^n$, it follows immediately from (8), the following equivalence holds.

$$\mathbf{z} = \mathsf{RE}(\mathbf{x}) \iff E'_{\mathbf{b}}(\mathbf{z}) = \mathsf{RE}(\mathbf{x} \oplus \mathbf{b}). \tag{9}$$

The above equivalence (9) is essential in the Stern's framework for proving knowledge of a regular word. The prover samples a uniformly random $\mathbf{b}$ and only needs to demonstrate to the verifier that the right-hand side of (9) holds. Therefore, the verifier is convinced that the left-hand side of the equivalence also holds without learning any information about $\mathbf{x}$ due to the "one-time pad" $\mathbf{b}$.

3.6 Code-Based Zero-Knowledge Arguments of Set Membership

In this section, we recall Nguyen et al.'s zero-knowledge argument of set membership [53] based on the Merkle-tree accumulator described in Section 3.1. Essentially, the goal of the prover $\mathcal{P}$ is to convince the verifier $\mathcal{V}$ that he knows a

and demonstrating to the verifier that $\pi(\mathbf{x}^*) \in \mathsf{B}(2n, n)$ for $\pi \xleftarrow{\$} \mathsf{S}_{2n}$. This technique, however, incurs permutation size $2n \cdot \log 2n$ while the technique in [38] just needs permutation size n.

value that is accumulated in the root of the Merkle tree. This protocol is very useful for designing privacy-enhancing protocols, and has been utilized in [53] to construct group signatures [22] and ring signatures [56]. In this work, we follow this line and use it as a sub-protocol in Section 7.2, in which we describe the supporting ZKAoK of our group encryption scheme. The protocol is summarized as follows.

Let n be an integer and $m = 2 \cdot 2^c \cdot \frac{n}{c}$. Given a uniformly random matrix $\mathbf{B} \in \mathbb{Z}_2^{n \times m}$ and the accumulated value $\mathbf{u} \in \mathbb{Z}_2^n$, the goal of $\mathcal{P}$ is to demonstrate to $\mathcal{V}$ the knowledge of a value $\mathbf{d} \in \mathbb{Z}_2^n$ and a valid witness $w \in \{0,1\}^\ell \times (\{0,1\}^n)^\ell$. The associated relation is as follows:

$$R_{\text{acc}} = \Big\{ \Big((\mathbf{B}, \mathbf{u}); (\mathbf{d}, w) \Big) \in \Big(\mathbb{Z}_2^{n \times m} \times \mathbb{Z}_2^n \Big) \times \Big(\mathbb{Z}_2^n \times \{0,1\}^\ell \times (\{0,1\}^n)^\ell \Big) : \mathsf{Verify}_{\mathbf{B}}(\mathbf{u}, \mathbf{d}, w) = 1 \Big\}.$$

We first recall the techniques that are developed from [38] and previous permutation techniques for handling regular words, which are essential for the ZKAoK for the above relation.

- For $d \in \{0,1\}$ and $\mathbf{x} \in \{0,1\}^{\frac{m}{2}}$, denote $\mathsf{Ext}(d, \mathbf{x})$ as $\begin{pmatrix} \overline{d} \cdot \mathbf{x} \\ d \cdot \mathbf{x} \end{pmatrix}$.
- For $e \in \{0,1\}$, for $\mathbf{b} \in \{0,1\}^n$, define the permutation $\Phi_{e,\mathbf{b}}$ that acts on $\mathbf{z} = \begin{pmatrix} \mathbf{z}_0 \\ \mathbf{z}_1 \end{pmatrix} \in \{0,1\}^m$, where $\mathbf{z}_0, \mathbf{z}_1 \in \{0,1\}^{\frac{m}{2}}$, as follows. It transforms $\mathbf{z}$ to $\Phi_{e,\mathbf{b}}(\mathbf{z}) = \begin{pmatrix} E'_{\mathbf{b}}(\mathbf{z}_e) \\ E'_{\mathbf{b}}(\mathbf{z}_{\overline{e}}) \end{pmatrix}$. Namely, it rearranges the blocks of $\mathbf{z}$ according to e and permutes each block using $E'_{\mathbf{b}}$.
- For any $d, e \in \{0,1\}$ and $\mathbf{v}, \mathbf{w}, \mathbf{b}, \mathbf{c} \in \{0,1\}^n$, it follows from (9) that the following equivalences hold:

$$\begin{cases} \mathbf{y} = \mathsf{Ext}(d, \mathsf{RE}(\mathbf{v})) & \iff \Phi_{e,\mathbf{b}}(\mathbf{y}) = \mathsf{Ext}(d \oplus e, \mathsf{RE}(\mathbf{v} \oplus \mathbf{b})) \\ \mathbf{z} = \mathsf{Ext}(\overline{d}, \mathsf{RE}(\mathbf{w})) & \iff \Phi_{\overline{e},\mathbf{c}}(\mathbf{z}) = \mathsf{Ext}(\overline{d} \oplus \overline{e}, \mathsf{RE}(\mathbf{w} \oplus \mathbf{c})). \end{cases} \tag{10}$$

Now let us look at the equations pertaining to the relation R_{acc}. Write $\mathbf{B} = [\mathbf{B}_0 | \mathbf{B}_1]$. Recall that w is of the form $([j_1 | j_2 | \cdots | j_\ell]^\top, \mathbf{w}_\ell, \ldots, \mathbf{w}_1)$ and that Verify algorithm computes $\mathbf{v}_\ell = \mathbf{d}, \mathbf{v}_{\ell-1}, \ldots, \mathbf{v}_1, \mathbf{v}_0 = \mathbf{u}$, where $\mathbf{v}_i$ for $i \in \{\ell-1, \ldots, 1, 0\}$ is computed as follows:

$$\mathbf{v}_i = \begin{cases} \mathbf{B}_0 \cdot \mathsf{RE}(\mathbf{v}_{i+1}) \oplus \mathbf{B}_1 \cdot \mathsf{RE}(\mathbf{w}_{i+1}) \text{ if } j_{i+1} = 0; \\ \mathbf{B}_0 \cdot \mathsf{RE}(\mathbf{w}_{i+1}) \oplus \mathbf{B}_1 \cdot \mathsf{RE}(\mathbf{v}_{i+1}) \text{ if } j_{i+1} = 1. \end{cases} \tag{11}$$

The equation (11) is equivalent to the following form.

$$\begin{aligned} \mathbf{v}_i &= \overline{j_{i+1}} \cdot \big(\mathbf{B}_0 \cdot \mathsf{RE}(\mathbf{v}_{i+1}) \oplus \mathbf{B}_1 \cdot \mathsf{RE}(\mathbf{w}_{i+1}) \big) \oplus j_{i+1} \cdot \big(\mathbf{B}_0 \cdot \mathsf{RE}(\mathbf{w}_{i+1}) \oplus \mathbf{B}_1 \cdot \mathsf{RE}(\mathbf{v}_{i+1}) \big) \\ &= \mathbf{B} \begin{pmatrix} \overline{j_{i+1}} \cdot \mathsf{RE}(\mathbf{v}_{i+1}) \\ j_{i+1} \cdot \mathsf{RE}(\mathbf{v}_{i+1}) \end{pmatrix} \oplus \mathbf{B} \begin{pmatrix} j_{i+1} \cdot \mathsf{RE}(\mathbf{w}_{i+1}) \\ \overline{j_{i+1}} \cdot \mathsf{RE}(\mathbf{w}_{i+1}) \end{pmatrix} \\ &= \mathbf{B} \cdot \mathsf{Ext}(j_{i+1}, \mathsf{RE}(\mathbf{v}_{i+1})) \oplus \mathbf{B} \cdot \mathsf{Ext}(\overline{j_{i+1}}, \mathsf{RE}(\mathbf{w}_{i+1})) \end{aligned}$$

Therefore, the goal of $\mathcal{P}$ is now equivalent to prove knowledge of $[j_1|j_2,\ldots|j_\ell]^\top$,$\mathbf{v}_1$, $\ldots$, $\mathbf{v}_\ell$, $\mathbf{w}_1,\ldots,\mathbf{w}_\ell$ such that the following hold.

$$\begin{cases} \mathbf{B}\cdot\mathsf{Ext}(j_1,\mathsf{RE}(\mathbf{v}_1))\oplus\mathbf{B}\cdot\mathsf{Ext}(\overline{j_1},\mathsf{RE}(\mathbf{w}_1))=\mathbf{u};\\ \mathbf{B}\cdot\mathsf{Ext}(j_2,\mathsf{RE}(\mathbf{v}_2))\oplus\mathbf{B}\cdot\mathsf{Ext}(\overline{j_2},\mathsf{RE}(\mathbf{w}_2))\oplus\mathbf{v}_1=\mathbf{0};\\ \cdots\cdots\\ \mathbf{B}\cdot\mathsf{Ext}(j_\ell,\mathsf{RE}(\mathbf{v}_\ell))\oplus\mathbf{B}\cdot\mathsf{Ext}(\overline{j_\ell},\mathsf{RE}(\mathbf{w}_\ell))\oplus\mathbf{v}_{\ell-1}=\mathbf{0}. \end{cases} \tag{12}$$

We aim to transform the above equations (12) into an instance of the abstract form described in Section 3.4 such that the equivalences in (2) hold. To this end, we apply the function Encode to $\mathbf{v}_i$ for $i\in[\ell-1]$. Let $\mathbf{x}_i=\mathsf{Encode}(\mathbf{v}_i)\in\{0,1\}^{2n}$ for $i\in[\ell-1]$. We therefore obtain $\mathbf{v}_i=\mathbf{I}_n^*\cdot\mathbf{x}_i$. To ease the notation, for $i\in[\ell]$, denote

$$\begin{cases} \mathbf{y}_i=\mathsf{Ext}(j_i,\mathsf{RE}(\mathbf{v}_i))\in\{0,1\}^m;\\ \mathbf{z}_i=\mathsf{Ext}(\overline{j_i},\mathsf{RE}(\mathbf{w}_i))\in\{0,1\}^m. \end{cases} \tag{13}$$

Now equations in (12) are equivalent to

$$\begin{cases} \mathbf{B}\cdot\mathbf{y}_1\oplus\mathbf{B}\cdot\mathbf{z}_1=\mathbf{u};\\ \mathbf{B}\cdot\mathbf{y}_2\oplus\mathbf{B}\cdot\mathbf{z}_2\oplus\mathbf{I}_n^*\cdot\mathbf{x}_1=\mathbf{0};\\ \cdots\cdots\\ \mathbf{B}\cdot\mathbf{y}_\ell\oplus\mathbf{B}\cdot\mathbf{z}_\ell\oplus\mathbf{I}_n^*\cdot\mathbf{x}_{\ell-1}=\mathbf{0}. \end{cases} \tag{14}$$

Through some basic algebra, we can transform the equations in (14) to a unifying equation of the form $\mathbf{M}_{\mathrm{acc}}\cdot\mathbf{w}_{\mathrm{acc}}=\mathbf{v}_{\mathrm{acc}}$, where $\mathbf{M}_{\mathrm{acc}}\in\mathbb{Z}_2^{\ell n\times L_{\mathrm{acc}}}$ and $\mathbf{v}_{\mathrm{acc}}\in\mathbb{Z}_2^{\ell n}$ are public and $\mathbf{w}_{\mathrm{acc}}\in\{0,1\}^{L_{\mathrm{acc}}}$ is secret with $L_{\mathrm{acc}}=2\ell m+2(\ell-1)n$ and

$$\mathbf{w}_{\mathrm{acc}}=(\,\mathbf{y}_1\parallel\cdots\parallel\mathbf{y}_\ell\parallel\mathbf{z}_1\parallel\cdots\parallel\mathbf{z}_\ell\parallel\mathbf{x}_1\parallel\cdots\parallel\mathbf{x}_{\ell-1}\,). \tag{15}$$

Now we specify the set $\mathsf{VALID}_{\mathrm{acc}}$ that contains of secret vector $\mathbf{w}_{\mathrm{acc}}$, the set $\mathcal{S}_{\mathrm{acc}}$, and permutations $\{\Gamma_\phi:\phi\in\mathcal{S}_{\mathrm{acc}}\}$ such that the equivalences in (2) hold. Let $\mathsf{VALID}_{\mathrm{acc}}$ contain all vectors of the form

$$\widehat{\mathbf{w}}_{\mathrm{acc}}=(\,\widehat{\mathbf{y}}_1\parallel\cdots\parallel\widehat{\mathbf{y}}_\ell\parallel\widehat{\mathbf{z}}_1\parallel\cdots\parallel\widehat{\mathbf{z}}_\ell\parallel\widehat{\mathbf{x}}_1\parallel\cdots\parallel\widehat{\mathbf{x}}_{\ell-1}\,)$$

satisfying the following constraints:

- For $i\in[\ell]$, there exists $\widehat{j_i}\in\{0,1\}$, $\widehat{\mathbf{v}}_i,\widehat{\mathbf{w}}_i\in\{0,1\}^n$ such that

$$\widehat{\mathbf{y}}_i=\mathsf{Ext}(\widehat{j_i},\mathsf{RE}(\widehat{\mathbf{v}}_i))\in\{0,1\}^m;\ \text{ and }\ \widehat{\mathbf{z}}_i=\mathsf{Ext}(\overline{\widehat{j_i}},\mathsf{RE}(\widehat{\mathbf{w}}_i))\in\{0,1\}^m.$$

- For $i\in[\ell-1]$, $\widehat{\mathbf{x}}_i=\mathsf{Encode}(\widehat{\mathbf{v}}_i)\in\{0,1\}^{2n}$.

Let $\mathcal{S}_{\mathrm{acc}}=(\{0,1\})^\ell\times(\{0,1\}^n)^\ell\times(\{0,1\}^n)^\ell$. Then for each

$$\phi=(\,g_1,\ldots,g_\ell,\,\mathbf{b}_1,\ldots,\mathbf{b}_\ell,\,\mathbf{c}_1,\ldots,\mathbf{c}_\ell\,)\in\mathcal{S}_{\mathrm{acc}},$$

define the permutation Γ_ϕ that transforms vector of the form

$$\widehat{\mathbf{w}}_{\text{acc}} = (\, \widehat{\mathbf{y}}_1 \,\|\, \cdots \,\|\, \widehat{\mathbf{y}}_\ell \,\|\, \widehat{\mathbf{z}}_1 \,\|\, \cdots \,\|\, \widehat{\mathbf{z}}_\ell \,\|\, \widehat{\mathbf{x}}_1 \,\|\, \cdots \,\|\, \widehat{\mathbf{x}}_{\ell-1} \,)$$

with $\widehat{\mathbf{y}}_i, \widehat{\mathbf{z}}_i \in \{0,1\}^m$ for $i \in [\ell]$ and $\widehat{\mathbf{x}}_i \in \{0,1\}^{2n}$ for $i \in [\ell - 1]$ to vector

$$\Gamma_\phi(\widehat{\mathbf{w}}_{\text{acc}}) = \big(\; \Phi_{g_1,\mathbf{b}_1}(\widehat{\mathbf{y}}_1) \,\|\, \cdots \,\|\, \Phi_{g_\ell,\mathbf{b}_\ell}(\widehat{\mathbf{y}}_\ell) \,\| \\ \Phi_{\overline{g_1},\mathbf{c}_1}(\widehat{\mathbf{z}}_1) \,\|\, \cdots \,\|\, \Phi_{\overline{g_\ell},\mathbf{c}_\ell}(\widehat{\mathbf{z}}_\ell) \,\|\, F_{\mathbf{b}_1}(\widehat{\mathbf{x}}_1) \,\|\, \cdots \,\|\, F_{\mathbf{b}_{\ell-1}}(\widehat{\mathbf{x}}_{\ell-1}) \,\big).$$

Based on the equivalence observed in (10) and (5), it can be checked that $\mathbf{w}^*_{\text{acc}} \in \mathsf{VALID}_{\text{acc}}$ is equivalent to $\Gamma_\phi(\widehat{\mathbf{w}}_{\text{acc}}) \in \mathsf{VALID}_{\text{acc}}$. Furthermore, if ϕ is randomly chosen from $\mathcal{S}_{\text{acc}}$, then $\Gamma_\phi(\widehat{\mathbf{w}}_{\text{acc}})$ is randomly distributed in $\mathsf{VALID}_{\text{acc}}$. In other words, we have successfully reduced the considered relation to an instance of R_{abstract}. Therefore, $\mathcal{P}$ and $\mathcal{V}$ interact as described in Figure 2. The resulting protocol is a statistical ZKAoK with perfect completeness, soundness error 2/3, and communication cost $\mathcal{O}(L_{\text{acc}}) = \mathcal{O}(\ell \cdot (m+n)) = \mathcal{O}(\lambda \cdot \log N)$ bits.

4 Zero-Knowledge Arguments for Quadratic Relations

In this section, we present our ZKAoK for quadratic relations. More concretely, our arguments demonstrate that a given value $\mathbf{c}$ is an honest evaluation of the form $\mathbf{A} \cdot \mathbf{r} \oplus \mathbf{e}$, where $\mathbf{A}, \mathbf{r}, \mathbf{e}$ are all secret and may satisfy other constraints. In Section 4.1, we present our ZKAoK for LPN relation with hidden $\mathbf{A} \in \mathbb{Z}_2^{n \times m}, \mathbf{r} \in \mathbb{Z}_2^m, \mathbf{e} \in \mathsf{B}(n,t)$. In Section 4.2 we then present our ZKAoK for a variant of LPN relation, where we consider the vector $\mathbf{r}$ with fixed hamming weight instead. We remark that while the first ZKAoK can be easily developed from previous works [37,43], where the authors proved quadratic statements about Learning With Errors (LWE) relation [55] and Ring Learning With Errors (RLWE) relation [46] respectively, the second one is a novel contribution of this work. These two protocols will be crucial sub-protocols of our main ZKAoK in Section 7.2.

4.1 Proving the LPN Relation with Hidden Matrix

We now present our ZKAoK for the LPN relation with hidden matrix. Let n, m, t be positive integers, $\mathbf{A} \in \mathbb{Z}_2^{n \times m}$, $\mathbf{r} \in \mathbb{Z}_2^m$, $\mathbf{e} \in \mathsf{B}(n,t)$, $\mathbf{c} \in \mathbb{Z}_2^n$. The target of $\mathcal{P}$ is to demonstrate the knowledge of $\mathbf{A}, \mathbf{r}, \mathbf{e}$ such that $\mathbf{c} = \mathbf{A} \cdot \mathbf{r} \oplus \mathbf{e}$. The associated relation is defined as follows:

$$R_{\text{LPN}} = \Big\{ \Big(\mathbf{c}; (\mathbf{A}, \mathbf{r}, \mathbf{e}) \Big) \in \mathbb{Z}_2^n \times \Big(\mathbb{Z}_2^{n \times m} \times \mathbb{Z}_2^m \times \mathsf{B}(n,t) \Big) : \mathbf{c} = \mathbf{A} \cdot \mathbf{r} \oplus \mathbf{e} \Big\}. \quad (16)$$

Before we present our ZKAoK, let us first introduce the double-bit extension ext and matrix-vector product expansion expand together with their corresponding permutation techniques that are directly adapted from [37,43].

Double-bit extension. To prove knowledge of a bit c such that c is a product of two secret bits a, r, define the following notations and techniques.

– For any two bits a, r, define extension of $c = a \cdot r$ to be the vector

$$\mathsf{ext}(c) \stackrel{\triangle}{=} \mathsf{ext}(a, r) = [\, \overline{a} \cdot \overline{r} |\, \overline{a} \cdot r |\, a \cdot \overline{r} |\, a \cdot r \,]^\top \in \{0,1\}^4.$$

– Define $\mathbf{h} = [0|0|0|1]$, then it is easy to see that $c = \mathbf{h} \cdot \mathsf{ext}(c)$.

– For any two bits $b, s \in \{0,1\}$, define $T_{b,s}$ as the permutation that transforms vector $\mathbf{z} = [\, z_{0,0} \,|\, z_{0,1} \,|\, z_{1,0} \,|\, z_{1,1} \,]^\top \in \{0,1\}^4$ into vector

$$T_{b,s}(\mathbf{z}) = [\, z_{b,s} \,|\, z_{b,\overline{s}} \,|\, z_{\overline{b},s} \,|\, z_{\overline{b},\overline{s}} \,]^\top \in \{0,1\}^4.$$

Note that for any bits a, r, b, s, we have

$$\mathbf{z} = \mathsf{ext}(a, r) \iff T_{b,s}(\mathbf{z}) = \mathsf{ext}(a \oplus b, r \oplus s). \tag{17}$$

Note that the above equivalence (17) is useful in Stern's framework. To prove the well-formedness of c that is a product of a, r, the prover extends c to $\mathbf{z} = \mathsf{ext}(a, r)$, and permutes $\mathbf{z}$ using two randomness b, s. Then it demonstrates to the verifier that the permuted vector $T_{b,s}(\mathbf{z})$ is well-formed. Due to the above equivalence, the verifier is convinced that c is well-formed as well. In addition, the "one-time pads" b, s hide the information of a, r. We remark that this technique is only useful when a, r satisfy other constraints, since otherwise we could simply prove knowledge of a bit c using previously described Encode and the associated permutation.

Matrix-vector product expansion. Let n, m be positive integers. Denote vector $\mathbf{a} = [\, a_{1,1} \,|\, \cdots \,|\, a_{1,n} \,|\, a_{2,1} \,|\, \cdots \,|\, a_{2,n} \,|\, \cdots \,|\, a_{m,1} \,|\, \cdots \,|\, a_{m,n} \,]^\top \in \{0,1\}^{mn}$ and vector $\mathbf{r} = [\, r_1 \,|\, \cdots \,|\, r_m \,]^\top \in \{0,1\}^m$. We now present the techniques to show the well-formedness of $\mathbf{c} \in \{0,1\}^{mn}$ such that $\mathbf{c}$ is of the form

$$\mathbf{c} = [\, c_{1,1} \,|\, \cdots \,|\, c_{1,n} \,|\, c_{2,1} \,|\, \cdots \,|\, c_{2,n} \,|\, \cdots \,|\, c_{m,1} \,|\, \cdots \,|\, c_{m,n} \,]^\top$$

with $c_{i,j} = a_{i,j} \cdot r_i$ for $i \in [m], j \in [n]$. Define the extension of $\mathbf{c}$ to be a vector of the following form:

$$\begin{aligned}\mathsf{expand}(\mathbf{a}, \mathbf{r}) = \big(\mathsf{ext}(c_{1,1}) \,\|\, \cdots \,\|\; & \mathsf{ext}(c_{1,n}) \,\|\, \mathsf{ext}(c_{2,1}) \,\|\, \cdots \,\|\, \mathsf{ext}(c_{2,n}) \,\|\, \cdots \,\| \\ & \mathsf{ext}(c_{m,1}) \,\|\, \cdots \,\|\, \mathsf{ext}(c_{m,n}) \big) \in \{0,1\}^{4mn}.\end{aligned}$$

Now for $\mathbf{b} = [\, b_{1,1} \,|\, \cdots \,|\, b_{1,n} \,|\, b_{2,1} \,|\, \cdots \,|\, b_{2,n} \,|\, \cdots \,|\, b_{m,1} \,|\, \cdots \,|\, b_{m,n} \,]^\top \in \{0,1\}^{mn}$ and $\mathbf{s} = [\, s_1 \,|\, \cdots \,|\, s_m \,]^\top \in \{0,1\}^m$, we define $T'_{\mathbf{b},\mathbf{s}}$ that transforms vector $\mathbf{z} \in \{0,1\}^{4mn}$ of the following form

$$\mathbf{z} = (\, \mathbf{z}_{1,1} \,\|\, \cdots \,\|\, \mathbf{z}_{1,n} \,\|\, \mathbf{z}_{2,1} \,\|\, \cdots \,\|\, \mathbf{z}_{2,n} \,\|\, \cdots \,\|\, \mathbf{z}_{m,1} \,\|\, \cdots \,\|\, \mathbf{z}_{m,n} \,),$$

which consisting of mn blocks of length 4, to vector $T'_{\mathbf{b},\mathbf{s}}$ of the following form

$$\begin{aligned}T'_{\mathbf{b},\mathbf{s}}(\mathbf{z}) = (\; & T_{b_{1,1},s_1}(\mathbf{z}_{1,1}) \,\|\, \cdots \,\|\, T_{b_{1,n},s_1}(\mathbf{z}_{1,n}) \,\|\, T_{b_{2,1},s_2}(\mathbf{z}_{2,1}) \,\|\, \cdots \,\| \\ & T_{b_{2,n},s_2}(\mathbf{z}_{2,n}) \,\|\, \cdots \,\|\, T_{b_{m,1},s_m}(\mathbf{z}_{m,1}) \,\|\, \cdots \,\|\, T_{b_{m,n},s_m}(\mathbf{z}_{m,n}) \,).\end{aligned}$$

It then follows from (17) that the following equivalence holds.

$$\mathbf{z} = \mathsf{expand}(\mathbf{a}, \mathbf{r}) \iff T'_{\mathbf{b},\mathbf{s}}(\mathbf{z}) = \mathsf{expand}(\mathbf{a} \oplus \mathbf{b}, \mathbf{r} \oplus \mathbf{s}). \tag{18}$$

The Zero-knowledge argument. We aim to transform the relation R_{LPN} to an instance of R_{abstract} such that the equivalences in (2) hold. Towards this goal, we proceed as follows.

Write $\mathbf{A} = [\,\mathbf{a}_1 \,|\, \cdots \,|\, \mathbf{a}_m\,] \in \mathbb{Z}_2^{n\times m}$ and $\mathbf{r} = [\,r_1 \,|\, \cdots \,|\, r_m\,]^\top \in \mathbb{Z}_2^m$, then we have

$$\begin{aligned}
\mathbf{A}\cdot\mathbf{r} &= \sum_{i=1}^{m} \mathbf{a}_i \cdot r_i = \sum_{i=1}^{m} [\,a_{i,1}\cdot r_i \,|\, a_{i,2}\cdot r_i \,|\, \cdots \,|\, a_{i,n}\cdot r_i\,]^\top \\
&= \sum_{i=1}^{m} \big[\,\mathbf{h}\cdot\mathsf{ext}(a_{i,1}, r_i) \,|\, \mathbf{h}\cdot\mathsf{ext}(a_{i,2}, r_i) \,|\, \cdots \,|\, \mathbf{h}\cdot\mathsf{ext}(a_{i,n}, r_i)\,\big]^\top \\
&= \sum_{i=1}^{m} \mathbf{H}_{n,1}\big(\,\mathsf{ext}(a_{i,1}, r_i) \,\|\, \mathsf{ext}(a_{i,2}, r_i) \,\|\, \cdots \,\|\, \mathsf{ext}(a_{i,n}, r_i)\,\big) \\
&= \sum_{i=1}^{m} \mathbf{H}_{n,1}\cdot\mathbf{z}_i \\
&= \underbrace{[\mathbf{H}_{n,1}|\cdots|\mathbf{H}_{n,1}]}_{m \text{ times}} \cdot (\,\mathbf{z}_1 \,\|\, \cdots \,\|\, \mathbf{z}_m\,),
\end{aligned}$$

where $\mathbf{H}_{n,1} = \begin{pmatrix} \mathbf{h} & & \\ & \mathbf{h} & \\ & & \ddots \\ & & & \mathbf{h} \end{pmatrix} \in \mathbb{Z}_2^{n\times 4n}$ and $\mathbf{z}_i = (\mathsf{ext}(a_{i,1}, r_i)\|\cdots\|\mathsf{ext}(a_{i,n}, r_i)) \in \mathbb{Z}_2^{4n}$. Denote $\mathbf{H}_{n,m} = \underbrace{[\mathbf{H}_{n,1}|\cdots|\mathbf{H}_{n,1}]}_{m \text{ times}} \in \mathbb{Z}_2^{n\times 4mn}$, $\mathbf{z} = (\mathbf{z}_1\|\cdots\|\mathbf{z}_m) \in \mathbb{Z}_2^{4mn}$, and $\mathbf{a} = [a_{1,1}|\cdots|a_{1,n}|a_{2,1}|\cdots|a_{2,n}|\cdots|a_{m,1}|\cdots|a_{m,n}]^\top \in \mathbb{Z}_2^{mn}$. Then $\mathbf{z}$ is indeed the expansion vector of $\mathbf{a}$ and $\mathbf{r}$, i.e., $\mathbf{z} = \mathsf{expand}(\mathbf{a}, \mathbf{r})$. If no ambiguity cause, we write $\mathbf{z} = \mathsf{expand}(\mathbf{A}, \mathbf{r})$. Hence, we obtain the following:

$$\mathbf{c} = \mathbf{A}\cdot\mathbf{r} \oplus \mathbf{e} \Longleftrightarrow \mathbf{c} = \mathbf{H}_{n,m}\cdot\mathsf{expand}(\mathbf{A}, \mathbf{r}) \oplus \mathbf{e}. \tag{19}$$

Denote $\mathbf{M}_{\mathrm{LPN}} = [\mathbf{H}_{n,m}|\mathbf{I}_n] \in \mathbb{Z}_2^{n\times L_{\mathrm{LPN}}}$ and $\mathbf{w}_{\mathrm{LPN}} = (\mathsf{expand}(\mathbf{A}, \mathbf{r})\|\mathbf{e}) \in \mathbb{Z}_2^{L_{\mathrm{LPN}}}$ with $L_{\mathrm{LPN}} = 4mn + n$. We then have $\mathbf{c} \stackrel{\triangle}{=} \mathbf{v}_{\mathrm{LPN}} = \mathbf{M}_{\mathrm{LPN}}\cdot\mathbf{w}_{\mathrm{LPN}} \bmod 2$.

Now we are ready to specify the set $\mathsf{VALID}_{\mathrm{LPN}}$ that contains of secret vector $\mathbf{w}_{\mathrm{LPN}}$, the set $\mathcal{S}_{\mathrm{LPN}}$, and permutations $\{\Gamma_\phi : \phi \in \mathcal{S}_{\mathrm{LPN}}\}$ such that the equivalences in (2) hold. To this end, let $\mathsf{VALID}_{\mathrm{LPN}}$ contain all vectors $\widehat{\mathbf{w}}_{\mathrm{LPN}} = (\widehat{\mathbf{z}}\|\widehat{\mathbf{e}}) \in \mathbb{Z}_2^{4mn+n}$ satisfying the following constraints:

- There exists $\widehat{\mathbf{a}} \in \mathbb{Z}_2^{nm}$ and $\widehat{\mathbf{r}} \in \mathbb{Z}_2^m$ such that $\widehat{\mathbf{z}} = \mathsf{expand}(\widehat{\mathbf{a}}, \widehat{\mathbf{r}})$.
- $\widehat{\mathbf{e}} \in \mathsf{B}(n, t)$.

It is easy to verify that $\mathbf{w}_{\mathrm{LPN}} \in \mathsf{VALID}_{\mathrm{LPN}}$. Let $\mathcal{S}_{\mathrm{LPN}} = \{0,1\}^{mn} \times \{0,1\}^m \times \mathsf{S}_n$. Then for each $\phi = (\mathbf{b}, \mathbf{s}, \sigma_e) \in \mathcal{S}_{\mathrm{LPN}}$, define the permutation Γ_ϕ that transforms vector of the form $\widehat{\mathbf{w}}_{\mathrm{LPN}} = (\,\widehat{\mathbf{z}} \,\|\, \widehat{\mathbf{e}}\,)$ with $\widehat{\mathbf{z}} \in \mathbb{Z}_2^{4mn}, \widehat{\mathbf{e}} \in \mathbb{Z}_2^n$ to vector $\Gamma_\phi(\widehat{\mathbf{w}}_{\mathrm{LPN}}) = (\,T'_{\mathbf{b},\mathbf{s}}(\widehat{\mathbf{z}}) \,\|\, \sigma_e(\widehat{\mathbf{e}})\,)$.

Based on the equivalence observed in (18) and (3), it can be checked that $\widehat{\mathbf{w}}_{\mathrm{LPN}} \in \mathsf{VALID}_{\mathrm{LPN}}$ is equivalent to $\Gamma_\phi(\widehat{\mathbf{w}}_{\mathrm{LPN}}) \in \mathsf{VALID}_{\mathrm{LPN}}$. Furthermore, if ϕ is randomly chosen from $\mathcal{S}_{\mathrm{LPN}}$, then $\Gamma_\phi(\widehat{\mathbf{w}}_{\mathrm{LPN}})$ is randomly distributed in $\mathsf{VALID}_{\mathrm{LPN}}$. In other words, we have successfully reduced the considered relation to an instance of R_{abstract}. Therefore, $\mathcal{P}$ and $\mathcal{V}$ interact as described in Figure 2. The resulting protocol is a statistical ZKAoK with perfect completeness, soundness error 2/3, and communication cost $\mathcal{O}(L_{\mathrm{LPN}}) = \mathcal{O}(mn) = \mathcal{O}(\lambda^2)$ bits.

4.2 Proving A Variant of LPN Relation with Hidden Matrix

Let n, m, t, t_r be positive integers, $\mathbf{A} \in \mathbb{Z}_2^{n\times m}$, $\mathbf{r} \in \mathsf{B}(m, t_r)$, $\mathbf{e} \in \mathsf{B}(n,t), \mathbf{c} \in \mathbb{Z}_2^n$. We now present our ZKAoK that allows $\mathcal{P}$ to prove its knowledge of $\mathbf{A}, \mathbf{r}, \mathbf{e}$ such that $\mathbf{c} = \mathbf{A} \cdot \mathbf{r} \oplus \mathbf{e}$. The associated relation is defined as follows:

$$R_{\mathrm{VLPN}} = \Big\{ \Big(\mathbf{c}; (\mathbf{A}, \mathbf{r}, \mathbf{e}) \Big) \in \mathbb{Z}_2^n \times \Big(\mathbb{Z}_2^{n\times m} \times \mathsf{B}(m, t_r) \times \mathsf{B}(n,t) \Big) : \mathbf{c} = \mathbf{A} \cdot \mathbf{r} \oplus \mathbf{e} \Big\}.$$

To prove $\mathbf{r}$ has fixed hamming, we introduce the following Hadamard product extension and extended matrix-vector product expansion and their corresponding permutations.

Hadamard product extension. Let vectors $\mathbf{a} \in \{0,1\}^m$, $\mathbf{r} \in \mathsf{B}(m, t_r)$ and $\mathbf{c} = [a_1 \cdot r_1 | a_2 \cdot r_2 | \cdots | a_m \cdot r_m]^\top$. The goal is to prove the well-formedness of $\mathbf{c}$, i.e., $\mathbf{c}$ is a Hadamard product of two binary vectors, one of which has fixed hamming weight t_r. We therefore introduce the following extension and permutation.

- Define extension of $c_i = a_i \cdot r_i$ as $\mathsf{ext}'(c_i) \stackrel{\triangle}{=} \mathsf{ext}'(a_i, r_i) = [\overline{a_i} \cdot r_i | a_i \cdot r_i]^\top \in \{0,1\}^2$. Let $\mathbf{h}' = [0|1]$, then we obtain $c_i = \mathbf{h}' \cdot \mathsf{ext}'(c_i)$.
- Define the extension of $\mathbf{c}$ to be vector of the form

  $$\mathsf{ext}'(\mathbf{a}, \mathbf{r}) = [\,\overline{a_1} \cdot r_1 \,|\, a_1 \cdot r_1 \,|\, \overline{a_2} \cdot r_2 \,|\, a_2 \cdot r_2 \,|\, \cdots \,|\, \overline{a_m} \cdot r_m \,|\, a_m \cdot r_m\,]^\top \in \{0,1\}^{2m}.$$

- For any $\mathbf{b} = [b_1|b_2|\cdots|b_m]^\top \in \{0,1\}^m$, $\sigma \in \mathsf{S}_m$, define permutation $\Psi_{\mathbf{b},\sigma}$ that transforms a vector

  $$\mathbf{z} = [\, z_1^{(0)} \mid z_1^{(1)} \mid z_2^{(0)} \mid z_2^{(1)} \mid \cdots \mid z_m^{(0)} \mid z_m^{(1)} \,]^\top \in \mathbb{Z}^{2m}$$

  to a vector

  $$\Psi_{\mathbf{b},\sigma}(\mathbf{z}) = [\, z_{\sigma(1)}^{(b_{\sigma(1)})} \mid z_{\sigma(1)}^{(\overline{b_{\sigma(1)}})} \mid z_{\sigma(2)}^{(b_{\sigma(2)})} \mid z_{\sigma(2)}^{(\overline{b_{\sigma(2)}})} \mid \cdots \mid z_{\sigma(m)}^{(b_{\sigma(m)})} \mid z_{\sigma(m)}^{(\overline{b_{\sigma(m)}})} \,]^\top.$$

- For any $\mathbf{a}, \mathbf{b} \in \{0,1\}^m, \mathbf{r} \in \mathsf{B}(m, t_r), \sigma \in \mathsf{S}_m$, it is verifiable that the following equivalence holds.

  $$\mathbf{z} = \mathsf{ext}'(\mathbf{a}, \mathbf{r}) \Longleftrightarrow \Psi_{\mathbf{b},\sigma}(\mathbf{z}) = \mathsf{ext}'\big(\sigma(\mathbf{a} \oplus \mathbf{b}), \sigma(\mathbf{r})\big). \tag{20}$$

Example. Let $m = 4, t_r = 2$, $\mathbf{a} = [1|1|0|1]^\top$, $\mathbf{b} = [0|1|0|1]^\top$, $\mathbf{r} = [1|0|0|1]^\top$, $\sigma(i) = i + 1$ for $i \in [3]$ and $\sigma(4) = 1$. We have $\mathbf{d} = \sigma(\mathbf{a} \oplus \mathbf{b}) = [0|0|0|1]^\top$, $\mathbf{e} = \sigma(\mathbf{r}) = [0|0|1|1]^\top$, and

$$\begin{aligned}
\mathbf{z} = \mathsf{ext}'(\mathbf{a}, \mathbf{r}) &= [\, z_1^{(0)} \mid z_1^{(1)} \mid z_2^{(0)} \mid z_2^{(1)} \mid z_3^{(0)} \mid z_3^{(1)} \mid z_4^{(0)} \mid z_4^{(1)} \,]^\top \\
&= [\ 0 \ \mid \ 1 \ \mid \ 0 \ \mid \ 0 \ \mid \ 0 \ \mid \ 0 \ \mid \ 0 \ \mid \ 1 \ ]^\top; \\
\Psi_{\mathbf{b},\sigma}(\mathbf{z}) &= [\, z_2^{(b_2)} \mid z_2^{(\overline{b_2})} \mid z_3^{(b_3)} \mid z_3^{(\overline{b_3})} \mid z_4^{(b_4)} \mid z_4^{(\overline{b_4})} \mid z_1^{(b_1)} \mid z_1^{(\overline{b_1})} \,]^\top \\
&= [\, z_2^{(1)} \mid z_2^{(0)} \mid z_3^{(0)} \mid z_3^{(1)} \mid z_4^{(1)} \mid z_4^{(0)} \mid z_1^{(0)} \mid z_1^{(1)} \,]^\top \\
&= [\ 0 \ \mid \ 0 \ \mid \ 0 \ \mid \ 0 \ \mid \ 1 \ \mid \ 0 \ \mid \ 0 \ \mid \ 1 \ ]^\top; \\
\mathsf{ext}'(\mathbf{d}, \mathbf{e}) &= [\, \overline{d_1} \cdot e_1 \mid d_1 \cdot e_1 \mid \overline{d_2} \cdot e_2 \mid d_2 \cdot e_2 \mid \overline{d_3} \cdot e_3 \mid d_3 \cdot e_3 \mid \overline{d_4} \cdot e_4 \mid d_4 \cdot e_4 \,] \\
&= [\ 0 \ \mid \ 0 \ \mid \ 0 \ \mid \ 0 \ \mid \ 1 \ \mid \ 0 \ \mid \ 0 \ \mid \ 1 \ ]^\top.
\end{aligned}$$

Extended matrix-vector product expansion. Let vectors $\mathbf{a}, \mathbf{r}$ be of the form $\mathbf{a} = [a_{1,1}|\cdots|a_{1,n}|\cdots|a_{m,1}|\cdots|a_{m,n}]^\top \in \mathbb{Z}_2^{mn}$ and $\mathbf{r} = [r_1|\cdots|r_m]^\top \in \mathsf{B}(m, t_r)$, and $\mathbf{c} \in \mathbb{Z}_2^{mn}$ be of the form

$$\mathbf{c} = [a_{1,1} \cdot r_1 | \cdots | a_{1,n} \cdot r_1 | a_{2,1} \cdot r_2 | \cdots | a_{2,n} \cdot r_2 | \cdots | a_{m,1} \cdot r_m | \cdots | a_{m,n} \cdot r_m]^\top.$$

We now present the techniques to show the well-formedness of $\mathbf{c}$.

Define extension of $\mathbf{c}$ to be a vector $\mathsf{expand}'(\mathbf{a}, \mathbf{r}) \in \mathbb{Z}_2^{2mn}$ of the form:

$$\begin{aligned}
\mathsf{expand}'(\mathbf{a}, \mathbf{r}) = [\ & \overline{a_{1,1}} \cdot r_1 \,|\, a_{1,1} \cdot r_1 \,|\, \overline{a_{1,2}} \cdot r_1 \,|\, a_{1,2} \cdot r_1 \,|\, \cdots \,|\, \overline{a_{1,n}} \cdot r_1 \,|\, a_{1,n} \cdot r_1 | \\
& \overline{a_{2,1}} \cdot r_2 \,|\, a_{2,1} \cdot r_2 \,|\, \overline{a_{2,2}} \cdot r_2 \,|a_{2,2} \cdot r_2 \,|\, \cdots \,|\, \overline{a_{2,n}} \cdot r_2 \,|\, a_{2,n} \cdot r_2 \,|\, \cdots | \\
\overline{a_{m,1}} \cdot r_m \,| \ & a_{m,1} \cdot r_m \,|\, \overline{a_{m,2}} \cdot r_m \,|\, a_{m,2} \cdot r_m \,|\, \cdots \,|\, \overline{a_{m,n}} \cdot r_m \,|\, a_{m,n} \cdot r_m]^\top
\end{aligned}$$

Now for $\mathbf{b} = [b_{1,1}| \cdots |b_{1,n}|b_{2,1}| \cdots |b_{2,n}| \cdots |b_{m,1}| \cdots |b_{m,n}]^\top \in \mathbb{Z}_2^{mn}$ and $\sigma \in \mathsf{S}_m$, we define $\Psi'_{\mathbf{b},\sigma}$ that transform vector $\mathbf{z} \in \{0,1\}^{2mn}$ of the following form

$$\begin{aligned}
\mathbf{z} = [\ & z_{1,1}^{(0)} \mid z_{1,1}^{(1)} \mid z_{1,2}^{(0)} \mid z_{1,2}^{(1)} \mid \cdots \mid z_{1,n}^{(0)} \mid z_{1,n}^{(1)} \mid \\
& z_{2,1}^{(0)} \mid z_{2,1}^{(1)} \mid z_{2,2}^{(0)} \mid z_{2,2}^{(1)} \mid \cdots \mid z_{2,n}^{(0)} \mid z_{2,n}^{(1)} \mid \cdots \mid \\
& z_{m,1}^{(0)} \mid z_{m,1}^{(1)} \mid z_{m,2}^{(0)} \mid z_{m,2}^{(1)} \mid \cdots \mid z_{m,n}^{(0)} \mid z_{m,n}^{(1)} \,]
\end{aligned}$$

to vector $\Psi'_{\mathbf{b},\sigma}$ of the following form

$$\begin{aligned}
\Psi'_{\mathbf{b},\sigma}(\mathbf{z}) = [\ & y_{1,1}^{(0)} \mid y_{1,1}^{(1)} \mid y_{1,2}^{(0)} \mid y_{1,2}^{(1)} \mid \cdots \mid y_{1,n}^{(0)} \mid y_{1,n}^{(1)} \mid \\
& y_{2,1}^{(0)} \mid y_{2,1}^{(1)} \mid y_{2,2}^{(0)} \mid y_{2,2}^{(1)} \mid \cdots \mid y_{2,n}^{(0)} \mid y_{2,n}^{(1)} \mid \cdots \mid \\
& y_{m,1}^{(0)} \mid y_{m,1}^{(1)} \mid y_{m,2}^{(0)} \mid y_{m,2}^{(1)} \mid \cdots \mid y_{m,n}^{(0)} \mid y_{m,n}^{(1)} \,]
\end{aligned}$$

such that $y_{i,j}^{(0)} = z_{\sigma(i),j}^{(b_{\sigma(i),j})}$ and $y_{i,j}^{(1)} = z_{\sigma(i),j}^{(\overline{b_{\sigma(i),j}})}$ for $i \in [n], j \in [m]$. For ease of notation, given $\mathbf{f} = (\mathbf{f}_1 \| \cdots \| \mathbf{f}_m) \in \{0,1\}^{mn}$, where each $\mathbf{f}_i \in \{0,1\}^n$, and $\sigma \in \mathsf{S}_m$, define

$$\sigma^{(n)}(\mathbf{f}) = (\, \mathbf{f}_{\sigma(1)} \,\|\, \cdots \,\|\, \mathbf{f}_{\sigma(m)} \,).$$

Precisely, $\sigma^{(n)}$ permutes the blocks of $\mathbf{f}$ using σ. The following equivalence then immediately follows from (20) for $\mathbf{a}, \mathbf{b} \in \{0,1\}^{mn}$, $\mathbf{r} \in \mathsf{B}(m, t_r)$, $\sigma_r \in \mathsf{S}_m$.

$$\mathbf{z} = \mathsf{expand}'(\mathbf{a}, \mathbf{r}) \Longleftrightarrow \Psi'_{\mathbf{b},\sigma_r}(\mathbf{z}) = \mathsf{expand}'\big(\sigma_r^{(n)}(\mathbf{a} \oplus \mathbf{b}), \sigma_r(\mathbf{r})\big). \tag{21}$$

The zero-knowledge argument. We now transform the relation R_{VLPN} to an instance of R_{abstract} such that the equivalences in (2) hold. Write $\mathbf{A} = [\,\mathbf{a}_1 \,|\, \cdots \,|\, \mathbf{a}_m\,] \in \mathbb{Z}_2^{n \times m}$ and $\mathbf{r} = [\,r_1 \,|\, \cdots \,|\, r_m\,]^\top \in \mathbb{Z}_2^m$, then we have

$$\begin{aligned}
\mathbf{A} \cdot \mathbf{r} &= \sum_{i=1}^{m} \mathbf{a}_i \cdot r_i = \sum_{i=1}^{m} [\,a_{i,1} \cdot r_i \,|\, a_{i,2} \cdot r_i \,|\, \cdots \,|\, a_{i,n} \cdot r_i\,]^\top \\
&= \sum_{i=1}^{m} \big[\,\mathbf{h}' \cdot \mathsf{ext}'(a_{i,1}, r_i) \,|\, \mathbf{h}' \cdot \mathsf{ext}'(a_{i,2}, r_i) \,|\, \cdots \,|\, \mathbf{h}' \cdot \mathsf{ext}'(a_{i,n}, r_i)\,\big]^\top \\
&= \sum_{i=1}^{m} \mathbf{H}'_{n,1}\big(\,\mathsf{ext}'(a_{i,1}, r_i) \,\|\, \mathsf{ext}'(a_{i,2}, r_i) \,\|\, \cdots \,\|\, \mathsf{ext}'(a_{i,n}, r_i)\,\big) \\
&= \sum_{i=1}^{m} \mathbf{H}'_{n,1} \cdot \mathbf{z}_i \\
&= \underbrace{[\mathbf{H}'_{n,1}|\cdots|\mathbf{H}'_{n,1}]}_{m \text{ times}} \cdot (\,\mathbf{z}_1 \,\|\, \cdots \,\|\, \mathbf{z}_m\,),
\end{aligned}$$

where $\mathbf{H}'_{n,1} = \begin{pmatrix} \mathbf{h}' & & & \\ & \mathbf{h}' & & \\ & & \ddots & \\ & & & \mathbf{h}' \end{pmatrix} \in \mathbb{Z}_2^{n \times 2n}$ and $\mathbf{z}_i = (\mathsf{ext}'(a_{i,1}, r_i) \| \cdots \| \mathsf{ext}'(a_{i,n}, r_i)) \in \mathbb{Z}_2^{2n}$. Denote $\mathbf{H}'_{n,m} = \underbrace{[\mathbf{H}'_{n,1}|\cdots|\mathbf{H}'_{n,1}]}_{m \text{ times}} \in \mathbb{Z}_2^{n \times 2mn}$, $\mathbf{z} = (\mathbf{z}_1 \| \cdots \| \mathbf{z}_m) \in \mathbb{Z}_2^{2mn}$, and $\mathbf{a} = [a_{1,1}|\cdots|a_{1,n}|a_{2,1}|\cdots|a_{2,n}|\cdots|a_{m,1}|\cdots|a_{m,n}]^\top \in \mathbb{Z}_2^{mn}$. Then $\mathbf{z}$ is indeed the extended expansion vector of $\mathbf{a}$ and $\mathbf{r}$, i.e., $\mathbf{z} = \mathsf{expand}'(\mathbf{a}, \mathbf{r})$. If no ambiguity caused, we write $\mathbf{z} = \mathsf{expand}'(\mathbf{A}, \mathbf{r})$. Hence, we obtain the following:

$$\mathbf{c} = \mathbf{A} \cdot \mathbf{r} \oplus \mathbf{e} \Longleftrightarrow \mathbf{c} = \mathbf{H}'_{n,m} \cdot \mathsf{expand}'(\mathbf{A}, \mathbf{r}) \oplus \mathbf{e}. \tag{22}$$

Denote $\mathbf{M}_{\mathrm{VLPN}} = [\mathbf{H}'_{n,m}|\mathbf{I}_n] \in \mathbb{Z}_2^{n \times L_{\mathrm{VLPN}}}$ and $\mathbf{w}_{\mathrm{VLPN}} = (\mathsf{expand}'(\mathbf{A}, \mathbf{r}) \| \mathbf{e}) \in \mathbb{Z}_2^{L_{\mathrm{VLPN}}}$ with $L_{\mathrm{VLPN}} = 2mn + n$. Hence $\mathbf{c} \stackrel{\triangle}{=} \mathbf{v}_{\mathrm{VLPN}} = \mathbf{M}_{\mathrm{VLPN}} \cdot \mathbf{w}_{\mathrm{VLPN}} \bmod 2$.

Now we are ready to specify the set $\mathsf{VALID}_{\mathrm{VLPN}}$ that contains of secret vector $\mathbf{w}_{\mathrm{VLPN}}$, the set $\mathcal{S}_{\mathrm{VLPN}}$, and permutations $\{\Gamma_\phi : \phi \in \mathcal{S}_{\mathrm{VLPN}}\}$ such that the equivalences in (2) hold. To this end, let $\mathsf{VALID}_{\mathrm{VLPN}}$ contain all vectors $\widehat{\mathbf{w}}_{\mathrm{VLPN}} = (\widehat{\mathbf{z}} \| \widehat{\mathbf{e}}) \in \mathbb{Z}_2^{2mn+n}$ satisfying the following constraints:

- There exists $\widehat{\mathbf{a}} \in \mathbb{Z}_2^{nm}$ and $\widehat{\mathbf{r}} \in \mathsf{B}(m, t_r)$ such that $\widehat{\mathbf{z}} = \mathsf{expand}'(\widehat{\mathbf{a}}, \widehat{\mathbf{r}})$.
- $\widehat{\mathbf{e}} \in \mathsf{B}(n, t)$.

It is easy to that the secret vector $\mathbf{w}_{\mathrm{VLPN}}$ belongs to $\mathsf{VALID}_{\mathrm{VLPN}}$. Let $\mathcal{S}_{\mathrm{VLPN}} = \{0,1\}^{mn} \times \mathsf{S}_m \times \mathsf{S}_n$. Then for each $\phi = (\mathbf{b},\, \sigma_r,\, \sigma_e) \in \mathcal{S}_{\mathrm{VLPN}}$, define the permutation Γ_ϕ that transforms vector of the form $\widehat{\mathbf{w}}_{\mathrm{VLPN}} = (\,\widehat{\mathbf{z}} \,\|\, \widehat{\mathbf{e}}\,)$ with $\widehat{\mathbf{z}} \in \mathbb{Z}_2^{2mn}, \widehat{\mathbf{e}} \in \mathbb{Z}_2^n$ to vector $\Gamma_\phi(\widehat{\mathbf{w}}_{\mathrm{VLPN}}) = (\,\Psi'_{\mathbf{b},\sigma_r}(\widehat{\mathbf{z}}) \,\|\, \sigma_e(\widehat{\mathbf{e}})\,)$.

Based on the equivalence observed in (21 and (3), it can be checked that the conditions in (2) are satisfied and we have successfully reduced the consider relation R_{VLPN} to an instance of R_{abstract}. Now $\mathcal{P}$ and $\mathcal{V}$ can interact as described in Figure 2. The resulting protocol is a statistical ZKAoK with perfect completeness, soundness error 2/3, and communication cost $\mathcal{O}(L_{\mathrm{VLPN}}) = \mathcal{O}(mn) = \mathcal{O}(\lambda^2)$ bits.

5 Message Filtering in Zero-Knowledge

In this section, we first specify the 2 policies we use for filtering messages encrypted in the code-based FDGE scheme of Section 6. Then we discuss our main ideas for proving in zero-knowledge that the underlying messages satisfy the given policies.

5.1 Formulation

Let $p,\, t,\, d \in \mathbb{Z}^+$ such that $p > t > d$. A string $\mathbf{y} = [y_1|\cdots|y_t]^\top \in \{0,1\}^t$ is called a substring of string $\mathbf{w} = [w_1|\cdots|w_p]^\top \in \{0,1\}^p$, denoted as $\mathbf{y} \sqsubset \mathbf{w}$, if there exists an integer $i \in [1, p-t+1]$ such that $y_j = w_{i+j-1}$ for all $j \in [1,t]$. The Hamming distance between $\mathbf{x}, \mathbf{y} \in \{0,1\}^t$, denoted by $d_H(\mathbf{x},\mathbf{y})$, is the number of coordinates at which $\mathbf{x}$ and $\mathbf{y}$ differ. In other words, $d_H(\mathbf{x},\mathbf{y}) = wt(\mathbf{x} \oplus \mathbf{y})$.

Let $\mathbf{w} \in \{0,1\}^p$ be an encrypted message and let $S = \{\mathbf{s}_1, \ldots, \mathbf{s}_k\}$ be a given list of $k \geq 1$ keywords, where $\mathbf{s}_i \in \{0,1\}^t$, for all $i \in [1,k]$. We will realize 2 commonly used policies of message filtering.

1. **"Permissive":** $\mathbf{w}$ is a legitimate message if and only if there exists $i \in [1,k]$ such that $\mathbf{s}_i$ is a substring of $\mathbf{w}$. The induced relation R_{permit} is defined as

$$R_{\mathrm{permit}} = \big\{\big((\mathbf{s}_1,\ldots,\mathbf{s}_k),\mathbf{w}\big) \in (\{0,1\}^t)^k \times \{0,1\}^p :\ \exists i \in [1,k] \ \text{ s.t. } \mathbf{s}_i \sqsubset \mathbf{w}\big\}. \quad (23)$$

2. **"Prohibitive":** $\mathbf{w}$ is a legitimate message if and only if for every length-t substring $\mathbf{y}$ of $\mathbf{w}$ and every $\mathbf{s}_i \in S$, their Hamming distance is at least d. The corresponding relation R_{prohibit} is defined as

$$\begin{aligned} R_{\mathrm{prohibit}} = \big\{\ \big((\mathbf{s}_1,\ldots,\mathbf{s}_k),\mathbf{w}\big) &\in (\{0,1\}^t)^k \times \{0,1\}^p : \\ & d_H(\mathbf{s}_i,\mathbf{y}) \geq d, \forall i \in [1,k], \forall \mathbf{y} \sqsubset \mathbf{w}\big\}. \end{aligned} \quad (24)$$

In the following, we will discuss our strategies for proving that message $\mathbf{w}$ satisfies each of the above policies.

5.2 Zero-Knowledge for the Permissive and Prohibitive Relations

Let $\mathbf{w} = [w_1|\cdots|w_p]^\top$, and for each $i \in [p-t+1]$, let $\mathbf{w}_{[i]} = [w_i|\cdots|w_{i+t-1}]^\top$ be its i-th substring of length t. Our ideas for proving that $\big((\mathbf{s}_1,\ldots,\mathbf{s}_k),\mathbf{w}\big) \in R_{\mathsf{permit}}$ in ZK are as follows. First, we form matrices

$$\mathbf{W} = [\mathbf{w}_{[1]} \mid \cdots \mid \mathbf{w}_{[p-t+1]}] = \begin{bmatrix} w_1 & w_2 & \cdots & w_{p-t+1} \\ w_2 & w_3 & \cdots & w_{p-t+2} \\ \vdots & \vdots & \vdots & \vdots \\ w_t & w_{t+1} & \cdots & w_p \end{bmatrix} \in \{0,1\}^{t\times(p-t+1)},$$

$\mathbf{S} = [\mathbf{s}_1 \mid \cdots \mid \mathbf{s}_k] \in \{0,1\}^{t\times k}$, and denote $\mathsf{permit}(\mathbf{w}) = (\mathbf{w}_{[1]}\|\cdots\|\mathbf{w}_{[p-t+1]}) \in \{0,1\}^{t(p-t+1)}$. We note that $\big((\mathbf{s}_1,\ldots,\mathbf{s}_k),\mathbf{w}\big) \in R_{\mathrm{permit}}$ if and only if there exist a column $\mathbf{w}_{[i]}$ of $\mathbf{W}$ and a column $\mathbf{s}_j$ of $\mathbf{S}$ such that $\mathbf{w}_{[i]} = \mathbf{s}_j$. Then, we observe that the task of the prover $\mathcal{P}$ is equivalent to proving the existence of $\mathbf{W},\mathbf{g},\mathbf{h}$ such that

$$\mathbf{W}\cdot\mathbf{g} = \mathbf{S}\cdot\mathbf{h} \quad\wedge\quad \mathbf{g} \in \mathsf{B}(p-t+1,1) \quad\wedge\quad \mathbf{h} \in \mathsf{B}(k,1).$$

To this end, we employ techniques for proving linear relation and quadratic relation (specifically the variant of LPN relation), as well as, for fix-weight relations in the framework of Stern's protocols. In the process, we prove the well-formedness of $\mathbf{W}$. The resulting protocol has communication cost $\mathcal{O}(t\cdot(p-t)+k)$ and is a sub-protocol in our FDGE construction of Section 6, where we additionally prove that $\mathbf{w}$ is the same as the plaintext encrypted in a given McEliece ciphertext.

The goal is to reduce the above relation R_{permit} to an instance of R_{abstract} so that conditions in (2) are fulfilled.

We observe that $\mathbf{W}\cdot\mathbf{g} \oplus \mathbf{S}\cdot\mathbf{h} = \mathbf{0} \bmod 2$ is equivalent to

$$\mathbf{H}'_{t,p-t+1} \cdot \mathsf{expand}'\big(\mathsf{permit}(\mathbf{w}),\ \mathbf{g}\big) \oplus \mathbf{S}\cdot\mathbf{h} = \mathbf{0} \bmod 2. \tag{25}$$

Denote $\mathbf{M}_{\mathrm{permit}} = [\mathbf{H}'_{t,p-t+1}|\mathbf{S}] \in \mathbb{Z}_2^{n\times L_{\mathrm{permit}}}$, $\mathbf{w}_{\mathrm{permit}} = (\mathbf{z}\|\mathbf{h}) \in \mathbb{Z}_2^{L_{\mathrm{permit}}}$ with $\mathbf{z} = \mathsf{expand}'(\mathsf{permit}(\mathbf{w}),\mathbf{g})$ and $L_{\mathrm{permit}} = 2t(p-t+1)+k$. Hence equation (25) can be written as $\mathbf{M}_{\mathrm{permit}}\cdot\mathbf{w}_{\mathrm{permit}} = \mathbf{0} \bmod 2$.

Now we specify the set $\mathsf{VALID}_{\mathrm{permit}}$ that contains of secret vector $\mathbf{w}_{\mathrm{permit}}$, the set $\mathcal{S}_{\mathrm{permit}}$, and permutations $\{\Gamma_\phi : \phi \in \mathcal{S}_{\mathrm{permit}}\}$ such that the equivalences in (2) hold. To this end, let $\mathsf{VALID}_{\mathrm{permit}}$ contain all vectors $\widehat{\mathbf{w}}_{\mathrm{permit}} = (\widehat{\mathbf{z}}\|\widehat{\mathbf{h}}) \in \mathbb{Z}_2^{2t(p-t+1)+k}$ satisfying the following constraints:

- $\widehat{\mathbf{z}} = \mathsf{expand}'(\widehat{\sigma}^{(t)}(\mathsf{permit}(\widehat{\mathbf{w}})),\widehat{\mathbf{g}})$ for some $\widehat{\mathbf{w}} \in \mathbb{Z}_2^p$, $\widehat{\mathbf{g}} \in \mathsf{B}(p-t+1,1)$, and $\widehat{\sigma} \in \mathsf{S}_{p-t+1}$.
- $\widehat{\mathbf{h}} \in \mathsf{B}(k,1)$.

It is easy to see that the secret vector $\mathbf{w}_{\mathrm{permit}}$ belongs to $\mathsf{VALID}_{\mathrm{permit}}$ for $\widehat{\sigma}$ being the identity of the group S_{p-t+1}. Let $\mathcal{S}_{\mathrm{permit}} = \{0,1\}^p \times \mathsf{S}_{p-t+1} \times \mathsf{S}_k$. Then for

each $\phi = (\mathbf{t},\, \sigma_g,\, \sigma_h) \in \mathcal{S}_{\text{permit}}$, define the permutation Γ_ϕ that transforms vector of the form $\widehat{\mathbf{w}}_{\text{permit}} = (\,\widehat{\mathbf{z}} \,\|\, \widehat{\mathbf{h}}\,)$ with $\widehat{\mathbf{z}} \in \mathbb{Z}_2^{2t(p-t+1)}, \widehat{\mathbf{h}} \in \mathbb{Z}_2^k$ to vector

$$\Gamma_\phi(\widehat{\mathbf{w}}_{\text{permit}}) = (\,\Psi'_{\mathsf{permit}(\mathbf{t}),\sigma_g}(\widehat{\mathbf{z}}) \,\|\, \sigma_h(\widehat{\mathbf{h}})\,).$$

We remark that, for $\widehat{\mathbf{w}} \in \mathbb{Z}_2^p$, $\widehat{\mathbf{g}} \in \mathsf{B}(p-t+1,1)$, $\widehat{\sigma} \in \mathsf{S}_{p-t+1}$, and $\phi = (\mathbf{t}, \sigma_g) \in \mathbb{Z}_2^p \times \mathsf{S}_{p-t+1}$, the following equivalence holds:

$$\begin{aligned}\widehat{\mathbf{z}} = \mathsf{expand}'(\widehat{\sigma}^{(t)}(\mathsf{permit}(\widehat{\mathbf{w}})),\ \widehat{\mathbf{g}}) \iff \\ \Psi'_{\mathsf{permit}(\mathbf{t}),\sigma_g}(\widehat{\mathbf{z}}) \ = \mathsf{expand}'(\sigma_g^{(t)}(\mathsf{permit}(\widehat{\mathbf{w}} \oplus \mathbf{t})), \sigma_g(\widehat{\mathbf{g}})).\end{aligned} \quad (26)$$

In contrast to equivalence (21), $\Psi'_{\mathsf{permit}(\mathbf{t}),\sigma_1}$ instead of $\Psi'_{\mathbf{d},\sigma_1}$ for a random $\mathbf{d} \in \{0,1\}^{t(p-t+1)}$ is employed here. This is critical to prove the well-formedness of $\mathsf{permit}(\widehat{\mathbf{w}})$. Based on the equivalence observed in (26) and (3), it can be checked that the conditions in (2) are satisfied and we have successfully reduced the consider relation R_{permit} to an instance of R_{abstract}. Now $\mathcal{P}$ and $\mathcal{V}$ can interact as described in Figure 2. The resulting protocol is a statistical ZKAoK with perfect completeness, soundness error $2/3$, and communication cost $\mathcal{O}(L_{\text{permit}}) = \mathcal{O}(t \cdot (p-t) + k)$ bits. It serves as a sub-protocol in our main zero-knowledge argument in Section 7.2, where we additionally prove that $\mathbf{w}$ is the same as the plaintext encrypted in a given McEliece ciphertext.

On the other hand, to prove that $\big((\mathbf{s}_1, \ldots, \mathbf{s}_k), \mathbf{w}\big) \in R_{\text{prohibit}}$, we consider $(p-t+1) \cdot k$ pairs $(\mathbf{w}_{[i]}, \mathbf{s}_j)$ and aim to prove that all the sums $\mathbf{z}_{i,j} = \mathbf{w}_{[i]} \oplus \mathbf{s}_j \in \{0,1\}^t$ have Hamming weight at least d. In other words, we reduce the problem to $(p-t+1) \cdot k$ sub-problems, for each of which, we needs to prove that $\mathbf{z}_{i,j}$ contains at least d coordinates equal to 1. To this end, we perform the following extension trick.

We append $(t-d)$ coordinates to $\mathbf{z}_{i,j} \in \{0,1\}^t$ to get $\mathbf{z}^*_{i,j} \in \{0,1\}^{2t-d}$ such that $wt(\mathbf{z}^*_{i,j}) = t$, i.e., $\mathbf{z}^*_{i,j} \in \mathsf{B}(2t-d, t)$. We note that such an extension is always possible if $wt(\mathbf{z}_{i,j}) \geq d$. Furthermore, the converse also holds: if $\mathbf{z}^*_{i,j} \in \mathsf{B}(2t-d,t)$, then the original $\mathbf{z}_{i,j}$ must have weight at least $t-(t-d) = d$. In addition, for any $\sigma \in \mathsf{S}_{2t-d}$, the following equivalence holds:

$$wt(\mathbf{z}_{i,j}) \geq d \iff wt(\mathbf{z}^*_{i,j}) \in \mathsf{B}(2t-d,t) \iff \sigma(\mathbf{z}^*_{i,j}) \in \mathsf{B}(2t-d,t). \quad (27)$$

Let $\mathbf{Q}^*_t = [\mathbf{I}_t | \mathbf{0}^{t\times(t-d)}] \in \mathbb{Z}_2^{t\times(2t-d)}$, then $\mathbf{z}_{i,j} = \mathbf{Q}^*_t \cdot \mathbf{z}^*_{i,j}$.

Let us now transform relation R_{prohibit} to an instance of R_{abstract} that satisfies the conditions in (2). Towards the goal, we employ the aforementioned technique as well as the one handling arbitrary binary vectors.

We first extend secret vectors in those $(p-t+1) \cdot k$ equation. Let $\mathbf{x} = \mathsf{Encode}(\mathbf{w}) \in \mathbb{Z}_2^{2p}$ and $\mathbf{z}^*_{i,j} \in \mathsf{B}(2t-d,t)$ be extension of $\mathbf{z}_{i,j}$ for $i \in [1, p-t+1]$ and $j \in [1,k]$. For $i \in [1, p-t+1]$, let $\mathbf{x}_{[i]} = \mathsf{Encode}(\mathbf{w}_{[i]}) \in \mathbb{Z}_2^{2t}$. Then

$$\mathbf{w}_{[i]} \oplus \mathbf{z}_{i,j} = \mathbf{s}_j \iff \mathbf{I}^*_t \cdot \mathbf{x}_{[i]} \oplus \mathbf{Q}^*_t \cdot \mathbf{z}^*_{i,j} = \mathbf{s}_j, \forall\ i \in [1, p-t+1], j \in [1,k]. \quad (28)$$

Recall that $\mathbf{I}^*_t \in \mathbb{Z}_2^{t\times 2t}$ is obtained by inserting a zero-column $\mathbf{0}^t$ right before each column of $\mathbf{I}_t$. Through basic algebra, we are able to form $\mathbf{M}_{\text{prohibit}} \in$

$\mathbb{Z}_2^{kt(p-t+1)\times L_{\text{prohibit}}}$, $\mathbf{v}_{\text{prohibit}} = \underbrace{(\mathbf{s}_1\|\cdots\|\mathbf{s}_k\|\cdots\cdots\|\mathbf{s}_1\|\cdots\|\mathbf{s}_k)}_{p-t+1 \text{ times}} \in \mathbb{Z}_2^{kt(p-t+1)}$ that are public and

$$\mathbf{w}_{\text{prohibit}} = (\mathbf{x}\|\mathbf{z}^*_{1,1}\|\cdots\|\mathbf{z}^*_{1,k}\|\cdots\cdots\|\mathbf{z}^*_{p-t+1,1}\|\cdots\|\mathbf{z}^*_{p-t+1,k}) \in \mathbb{Z}_2^{L_{\text{prohibit}}}$$

that is secret for $L_{\text{prohibit}} = 2p + k \cdot (2t - d) \cdot (p - t + 1)$ such that (28) is transformed to $\mathbf{M}_{\text{prohibit}} \cdot \mathbf{w}_{\text{prohibit}} = \mathbf{v}_{\text{prohibit}} \bmod 2$.

Next, we specify the set $\mathsf{VALID}_{\text{prohibit}}$ that consists of $\mathbf{w}_{\text{prohibit}}$, the set $\mathcal{S}_{\text{prohibit}}$, and permutations $\{\Gamma_\phi : \phi \in \mathcal{S}_{\text{prohibit}}\}$. Let $\mathsf{VALID}_{\text{prohibit}}$ be a set that contains all vectors $\widehat{\mathbf{w}}_{\text{prohibit}} = (\widehat{\mathbf{x}}\|\widehat{\mathbf{z}}^*_{1,1}\|\cdots\|\widehat{\mathbf{z}}^*_{1,k}\|\widehat{\mathbf{z}}^*_{p-t+1,1}\|\cdots\|\widehat{\mathbf{z}}^*_{p-t+1,k}) \in \mathbb{Z}_2^{L_{\text{prohibit}}}$ such that

- There exists $\widehat{\mathbf{w}} \in \mathbb{Z}_2^p$ so that $\widehat{\mathbf{x}} = \mathsf{Encode}(\widehat{\mathbf{w}})$;
- $\widehat{\mathbf{z}}^*_{i,j} \in \mathsf{B}(2t - d, t)$ for all $i \in [1, p - t + 1], j \in [1, k]$.

It is straightforward that $\mathbf{w}_{\text{prohibit}} \in \mathsf{VALID}_{\text{prohibit}}$. Define $\mathcal{S}_{\text{prohibit}} = \{0,1\}^p \times (\mathsf{S}_{2t-d})^{k(p-t+1)}$. Then for any $\phi = (\mathbf{b}, \sigma_{1,1}, \ldots, \sigma_{1,k}, \ldots, \sigma_{p-t+1,1}, \ldots, \sigma_{p-t+1,k}) \in \mathcal{S}_{\text{prohibit}}$, for any $\widehat{\mathbf{w}}_{\text{prohibit}} = (\widehat{\mathbf{x}}\|\widehat{\mathbf{z}}^*_{1,1}\|\cdots\|\widehat{\mathbf{z}}^*_{1,k}\|\cdots\|\widehat{\mathbf{z}}^*_{p-t+1,1}|\cdots\|\widehat{\mathbf{z}}^*_{p-t+1,k})$ with $\widehat{\mathbf{x}} \in \mathbb{Z}_2^{2p}$ and $\widehat{\mathbf{z}}^*_{i,j} \in \mathbb{Z}_2^{2t-d}$, Γ_ϕ permutes $\widehat{\mathbf{w}}_{\text{prohibit}}$ to vector

$$\Gamma_\phi(\widehat{\mathbf{w}}_{\text{prohibit}}) = \big(F_{\mathbf{b}}(\widehat{\mathbf{x}}) \,\|\, \sigma_{1,1}(\widehat{\mathbf{z}}^*_{1,1}) \,\|\cdots\|\, \sigma_{1,k}(\widehat{\mathbf{z}}^*_{1,k}) \,\|\cdots \,\|\, \sigma_{p-t+1,1}(\widehat{\mathbf{z}}^*_{p-t+1,1})\|\cdots\|\sigma_{p-t+1,k}(\widehat{\mathbf{z}}^*_{p-t+1,k})\big)$$

Based on the equivalences (5) and (27), $\widehat{\mathbf{w}}_{\text{prohibit}} \in \mathsf{VALID}_{\text{prohibit}}$ if and only if $\Gamma_\phi(\widehat{\mathbf{w}}_{\text{prohibit}}) \in \mathsf{VALID}_{\text{prohibit}}$. Moreover, if $\phi \xleftarrow{\$} \mathcal{S}_{\text{prohibit}}$, then $\Gamma_\phi(\widehat{\mathbf{w}}_{\text{prohibit}})$ is uniformly distributed in $\mathsf{VALID}_{\text{prohibit}}$. Therefore, $\mathcal{P}$ and $\mathcal{V}$ interact as in Figure 2. The resulting protocol is a statistical ZKAoK with perfect completeness, soundness error 2/3, and communication cost $\mathcal{O}(L_{\text{prohibit}}) = \mathcal{O}(k \cdot t \cdot (p-t+1))$. Similarly to the case of R_{permit}, this protocol can also be integrated into the zero-knowledge argument in Section 7.2, allowing us to realize the "prohibitive" filtering policy.

6 A Code-Based Fully Dynamic Group Encryption

To build a code-based FDGE scheme, we require a key-private CCA2-secure encryption scheme [6], a digital signature scheme, and a zero-knowledge proof (argument) of knowledge protocol. In this paper, we work with the ZKAoK within Stern's framework [58]. In terms of the encryption scheme, we choose to work with the McEliece cryptosystem [47], specifically the randomized variant from [54]. The latter indeed has pseudorandom ciphertexts, which implies key-private CPA-security. To further achieve CCA2-security, we apply the Naor-Yung double encryption technique [52]. Note that there are other CCA2-secure variants of McEliece scheme like [24,21,35]. However, they either do not operate

well in the Stern's framework or are completely impractical. Regarding the digital signature, we employ the Merkle-tree accumulator described in Section 3.1. Precisely, when a user requests to join the group, it first generates its encryption key pair $(\mathsf{pk}, \mathsf{sk})$, and sends pk and its non-zero hash value $\mathbf{d}$ to GM. The latter, if accepts, then computes the Merkle tree root, where the leaf nodes are the hash values of all users. The witness for $\mathbf{d}$ is the proof of user's membership. To achieve dynamicity, following [44], we use the updating algorithm described in Section 3.2 to set up the system so that (1) the value of the leaf node associated with a user who has not joined or who has been removed from the group is $\mathbf{0}$ (2) while it is updated to $\mathbf{d}$ when this user joins the group. When a sender encrypts messages to a user at some epoch, it has to show that the user's *non-zero* hash value is accumulated in the tree in this epoch. This mechanism effectively distinguish active users who are valid recipients of ciphertexts from those who are not.

As in the KTY model [34], we also require that user encryption keys are valid (i.e., in the language $\mathcal{L}_{\mathsf{pk}}^{\mathsf{pp}}$). One possible solution would be requiring a proof of knowledge of the McEliece decryption key when a user joins the group. This is however quite complicated and inefficient. Instead, GM encrypts random messages under the user's encryption key and asks the user to output the correct messages. By choosing the parameters properly, the running time of guessing correctly the messages if the user does not know the underlying decryption key is exponential. This then enforces validity of user encryption keys.

6.1 Description of the Scheme

Our scheme allows encryption witness $\mathbf{w} \in \{0,1\}^p$ that satisfies the permissive relation R_{permit} and/or the prohibitive relation R_{prohibit}. For simplicity, we present R_{permit} in the following construction. The details are described below.

$\mathsf{Setup}_{\mathsf{init}}(1^\lambda)$ On input the security parameter 1^λ, this algorithm proceeds as follows.

- Specify an integer $\ell = \ell(\lambda)$ that determines the maximum expected number $N = 2^\ell$ of potential users.
- Choose $n = \mathcal{O}(\lambda)$, $c = \mathcal{O}(1)$ such that c divides n, and set $m = 2^c \cdot \frac{2n}{c}$. Choose an integer $t_m < m$.
- Choose $t_1 = t_1(\lambda)$, $k_1 = k_1(\lambda)$ and $t_2 = t_2(\lambda)$, $k_2 = k_2(\lambda)$ such that (n, k_1, t_1), (n, k_2, t_2) are two sets of parameters for the McEliece encryption scheme (see Section 3.3).
- Sample a random matrix $\mathbf{B} \xleftarrow{\$} \mathbb{Z}_2^{n \times m}$ that specifies a hash function $h_{\mathbf{B}}$ as in Definition 6.
- Pick a statistical hiding and computationally binding commitment scheme $\mathsf{COM} : \{0,1\}^* \to \{0,1\}^n$ like the one in [53, Section 3.1]. This will serve as a building block for the argument systems in Section 7.2.
- Let $\mathcal{H}_{\mathsf{FS}} : \{0,1\}^* \to \{1,2,3\}^\kappa$, where $\kappa = \omega(\log \lambda)$, be a hash function that will be modeled as a random oracle in the Fiat-Shamir transforms [26].

Output public parameters

$$\mathsf{pp} = \{N, \ell, n, c, m, t_m, t_1, k_1, p, t_2, k_2, v, \mathbf{B}, \mathsf{COM}, \kappa, \mathcal{H}_{\mathsf{FS}}\}.$$

$\mathsf{Setup}_{\mathsf{OA}}(\mathsf{pp})$ This algorithm is run by the OA. Given the input pp, it triggers the McEliece key generation algorithm $\mathsf{KeyGen}_{\mathsf{ME}}(n, k_1, t_1)$ (see Section 3.3) twice to obtain encryption key pairs $(\mathbf{G}_{\mathsf{oa},0}, \mathsf{sk}_{\mathsf{ME}}^{(\mathsf{oa},0)})$ and $(\mathbf{G}_{\mathsf{oa},1}, \mathsf{sk}_{\mathsf{ME}}^{(\mathsf{oa},1)})$. Set $\mathsf{pk}_{\mathsf{OA}} = (\mathbf{G}_{\mathsf{oa},0}, \mathbf{G}_{\mathsf{oa},1})$ and $\mathsf{sk}_{\mathsf{OA}} = (\mathsf{sk}_{\mathsf{ME}}^{(\mathsf{oa},0)}, \mathsf{sk}_{\mathsf{ME}}^{(\mathsf{oa},1)})$.

$\mathsf{Setup}_{\mathsf{GM}}(\mathsf{pp})$ This algorithm is run by the GM. It samples $\mathsf{sk}_{\mathsf{GM}} \xleftarrow{\$} \mathsf{B}(m, t_m)$, then computes $\mathsf{pk}_{\mathsf{GM}} = \mathbf{B} \cdot \mathsf{sk}_{\mathsf{GM}} \bmod 2$, and outputs $(\mathsf{pk}_{\mathsf{GM}}, \mathsf{sk}_{\mathsf{GM}})$. It also initializes the following.

- Let the registration table be $\mathbf{reg} := (\mathbf{reg}[0], \mathbf{reg}[1], \ldots, \mathbf{reg}[N–1])$, where for each $i \in [0, N-1]$: $\mathbf{reg}[i][1] = \mathbf{0}^n, \mathbf{reg}[i][2] = -1$, and $\mathbf{reg}[i][3] = -1$. Here, $\mathbf{reg}[i][1]$ denotes the hash value of the public encryption key of a registered user while $\mathbf{reg}[i][2], \mathbf{reg}[i][3]$ represent the epoch at which the user joins and leaves the group, respectively.
- Construct a Merkle tree $\mathcal{T}$ on top of $\mathbf{reg}[0][1], \ldots, \mathbf{reg}[N-1][1]$. (Note that $\mathcal{T}$ is an all-zero tree at this stage, when a new user joins the group, it will affect the Merkle tree.)
- Initialize a counter of registered users $j := 0$.

Then, GM outputs its public key $\mathsf{pk}_{\mathsf{GM}}$ and announces $\mathbf{reg}$ and the initial group information $\mathsf{info} = \emptyset$ while keeping $\mathcal{T}$ and j for himself. We remark that $\mathbf{reg}$ and info are visible to everyone but only editable by a party who knows $\mathsf{sk}_{\mathsf{GM}}$. In addition, anyone is able to verify the well-formedness of $\mathbf{reg}$ and info.

$\langle \mathsf{G}_{\mathcal{R}}, \mathsf{Sample}_{\mathcal{R}} \rangle$ The algorithm $\mathsf{G}_{\mathcal{R}}(1^\lambda, \mathsf{pp})$ proceeds by sampling parameters t, k for the relation R_{permit} (23). Let $(\mathsf{pk}_{\mathcal{R}}, \mathsf{sk}_{\mathcal{R}}) = ((p, t, k), \epsilon)$. Given $\mathsf{pk}_{\mathcal{R}}$, the algorithm $\mathsf{Sample}_{\mathcal{R}}$ outputs a set of keywords $S = \{\mathbf{s}_1, \ldots, \mathbf{s}_k\}$, $\mathbf{w} \in \mathbb{Z}_2^p$ such that $(S, \mathbf{w}) \in R_{\text{permit}}$.

$\langle \mathsf{Join}, \mathsf{Issue} \rangle$**.** This is an interactive protocol securely run between a user and the GM. If a user requests to join the group at epoch τ, he will follow steps below.

1. The user first generates its encryption key pair. It runs McEliece key generation $\mathsf{KeyGen}_{\mathsf{ME}}(n, k_2, t_2)$ twice, obtaining $(\mathbf{G}_0, \mathsf{sk}_{\mathsf{ME}}^{(0)})$ and $(\mathbf{G}_1, \mathsf{sk}_{\mathsf{ME}}^{(1)})$. Set encryption key $\mathsf{pk}' = (\mathbf{G}_0, \mathbf{G}_1)$ and secret key $\mathsf{sk} = (\mathsf{sk}_{\mathsf{ME}}^{(0)}, \mathsf{sk}_{\mathsf{ME}}^{(1)})$.
2. It then computes the hash of its encryption key pk'. For $b \in \{0, 1\}$, write $\mathbf{G}_b = [\mathbf{g}_{k_2 b} | \cdots | \mathbf{g}_{k_2 b + k_2 - 1}]$. Let $D = \{\mathbf{g}_0, \mathbf{g}_1, \ldots, \mathbf{g}_{2k_2-1}\}$. It then runs the accumulation algorithm $\mathsf{Accu}_{\mathbf{B}}(D)$ (see Section 3.1) to build a (sub)-Merkle tree based on D and the hash function $h_{\mathbf{B}}$, obtaining an accumulated hash value $\mathbf{d} \in \mathbb{Z}_2^n$. We call $\mathbf{d}$ the hash of pk'. If there is no ambiguity, we sometimes write $\mathsf{Accu}_{\mathbf{B}}(\mathsf{pk}')$ instead of $\mathsf{Accu}_{\mathbf{B}}(D)$.
3. If $\mathbf{d} = \mathbf{0}^n$, the user repeats Step 1 and 2. Otherwise, he sends $(\mathsf{pk}', \mathbf{d})$ to the GM.

Upon receiving the tuple $(\mathsf{pk}', \mathbf{d})$ from the user, the GM first computes the ranks r_1, r_2 of $\mathbf{G}_0, \mathbf{G}_1$, respectively, and $\mathbf{d}' = \mathsf{Accu}_{\mathbf{B}}(\mathsf{pk}')$. If $r_1 \neq k_2$ or $r_2 \neq k_2$ or $\mathbf{d}' \neq \mathbf{d}$ or $\mathbf{d}' = \mathbf{0}^n$, GM rejects. Otherwise, the two parties proceed as follows.

1. First, GM encrypts two random messages by running the deterministic McEliece encryption algorithm using the key pk'. It first samples $\mathbf{m}_0, \mathbf{m}_1 \xleftarrow{\$} \mathbb{Z}_2^{k_2}$ and $\mathbf{e}_0, \mathbf{e}_1 \xleftarrow{\$} \mathsf{B}(n, t_2)$, then computes $\mathbf{y}_0 = \mathbf{G}_0 \cdot \mathbf{m}_0 \oplus \mathbf{e}_0, \mathbf{y}_1 = \mathbf{G}_1 \cdot \mathbf{m}_1 \oplus \mathbf{e}_1$, and sends $\mathbf{y}_0, \mathbf{y}_1$ to the user.
2. Upon receiving the ciphertexts, user runs the deterministic McEliece decryption algorithm, obtaining $\mathbf{m}'_0, \mathbf{m}'_1$. The user then sends $\mathbf{m}'_0, \mathbf{m}'_1$ to the GM.
3. If $\mathbf{m}'_0 \neq \mathbf{m}_0$ or $\mathbf{m}'_1 \neq \mathbf{m}_1$, GM rejects. Otherwise GM issues an identifier to the user as $\mathsf{uid} = \mathsf{bin}(j) \in \{0,1\}^\ell$. The user then sets his public key as $\mathsf{pk} = (\mathsf{pk}', \mathsf{bin}(j))$. From now on, we write $\mathsf{pk}'_j = (\mathbf{G}_{j,0}, \mathbf{G}_{j,1}), \mathsf{sk}_j = (\mathsf{sk}_{\mathsf{ME}}^{(j,0)}, \mathsf{sk}_{\mathsf{ME}}^{(j,1)})$ to distinguish keys of different users.
4. GM also performs the following updates:
   - Update $\mathcal{T}$ by running the algorithm $\mathsf{TUpdate}_{\mathbf{B}}(\mathsf{bin}(j), \mathbf{d})$.
   - Register the user to table $\mathbf{reg}$ as $\mathbf{reg}[j][1] := \mathbf{d}$; $\mathbf{reg}[j][2] := \tau$.
   - Increase the counter $j := j + 1$.

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

1. Let the set $\mathcal{S}$ contain all the identifiers of registered users to be revoked. If $\mathcal{S} = \emptyset$, then go to Step 2.
   Otherwise, $\mathcal{S} = \{i_1, \ldots, i_r\}$, for some $i_1, \ldots, i_r \in [0, N-1]$. Then, for all $t \in [r]$, GM runs $\mathsf{TUpdate}_{\mathbf{B}}(\mathsf{bin}(i_t), \mathbf{0}^n)$ to update the tree $\mathcal{T}$. Meanwhile, GM updates $\mathbf{reg}[j][3] = \tau_{\text{new}}$.
2. At this point, each of the zero leaves in the tree $\mathcal{T}$ corresponds to either a revoked user or a potential user who has not yet registered. In other words, only active users in the new epoch τ_{new} have non-zero hashes of their encryption keys, denoted by $\{\mathbf{d}_j\}_j$, accumulated in the root $\mathbf{u}_{\tau_{\text{new}}}$ of the updated tree.
   For each j, let $w^{(j)} \in \{0,1\}^\ell \times (\{0,1\}^n)^\ell$ be the witness for the fact that $\mathbf{d}_j$ is accumulated in $\mathbf{u}_{\tau_{\text{new}}}$. Then GM publishes the group information of the new epoch as:

$$\mathsf{info}_{\tau_{\text{new}}} = \left(\mathbf{u}_{\tau_{\text{new}}}, \{w^{(j)}\}_j\right).$$

We remark that even though $\mathsf{info}_{\tau_{\text{new}}}$ can be as large as $\mathcal{O}(\lambda \cdot 2^\ell \cdot \ell)$, it is not necessary for the sender or verifier to download them all. In deed, the sender when running the $\mathcal{P}$ algorithm only needs to download the respective witness $w^{(j)}$ of size $\mathcal{O}(\lambda \cdot \ell)$ bits. Meanwhile, the verifier who runs the $\mathcal{V}$ algorithm only needs to download $\mathbf{u}_{\tau_{\text{new}}}$ of size $\mathcal{O}(\lambda)$ bits. It is also worth

noting that one is able to verify the well-formedness of registration table $\mathbf{reg}$ from group information $\mathsf{info}_{\tau_{\text{current}}}$ and $\mathsf{info}_{\tau_{\text{new}}}$[10], and vice versa[11].

$\mathsf{Enc}(\mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathsf{OA}}, \mathsf{info}_\tau, \mathbf{w}, \mathsf{pk}, L)$ Parse $\mathsf{pk}_{\mathsf{OA}} = (\mathbf{G}_{\mathsf{oa},0}, \mathbf{G}_{\mathsf{oa},1})$, $\mathsf{pk} = (\mathsf{pk}'_j, \mathsf{bin}(j))$ for some $j \in [0, N-1]$ and let $L \in \{0,1\}^*$. This algorithm is run by a sender who wishes to send a message $\mathbf{w} \in \mathbb{Z}_2^p$ such that $(S, \mathbf{w}) \in R_{\text{permit}}$ to a chosen user j with encryption key pk'_j. If user j is an active user at current epoch τ, the sender downloads the corresponding witness $w^{(j)} = (\mathsf{bin}(j), (\mathbf{w}_\ell, \cdots, \mathbf{w}_1))$ from info_τ and performs the following steps.

1. It first encrypts the message $\mathbf{w}$ under the encryption key pk'_j.
   - Parse $\mathsf{pk}'_j = (\mathbf{G}_{j,0}, \mathbf{G}_{j,1})$.
   - Sample randomnesses $\mathbf{r}_{w,0}, \mathbf{r}_{w,1} \xleftarrow{\$} \mathbb{Z}_2^{k_2-p}$ and noises $\mathbf{e}_{w,0}, \mathbf{e}_{w,1} \xleftarrow{\$} \mathsf{B}(n, t_2)$.
   - For $b \in \{0,1\}$, compute

$$\mathbf{c}_{w,b} = \mathbf{G}_{j,b} \cdot \begin{pmatrix} \mathbf{r}_{w,b} \\ \mathbf{w} \end{pmatrix} \oplus \mathbf{e}_{w,b} \in \mathbb{Z}_2^n. \tag{29}$$

   Let $\mathbf{c}_w = (\mathbf{c}_{w,0}, \mathbf{c}_{w,1}) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n$.
2. Next, it encrypts the user's identity j under the key $\mathsf{pk}_{\mathsf{OA}} = (\mathbf{G}_{\mathsf{oa},0}, \mathbf{G}_{\mathsf{oa},1})$.
   - Let $\mathsf{bin}(j) = [j_1 | \ldots | j_\ell]^\top \in \{0,1\}^\ell$.
   - Sample randomnesses $\mathbf{r}_{\mathsf{oa},0}, \mathbf{r}_{\mathsf{oa},1} \xleftarrow{\$} \mathbb{Z}_2^{k_1-\ell}$ and noises $\mathbf{e}_{\mathsf{oa},0}, \mathbf{e}_{\mathsf{oa},1} \xleftarrow{\$} \mathsf{B}(n, t_1)$.
   - For $b \in \{0,1\}$, compute

$$\mathbf{c}_{\mathsf{oa},b} = \mathbf{G}_{\mathsf{oa},b} \cdot \begin{pmatrix} \mathbf{r}_{\mathsf{oa},b} \\ \mathsf{bin}(j) \end{pmatrix} \oplus \mathbf{e}_{\mathsf{oa},b} \in \mathbb{Z}_2^n. \tag{30}$$

   Let $\mathbf{c}_{\mathsf{oa}} = (\mathbf{c}_{\mathsf{oa},0}, \mathbf{c}_{\mathsf{oa},1}) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n$.
3. It then generates a proof showing that $\mathbf{c}_{w,0}, \mathbf{c}_{w,1}$ both encrypt $\mathbf{w}$ and that $\mathbf{c}_{\mathsf{oa},0}, \mathbf{c}_{\mathsf{oa},1}$ both encrypt $\mathsf{bin}(j)$. This is done by running the protocol in Section 7.1 on public input $(\mathbf{G}_{\mathsf{oa},0}, \mathbf{G}_{\mathsf{oa},1}, \mathbf{c}_w, \mathbf{c}_{\mathsf{oa}}, L)$ and secret input $(\mathbf{G}_{j,0}, \mathbf{G}_{j,1}, \mathbf{w}, \mathsf{bin}(j), \mathbf{r}_{w,0}, \mathbf{r}_{w,1}, \mathbf{e}_{w,0}, \mathbf{e}_{w,1}, \mathbf{r}_{\mathsf{oa},0}, \mathbf{r}_{\mathsf{oa},1}, \mathbf{e}_{\mathsf{oa},0}, \mathbf{e}_{\mathsf{oa},1})$. The protocol is repeated κ times to achieve negligible soundness error and made non-interactive via Fiat-Shamir transform [26]. The resultant proof is a triple of form $\pi_{ct} = (\{\mathsf{CMT}_{ct,i}\}_{i=1}^\kappa, \mathsf{Ch}_{ct}, \{\mathsf{RSP}_{ct,i}\}_{i=1}^\kappa)$ such that

$$\mathsf{Ch}_{ct} = \mathcal{H}_{\mathsf{FS}}(\{\mathsf{CMT}_{ct,i}\}_{i=1}^\kappa, \mathbf{G}_{\mathsf{oa},0}, \mathbf{G}_{\mathsf{oa},1}, \mathbf{c}_w, \mathbf{c}_{\mathsf{oa}}, L).$$

Output the ciphertext $\psi = (\mathbf{c}_{w,0}, \mathbf{c}_{w,1}, \mathbf{c}_{\mathsf{oa},0}, \mathbf{c}_{\mathsf{oa},1}, \pi_{ct})$ and coins

$$\mathsf{coins}_\psi = (\mathbf{r}_{w,0}, \mathbf{r}_{w,1}, \mathbf{e}_{w,0}, \mathbf{e}_{w,1}, \mathbf{r}_{\mathsf{oa},0}, \mathbf{r}_{\mathsf{oa},1}, \mathbf{e}_{\mathsf{oa},0}, \mathbf{e}_{\mathsf{oa},1}). \tag{31}$$

[10] For instance, if $w^{(j)}$ does not appear in $\mathsf{info}_{\tau_{\text{current}}}$ but $\mathsf{info}_{\tau_{\text{new}}}$ then $\mathbf{reg}[j][2] = \tau_{\text{new}}$. On the other hand, if $w^{(j)}$ appears in $\mathsf{info}_{\tau_{\text{current}}}$ but not in $\mathsf{info}_{\tau_{\text{new}}}$ then $\mathbf{reg}[j][3] = \tau_{\text{new}}$.

[11] It is easy to figure out all active users at specific time τ from $\mathbf{reg}$, and thus enables verification of well-formedness of info_τ.

$\mathcal{P}\big(\mathsf{pp}, \mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathsf{OA}}, \mathsf{info}_\tau, S, \psi, L, \mathbf{w}, \mathsf{pk}, \mathsf{coins}_\psi\big)$ Let coins_ψ be of the form (31) and $\psi = (\mathbf{c}_{w,0}, \mathbf{c}_{w,1}, \mathbf{c}_{\mathsf{oa},0}, \mathbf{c}_{\mathsf{oa},1}, \pi_{ct})$. This algorithm is implemented by the sender above who has encrypted a message $\mathbf{w}$ to a user j at time epoch τ. The sender extracts $\mathbf{B}$ from pp. In addition to the witness $w^{(j)}$, he downloads $\mathbf{u}_\tau$ as well from info_τ. The goal of the sender is to convince the verifier in zero-knowledge that the following conditions hold.

1. The secret message $\mathbf{w} \in \mathbb{Z}_2^p$ is such that $(S, \mathbf{w}) \in R_{\text{permit}}$.
2. The user encryption key pk'_j is correctly hashed to a non-zero value $\mathbf{d}_j$. In other words, $\mathsf{Accu}_{\mathbf{B}}(\mathsf{pk}'_j) = \mathbf{d}_j$ and $\mathbf{d}_j \neq \mathbf{0}^n$.
3. The non-zero hash value $\mathbf{d}_j$ is honestly accumulated to value $\mathbf{u}_\tau$ at epoch τ, i.e., the equation $\mathsf{Verify}_{\mathbf{B}}(\mathbf{u}_\tau, \mathbf{d}_j, w^{(j)}) = 1$ holds.
4. $(\mathbf{c}_{w,0}, \mathbf{c}_{w,1}), (\mathbf{c}_{\mathsf{oa},0}, \mathbf{c}_{\mathsf{oa},1})$ are honest encryptions of $\mathbf{w}$ and $\mathsf{bin}(j)$, respectively. In other words, for $b \in \{0,1\}$, equations (29) and (30) hold.
5. The randomnesses $\mathbf{r}_{w,0}, \mathbf{r}_{w,1}, \mathbf{r}_{\mathsf{oa},0}, \mathbf{r}_{\mathsf{oa},1}$ are binary vectors while noises $\mathbf{e}_{w,0}, \mathbf{e}_{w,1}$ and $\mathbf{e}_{\mathsf{oa},0}, \mathbf{e}_{\mathsf{oa},1}$ are in the sets $\mathsf{B}(n, t_2)$ and $\mathsf{B}(n, t_1)$, respectively.

This is done by running the interactive protocol in Section 7.2 on public input $\big(\mathbf{B}, \mathsf{pk}_{\mathsf{OA}}, \mathbf{u}_\tau, S, \psi, L\big)$ and secret input $(\mathbf{w}, \mathsf{pk}, \mathsf{coins}_\psi, w^{(j)})$. To achieve negligible soundness error, the protocol is repeated κ times. To remove interaction, the protocol is further applied Fiat-Shamir heuristic [26]. The resulting proof is a triple $\pi_\psi = (\{\mathsf{CMT}_i\}_{i=1}^\kappa, \mathsf{Ch}, \{\mathsf{RSP}_i\}_{i=1}^\kappa)$ where

$$\mathsf{Ch} = \mathcal{H}_{\mathsf{FS}}(\{\mathsf{CMT}_i\}_{i=1}^\kappa, \mathbf{B}, \mathsf{pk}_{\mathsf{OA}}, \mathbf{u}_\tau, S, \psi, L) \in \{1,2,3\}^\kappa.$$

$\mathcal{V}\big((\mathsf{pp}, \mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathsf{OA}}, \mathsf{info}_\tau, S, \psi, L), \pi_\psi\big)$ This algorithm verifies the legitimacy of the ciphertext label pair (ψ, L) with respect to epoch τ and the set of keywords S by checking the validity of the proof π_ψ. It proceeds as follows.

1. Download $\mathbf{u}_\tau$ from info_τ.
2. Parse $\pi_\psi = (\{\mathsf{CMT}_i\}_{i=1}^\kappa, \mathsf{Ch}, \{\mathsf{RSP}_i\}_{i=1}^\kappa)$.
3. If $\mathsf{Ch} = [\mathsf{ch}_1 | \cdots | \mathsf{ch}_\kappa]^\top \neq \mathcal{H}_{\mathsf{FS}}(\{\mathsf{CMT}_i\}_{i=1}^\kappa, \mathbf{B}, \mathsf{pk}_{\mathsf{OA}}, \mathbf{u}_\tau, S, \psi, L)$, return 0.
4. For $i \in [1, \kappa]$, verify the validity of RSP_i with respect to the commitment CMT_i and the challenge ch_i. If any of the verifications does not hold, return 0. Else return 1.

$\mathsf{Dec}(\mathsf{info}_\tau, \mathsf{sk}, \psi, L)$ This algorithm is run by a user j with secret key sk. Parse $\mathsf{sk} = (\mathsf{sk}_{\mathsf{ME}}^{(j,0)}, \mathsf{sk}_{\mathsf{ME}}^{(j,1)})$. It performs the following steps.

1. Parse $\psi = (\mathbf{c}_{w,0}, \mathbf{c}_{w,1}, \mathbf{c}_{\mathsf{oa},0}, \mathbf{c}_{\mathsf{oa},1}, \pi_{ct})$. It verifies the validity of π_{ct} as follows.
   - Let $\pi_{ct} = (\{\mathsf{CMT}_{ct,i}\}_{i=1}^\kappa, \mathsf{Ch}_{ct}, \{\mathsf{RSP}_{ct,i}\}_{i=1}^\kappa)$.
   - If $\mathsf{Ch}_{ct} \neq \mathcal{H}_{\mathsf{FS}}(\{\mathsf{CMT}_{ct,i}\}_{i=1}^\kappa, \mathbf{G}_{\mathsf{oa},0}, \mathbf{G}_{\mathsf{oa},1}, \mathbf{c}_{w,0}, \mathbf{c}_{w,1}, \mathbf{c}_{\mathsf{oa},0}, \mathbf{c}_{\mathsf{oa},1}, L)$, return $\perp$. Otherwise, let $\mathsf{Ch}_{ct} = [\mathsf{ch}_{ct,1} | \cdots | \mathsf{ch}_{ct,\kappa}]^\top$.
   - For $i \in [1, \kappa]$, verify the validity of $\mathsf{RSP}_{ct,i}$ with respect to the commitment $\mathsf{CMT}_{ct,i}$ and the challenge $\mathsf{ch}_{ct,i}$. If any of the verifications does not hold, return $\perp$.

2. If the above step does not return 0, it then runs the McElice decryption algorithm $\mathsf{Dec}_{\mathsf{ME}}(\mathsf{sk}_{\mathsf{ME}}^{(j,0)}, \mathbf{c}_{w,0})$ (see Section 3.3), obtaining $\mathbf{w}'$.
3. If $(S, \mathbf{w}') \in R_{\text{permit}}$, return $\mathbf{w}'$. Otherwise, return $\perp$.

$\mathsf{Open}(\mathsf{info}_\tau, \mathsf{sk}_{\mathsf{OA}}, \psi, L)$ This algorithm is run by the OA who possesses the key $\mathsf{sk}_{\mathsf{OA}} = (\mathsf{sk}_{\mathsf{ME}}^{(\mathsf{oa},0)}, \mathsf{sk}_{\mathsf{ME}}^{(\mathsf{oa},1)})$. It proceeds as follows.

1. Parse $\psi = (\mathbf{c}_{w,0}, \mathbf{c}_{w,1}, \mathbf{c}_{\mathsf{oa},0}, \mathbf{c}_{\mathsf{oa},1}, \pi_{ct})$. It verifies π_{ct} as in the algorithm Dec. It π_{ct} is invalid, it returns $\perp$.
2. Otherwise, it runs the decryption algorithm $\mathsf{Dec}_{\mathsf{ME}}(\mathsf{sk}_{\mathsf{ME}}^{(\mathsf{oa},0)}, \mathbf{c}_{\mathsf{oa},0})$, obtaining $[j'_1|\cdots|j'_\ell]^\top$.
3. If info_τ does not include a witness containing the string $[j'_1|\cdots|j'_\ell]^\top$, then return $\perp$.
4. Let $j' \in [0, N-1]$ be the integer that has binary representation $[j'_1|\cdots|j'_\ell]^\top$. Output j'.

6.2 Asymptotic Efficiency, Correctness, and Security

Efficiency. We now analyze the efficiency of our construction with respect to the security parameter λ.

- The public key and secret key of GM have bit size $\mathcal{O}(\lambda)$.
- The public key and secret key of OA and each user have bit size $\mathcal{O}(\lambda^2)$.
- At each epoch, the sender who runs the $\mathcal{P}$ algorithm needs to download data of bit size $\mathcal{O}(\lambda \cdot \ell)$ while the verifier who runs the $\mathcal{V}$ algorithm needs to download data of bit size $\mathcal{O}(\lambda)$.
- The size of ciphertext ψ is $\mathcal{O}(\lambda^2)$. Regarding proof size, it is dominated by that of witness $\mathbf{w}_{\text{GE}}$ as in Section 7.2. The total cost is $\kappa \cdot \mathcal{O}(L_{\mathsf{GE}}) = \omega(\log \lambda) \cdot \mathcal{O}(\lambda^2 + \ell \cdot \lambda)$, where L_{GE} is the size of $\mathbf{w}_{\text{GE}}$.

Correctness. The above FDGE scheme is correct with all but negligible probability. It relies on the following three facts: (a) the correctness of the underlying McEliece encryption scheme and (b) the perfect completeness of the zero-knowledge argument used in the Enc algorithm and (c) the perfect completeness of the zero-knowledge argument used in the $\mathcal{P}$ algorithm. Therefore, in $\mathbf{Expt}_{\mathcal{A}}^{\mathsf{corr}}(1^\lambda)$ defined in Section 2, the $\mathcal{V}$ algorithm will output 1 by fact (c), and the Dec and Open algorithms will output $\mathbf{w}' = \mathbf{w}$ and $\mathsf{pk}' = \mathsf{pk}$, respectively, by fact (a) and (b).

Security. In Theorem 2, we state that the given FDGE satisfies the proposed security requirements in Section 2.2.

Theorem 2. *Assume the zero-knowledge argument used in the* Enc *algorithm is simulation-sound and zero-knowledge, the zero-knowledge argument used in the* $\mathcal{P}$ *algorithm is sound and zero-knowledge, the randomized McEliece encryption schemes have pseudorandom ciphertexts, and the hash function* $h_{\mathbf{B}}$ *is collision resistant. Then, in the random oracle model, the above* FDGE *scheme satisfies message secrecy, anonymity, and soundness.*

Details of the security proofs are given in Section 8.

7 Supporting Zero-Knowledge Arguments for the Code-Based FDGE Scheme

This section presents the zero-knowledge arguments invoked by the sender who encrypts the message and who needs to show its honest behaviour of correct encryption.

7.1 The Zero-Knowledge Argument for the Enc Algorithm

This section presents the zero-knowledge argument for the Enc algorithm, in which we show that $\mathbf{G}_{j,0}, \mathbf{G}_{j,1}$ encrypt the same message $\mathbf{w}$ and that $\mathbf{G}_{\mathsf{oa},0}, \mathbf{G}_{\mathsf{oa},1}$ encrypt $\mathsf{bin}(j)$. This is to ensure the CCA2-security of the encryption schemes. As a result, it is not required to show that $(S, \mathbf{w}) \in R_{\text{permit}}$ or the user encryption key is valid. The public inputs of this protocol are $\mathbf{G}_{\mathsf{oa},0}, \mathbf{G}_{\mathsf{oa},1} \in \mathbb{Z}_2^{n \times k_1}$, $\mathbf{c}_{w,0}$, $\mathbf{c}_{w,1}$, $\mathbf{c}_{\mathsf{oa},0}$, $\mathbf{c}_{\mathsf{oa},1} \in \mathbb{Z}_2^n$, and our goal is to prove knowledge of $\mathbf{r}_{\mathsf{oa},0}, \mathbf{r}_{\mathsf{oa},1} \in \mathbb{Z}_2^{k_1-\ell}$, $\mathbf{e}_{\mathsf{oa},0}, \mathbf{e}_{\mathsf{oa},1} \in \mathsf{B}(n, t_1)$, $\mathsf{bin}(j) \in \mathbb{Z}_2^\ell$, $\mathbf{G}_{j,0}, \mathbf{G}_{j,1} \in \mathbb{Z}_2^{n \times k_2}$, $\mathbf{r}_{w,0}, \mathbf{r}_{w,1} \in \mathbb{Z}_2^{k_2-p}$, and $\mathbf{e}_{w,0}, \mathbf{e}_{w,1} \in \mathsf{B}(n, t_2)$, $\mathbf{w} \in \mathbb{Z}_2^p$ such that the following equations hold.

$$\mathbf{c}_{w,0} = \mathbf{G}_{j,0} \cdot \begin{pmatrix} \mathbf{r}_{w,0} \\ \mathbf{w} \end{pmatrix} \oplus \mathbf{e}_{w,0}, \quad \mathbf{c}_{w,1} = \mathbf{G}_{j,1} \cdot \begin{pmatrix} \mathbf{r}_{w,1} \\ \mathbf{w} \end{pmatrix} \oplus \mathbf{e}_{w,1}; \tag{32}$$

$$\mathbf{c}_{\mathsf{oa},0} = \mathbf{G}_{\mathsf{oa},0} \cdot \begin{pmatrix} \mathbf{r}_{\mathsf{oa},0} \\ \mathsf{bin}(j) \end{pmatrix} \oplus \mathbf{e}_{\mathsf{oa},0}, \quad \mathbf{c}_{\mathsf{oa},1} = \mathbf{G}_{\mathsf{oa},1} \cdot \begin{pmatrix} \mathbf{r}_{\mathsf{oa},1} \\ \mathsf{bin}(j) \end{pmatrix} \oplus \mathbf{e}_{\mathsf{oa},1}. \tag{33}$$

Note that $\mathbf{G}_{j,0}, \mathbf{G}_{j,1}$ should be kept secret since we would like to conceal the identity of the receiver. We aim to transform the above relations to an instance of R_{abstract} such that equivalences in (2) are fulfilled. To this end, we proceed in three steps. We first rearrange each of equations in (32)–(33) to a form of $\mathbf{M} \cdot \mathbf{w}' = \mathbf{v} \bmod 2$ such that $\mathbf{M}, \mathbf{v}$ are constructed from public inputs and $\mathbf{w}'$ is constructed from secret inputs. Specifically, we aim for $\mathbf{w}'$ whose constraints are invariant under random permutations. Next, we combine equations obtained from the first step to a unified form $\mathbf{M}_{\text{ct}} \cdot \mathbf{w}_{\text{ct}} = \mathbf{v}_{\text{ct}} \bmod 2$, where $\mathbf{M}_{\text{ct}}, \mathbf{v}_{\text{ct}}$ are public and $\mathbf{w}_{\text{ct}}$ is a concatenation of the secret vectors obtained in the previous step. Finally, we specify the set $\mathsf{VALID}_{\text{ct}}$ that consists of our secret vector $\mathbf{w}_{\text{ct}}$, the set $\mathcal{S}_{\text{ct}}$, and permutations $\{\Gamma_\phi : \phi \in \mathcal{S}_{\text{ct}}\}$ so that the equivalences in (2) hold.

The First Step-Equation (32). To convert the equations in (32), which incur to prove quadratic statements about LPN relation, we follow the techniques presented in Section 4.1. For $b \in \{0, 1\}$, denote $\mathbf{G}_{j,b} = [\mathbf{G}_{j,b}^{(0)} | \mathbf{G}_{j,b}^{(1)}]$ such that $\mathbf{G}_{j,b}^{(0)} \in \mathbb{Z}_2^{n \times (k_2-p)}$ and $\mathbf{G}_{j,b}^{(1)} \in \mathbb{Z}_2^{n \times p}$. Following from (19), equations in (32) are equivalent to

$$\begin{aligned} c_{w,0} &= \mathbf{G}_{j,0}^{(0)} \cdot \mathbf{r}_{w,0} \oplus \mathbf{G}_{j,0}^{(1)} \cdot \mathbf{w} \oplus \mathbf{e}_{w,0} \\ &= \mathbf{H}_{n,k_2-p} \cdot \mathsf{expand}(\mathbf{G}_{j,0}^{(0)}, \mathbf{r}_{w,0}) \oplus \mathbf{H}_{n,p} \cdot \mathsf{expand}(\mathbf{G}_{j,0}^{(1)}, \mathbf{w}) \oplus \mathbf{e}_{w,0}, \\ c_{w,1} &= \mathbf{H}_{n,k_2-p} \cdot \mathsf{expand}(\mathbf{G}_{j,1}^{(0)}, \mathbf{r}_{w,1}) \oplus \mathbf{H}_{n,p} \cdot \mathsf{expand}(\mathbf{G}_{j,1}^{(1)}, \mathbf{w}) \oplus \mathbf{e}_{w,1}. \end{aligned} \tag{34}$$

To simplify the notation, denote

$$\begin{cases}\mathbf{k}_{j,b}^{(0)} = \mathsf{expand}(\mathbf{G}_{j,b}^{(0)}, \mathbf{r}_{w,b}) \in \mathbb{Z}_2^{4n(k_2-p)} \text{ for } b \in \{0,1\};\\ \mathbf{k}_{j,b}^{(1)} = \mathsf{expand}(\mathbf{G}_{j,b}^{(1)}, \mathbf{w}) \in \mathbb{Z}_2^{4np} \text{ for } b \in \{0,1\}.\end{cases}$$

According to basic algebra, we can form public matrix $\mathbf{M}_{\text{quad}} \in \mathbb{Z}_2^{2n \times L_{\text{quad}}}$ and $\mathbf{v}_{\text{quad}} = (\mathbf{c}_{w,0} \| \mathbf{c}_{w,1}) \in \mathbb{Z}_2^{2n}$ with $L_{\text{quad}} = 8nk_2 + 2n$ such that equations in (34) can be written as

$$\mathbf{M}_{\text{quad}} \cdot \mathbf{w}_{\text{quad}} = \mathbf{v}_{\text{quad}} \bmod 2, \tag{35}$$

where $\mathbf{w}_{\text{quad}} \in \mathbb{Z}_2^{L_{\text{quad}}}$ is of the following form:

$$\mathbf{w}_{\text{quad}} = (\mathbf{k}_{j,0}^{(0)} \| \mathbf{k}_{j,0}^{(1)} \| \mathbf{k}_{j,1}^{(0)} \| \mathbf{k}_{j,1}^{(1)} \| \mathbf{e}_{w,0} \| \mathbf{e}_{w,1}).$$

The First Step-Equation (33). Next, to transform equations in (33) which simply require proving linear statements, we utilize the techniques in Section 3.5. Let $\mathsf{bin}(j) = [j_1 | \cdots | j_\ell]^\top$, $\mathbf{f} = \mathsf{Encode}(\mathsf{bin}(j)) \in \mathbb{Z}_2^{2\ell}$, and $\mathbf{h}_{\mathsf{oa},b} = \mathsf{Encode}(\mathbf{r}_{\mathsf{oa},b}) \in \mathbb{Z}_2^{2(k_1-\ell)}$ for $b \in \{0,1\}$. Following (4), equations in (33) are equivalent to

$$\mathbf{c}_{\mathsf{oa},0} = \mathbf{G}_{\mathsf{oa},0} \cdot \mathbf{I}_{k_1}^* \cdot \begin{pmatrix}\mathbf{h}_{\mathsf{oa},0}\\ \mathbf{f}\end{pmatrix} \oplus \mathbf{e}_{\mathsf{oa},0},\ \mathbf{c}_{\mathsf{oa},1} = \mathbf{G}_{\mathsf{oa},1} \cdot \mathbf{I}_{k_1}^* \cdot \begin{pmatrix}\mathbf{h}_{\mathsf{oa},1}\\ \mathbf{f}\end{pmatrix} \oplus \mathbf{e}_{\mathsf{oa},1}. \tag{36}$$

We further transform equations in (36) to the following equivalent form

$$\mathbf{M}_{\text{oa}} \cdot \mathbf{w}_{\text{oa}} = \mathbf{v}_{\text{oa}} \bmod 2, \tag{37}$$

where $\mathbf{M}_{\text{oa}} \in \mathbb{Z}_2^{2n \times L_{\text{oa}}}$ and $\mathbf{v}_{\text{oa}} = (\mathbf{c}_{\mathsf{oa},0} \| \mathbf{c}_{\mathsf{oa},1}) \in \mathbb{Z}_2^{2n}$ are public, and $\mathbf{w}_{\text{oa}} \in \mathbb{Z}_2^{L_{\text{oa}}}$ is secret with $L_{\text{oa}} = 4k_1 - 2\ell + 2n$ and $\mathbf{w}_{\text{oa}}$ being the following form:

$$\mathbf{w}_{\text{oa}} = (\mathbf{h}_{\mathsf{oa},0} \| \mathbf{h}_{\mathsf{oa},1} \| \mathbf{f} \| \mathbf{e}_{\mathsf{oa},0} \| \mathbf{e}_{\mathsf{oa},1}).$$

The Second Step. We now combine the equations (35) and (37) into a unified form. To this end, denote

$$\mathbf{M}_{\text{ct}} = \begin{pmatrix}\mathbf{M}_{\text{quad}} & \\ & \mathbf{M}_{\text{oa}}\end{pmatrix} \in \mathbb{Z}_2^{4n \times L_{\text{ct}}}; \mathbf{w}_{\text{ct}} = (\mathbf{w}_{\text{quad}} \| \mathbf{w}_{\text{oa}}) \in \mathbb{Z}_2^{L_{\text{ct}}};$$
$$\mathbf{v}_{\text{ct}} = (\mathbf{v}_{\text{quad}} \| \mathbf{v}_{\text{oa}}) \in \mathbb{Z}_2^{4n}; L_{\text{ct}} = L_{\text{quad}} + L_{\text{oa}} = 8nk_2 + 4n + 4k_1 - 2\ell.$$

Then $\mathbf{M}_{\text{ct}} \cdot \mathbf{w}_{\text{ct}} = \mathbf{v}_{\text{ct}} \bmod 2$.

The Third Step. We now specify the set $\mathsf{VALID}_{\text{ct}}, \mathcal{S}_{\text{ct}}$ and the associated permutations $\{\Gamma_\phi : \phi \in \mathcal{S}_{\text{ct}}\}$. Let $\mathsf{VALID}_{\text{ct}}$ be the set that contains vector

$$\widehat{\mathbf{w}}_{\text{ct}} = \big(\widehat{\mathbf{k}}_{j,0}^{(0)} \| \widehat{\mathbf{k}}_{j,0}^{(1)} \| \widehat{\mathbf{k}}_{j,1}^{(0)} \| \widehat{\mathbf{k}}_{j,1}^{(1)} \| \widehat{\mathbf{e}}_{w,0} \| \widehat{\mathbf{e}}_{w,1} \| \widehat{\mathbf{h}}_{\mathsf{oa},0} \| \widehat{\mathbf{h}}_{\mathsf{oa},1} \| \widehat{\mathbf{f}} \| \widehat{\mathbf{e}}_{\mathsf{oa},0} \| \widehat{\mathbf{e}}_{\mathsf{oa},1}\big) \in \mathbb{Z}_2^{L_{\text{ct}}} \tag{38}$$

such that

(a) There exists $\widehat{\mathbf{w}} \in \mathbb{Z}_2^p$, $\widehat{\mathbf{e}}_{w,0}, \widehat{\mathbf{e}}_{w,1} \in \mathsf{B}(n, t_2)$, $\widehat{\mathbf{e}}_{\mathsf{oa},0}, \widehat{\mathbf{e}}_{\mathsf{oa},1} \in \mathsf{B}(n, t_1)$.

(b) For $b \in \{0,1\}$, there exists $\widehat{\mathbf{r}}_{w,b} \in \mathbb{Z}_2^{k_2-p}$, $\widehat{\mathbf{p}}_b^{(0)} \in \mathbb{Z}_2^{n(k_2-p)}$, $\widehat{\mathbf{p}}_b^{(1)} \in \mathbb{Z}_2^{np}$ such that

$$\widehat{\mathbf{k}}_{j,b}^{(0)} = \mathsf{expand}(\widehat{\mathbf{p}}_b^{(0)}, \widehat{\mathbf{r}}_{w,b}) \in \mathbb{Z}_2^{4n(k_2-p)}, \text{ and } \widehat{\mathbf{k}}_{j,b}^{(1)} = \mathsf{expand}(\widehat{\mathbf{p}}_b^{(1)}, \widehat{\mathbf{w}}) \in \mathbb{Z}_2^{4np}.$$

(c) For $b \in \{0,1\}$, there exits $\widehat{\mathbf{r}}_{\mathsf{oa},b} \in \mathbb{Z}_2^{k_1-\ell}$ such that $\widehat{\mathbf{h}}_{\mathsf{oa},b} = \mathsf{Encode}(\widehat{\mathbf{r}}_{\mathsf{oa},b}) \in \mathbb{Z}_2^{2(k_1-\ell)}$.

(d) There exists $\widehat{\mathbf{j}} = [\widehat{j}_i | \cdots | \widehat{j}_\ell]^\top \in \mathbb{Z}_2^\ell$ such that $\widehat{\mathbf{f}} = \mathsf{Encode}(\widehat{\mathbf{j}}) \in \mathbb{Z}_2^{2\ell}$.

One can check that our secret vector $\mathbf{w}_{\mathrm{ct}}$ belongs to $\mathsf{VALID}_{\mathrm{ct}}$. Let $\mathcal{S}_{\mathrm{ct}} = \mathbb{Z}_2^p \times (\mathsf{S}_n)^4 \times (\mathbb{Z}_2^{k_2-p})^2 \times (\mathbb{Z}_2^{n(k_2-p)})^2 \times (\mathbb{Z}_2^{np})^2 \times (\mathbb{Z}_2^{k_1-\ell})^2 \times \mathbb{Z}_2^\ell$. Then for each $\phi = (\mathbf{t}, \sigma_{w,0}, \sigma_{w,1}, \sigma_{\mathsf{oa},0}, \sigma_{\mathsf{oa},1}, \mathbf{t}_{w,0}, \mathbf{t}_{w,1}, \mathbf{t}_0^{(0)}, \mathbf{t}_1^{(0)}, \mathbf{t}_0^{(1)}, \mathbf{t}_1^{(1)}, \mathbf{t}_{\mathsf{oa},0}, \mathbf{t}_{\mathsf{oa},1}, \mathbf{t}_f) \in \mathcal{S}_{\mathrm{ct}}$, define a permutation that transforms $\widehat{\mathbf{w}}_{\mathrm{ct}}$ of form (38) to a vector $\Gamma_\phi(\widehat{\mathbf{w}}_{\mathrm{ct}})$ of the following form

$$\Gamma_\phi(\widehat{\mathbf{w}}_{\mathrm{ct}}) = \big(\mathbf{k}_{j,0}^{(0)\star} \| \mathbf{k}_{j,0}^{(1)\star} \| \mathbf{k}_{j,1}^{(0)\star} \| \mathbf{k}_{j,1}^{(1)\star} \| \mathbf{e}_{w,0}^\star \| \mathbf{e}_{w,1}^\star \| \mathbf{h}_{\mathsf{oa},0}^\star \| \mathbf{h}_{\mathsf{oa},1}^\star \| \mathbf{f}^\star \| \mathbf{e}_{\mathsf{oa},0}^\star \| \mathbf{e}_{\mathsf{oa},1}^\star\big)$$

such that

(a) For $b \in \{0,1\}$, $\mathbf{e}_{w,b}^\star = \sigma_{w,b}(\widehat{\mathbf{e}}_{w,b})$ and $\mathbf{e}_{\mathsf{oa},b}^\star = \sigma_{\mathsf{oa},b}(\widehat{\mathbf{e}}_{\mathsf{oa},b})$.

(b) For $b \in \{0,1\}$, $\mathbf{k}_{j,b}^{(0)\star} = T'_{\mathbf{t}_b^{(0)}, \mathbf{t}_{w,b}}(\widehat{\mathbf{k}}_{j,b}^{(0)})$ and $\mathbf{k}_{j,b}^{(1)\star} = T'_{\mathbf{t}_b^{(1)}, \mathbf{t}}(\widehat{\mathbf{k}}_{j,b}^{(1)})$.

(c) For $b \in \{0,1\}$, $\mathbf{h}_{\mathsf{oa},b}^\star = F_{\mathbf{t}_{\mathsf{oa},b}}(\widehat{\mathbf{h}}_{\mathsf{oa},b})$.

(d) $\mathbf{f}^\star = F_{\mathbf{t}_f}(\widehat{\mathbf{f}})$.

It is implied by the equivalences observed in (3), (5) and (18) that, for all $\phi \in \mathcal{S}_{\mathrm{ct}}$, $\widehat{\mathbf{w}}_{\mathrm{ct}} \in \mathsf{VALID}_{\mathrm{ct}} \iff \Gamma_\phi(\widehat{\mathbf{w}}_{\mathrm{ct}}) \in \mathsf{VALID}_{\mathrm{ct}}$. Moreover, if ϕ is uniformly random in $\mathcal{S}_{\mathrm{ct}}$ and $\widehat{\mathbf{w}}_{\mathrm{ct}} \in \mathsf{VALID}_{\mathrm{ct}}$, then $\Gamma_\phi(\widehat{\mathbf{w}}_{\mathrm{ct}})$ is uniformly random in $\mathsf{VALID}_{\mathrm{ct}}$. Therefore, we have successfully reduced the considered statement to an instance of R_{abstract}. Now $\mathcal{P}$ and $\mathcal{V}$ interacts as in Figure 2. The protocol is a statistical ZKAoK with perfect completeness, soundness error 2/3, and communication cost $\mathcal{O}(L_{\mathrm{ct}}) = \mathcal{O}(\lambda^2)$ bits.

7.2 The Zero-Knowledge Argument for the $\mathcal{P}$ Algorithm

In this section, we present the zero-knowledge protocol that will be invoked by the sender who has encrypted a message $\mathbf{w}$ to a user j in the group at epoch τ and now has to show its honest behavior. This protocol is an extension of the one in Section 7.1, for which the prover has to show additionally the following three facts.

(I) The secret message $\mathbf{w} \in \mathbb{Z}_2^p$ is such that $(S, \mathbf{w}) \in R_{\mathrm{permit}}$.

(II) The secret hash value $\mathbf{d}_j \in \mathbb{Z}_2^n \backslash \{\mathbf{0}^n\}$ is properly accumulated into the root $\mathbf{u}_\tau$ of the (main)-Merkle tree. Equivalently, there exists witness $w^{(j)} \in \mathbb{Z}_2^\ell \times (\mathbb{Z}_2^n)^\ell$ such that

$$\mathsf{Verify}_{\mathbf{B}}(\mathbf{u}_\tau, \mathbf{d}_j, w^{(j)}) = 1. \tag{39}$$

(III) The user key $(\mathbf{G}_{j,0}, \mathbf{G}_{j,1})$, which should be kept hidden, is honestly hashed to the (non-zero) root $\mathbf{d}_j$ of a (sub)-Merkle tree. In other words,

$$\mathsf{Accu}_{\mathbf{B}}\big((\mathbf{G}_{j,0}, \mathbf{G}_{j,1})\big) = \mathbf{d}_j. \tag{40}$$

Following the same strategy in Section 7.1, we proceed in three steps. we first rearrange each statement in case (I) – (III) to a form of $\mathbf{M} \cdot \mathbf{w}' = \mathbf{v} \bmod 2$ such that the constraints of secret vector $\mathbf{w}'$ are invariant under some permutations.

The First Step-Case (I). This relation has been proved in Section 5.2, in which we obtain an equivalent form

$$\mathbf{M}_{\text{permit}} \cdot \mathbf{w}_{\text{permit}} = \mathbf{0}^n \bmod 2, \tag{41}$$

where $\mathbf{M}_{\text{permit}} \in \mathbb{Z}_2^{n \times L_{\text{permit}}}$, $\mathbf{w}_{\text{permit}} = (\mathsf{expand}'(\mathsf{permit}(\mathbf{w}), \mathbf{g}) \| \mathbf{h}) \in \mathbb{Z}_2^{L_{\text{permit}}}$ with $\mathbf{g} \in \mathsf{B}(p-t+1, 1)$, $\mathbf{h} \in \mathsf{B}(k, 1)$, and $L_{\text{permit}} = 2t(p-t+1)+k$.

The First Step-Case (II). In this equation, we have to prove knowledge of $(\mathbf{d}_j, w^{(j)})$. This is indeed what we have done in Section 3.6. Let $w^{(j)} = \big(\lfloor j_1 | \cdots | j_\ell \rfloor^\top, \mathbf{w}_\ell, \ldots, \mathbf{w}_1\big)$ and $\mathbf{v}_\ell = \mathbf{d}_j, \mathbf{v}_{\ell-1}, \ldots, \mathbf{v}_1$ be computed from the Verify algorithm as explained in Section 3.1. Following Section 3.6, equation (39) is equivalent to

$$\mathbf{M}_{\text{acc}} \cdot \mathbf{w}_{\text{acc}} = \mathbf{v}_{\text{acc}} \bmod 2, \tag{42}$$

where $\mathbf{M}_{\text{acc}} \in \mathbb{Z}_2^{\ell n \times L_{\text{acc}}}$ and $\mathbf{v}_{\text{acc}} \in \mathbb{Z}_2^{\ell n}$ are public and $\mathbf{w}_{\text{acc}} \in \{0,1\}^{L_{\text{acc}}}$ is secret with $L_{\text{acc}} = 2\ell m + 2(\ell-1)n$. Concretely, $\mathbf{w}_{\text{acc}} \in \{0,1\}^{L_{\text{acc}}}$ is of the form

$$\mathbf{w}_{\text{acc}} = (\mathbf{y}_1 \| \cdots \| \mathbf{y}_\ell \| \mathbf{z}_1 \| \cdots \| \mathbf{z}_\ell \| \mathbf{x}_1 \| \cdots \| \mathbf{x}_{\ell-1})$$

such that

$$\begin{cases} \mathbf{y}_i = \mathsf{Ext}(j_i, \mathsf{RE}(\mathbf{v}_i)) \text{ for } i \in [1, \ell]; \\ \mathbf{z}_i = \mathsf{Ext}(\overline{j_i}, \mathsf{RE}(\mathbf{w}_i)) \text{ for } i \in [1, \ell]; \\ \mathbf{x}_i = \mathsf{Encode}(\mathbf{v}_i) \text{ for } i \in [1, \ell-1]. \end{cases}$$

The First Step-Case (III). Now let us look at equation (40), in which we have to prove knowledge of $(\mathbf{G}_{j,0}, \mathbf{G}_{j,1}, \mathbf{d}_j)$. Recall that for $b \in \{0,1\}$, $\mathbf{G}_{j,b} = [\mathbf{g}_{j,k_2 b+0} | \cdots | \mathbf{g}_{j,k_2 b+k_2-1}]$, $D_j = \{\mathbf{g}_{j,0} \ldots, \mathbf{g}_{j,2k_2-1}\}$, and $2k_2 = 2^v$. Intuitively, this would require proving knowledge of $(\mathbf{d}_j, \mathbf{g}_{j,i}, w^{'(i)}) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n \times \big(\mathbb{Z}_2^v \times (\mathbb{Z}_2^n)^v\big)$ such that $\mathsf{Verify}_{\mathbf{B}}(\mathbf{d}_j, \mathbf{g}_{j,i}, w^{'(i)}) = 1$ for each $i \in [0, 2k_2-1]$. However, this would incur $2k_2 \cdot \log(2k_2) = 2k_2 v$ equations. We show that, however, $2k_2 - 1$[12] equations suffice. Recall that $\mathbf{d}_j = \mathbf{v}_\ell \in \{0,1\}^n \setminus \{\mathbf{0}^n\}$ is computed bottom-up starting from the set D_j. Rewrite $D_j = \{\mathbf{p}_{v,0}^j, \mathbf{p}_{v,1}^j, \ldots, \mathbf{p}_{v,2^v-1}^j\}$. Then equation (40) is

[12] It is actually $2k_2$ since we add one more equation to handle $\mathbf{v}_\ell$.

equivalent to

$$\begin{cases} \mathbf{B}_0 \cdot \mathsf{RE}(\mathbf{p}^j_{1,0}) \oplus \mathbf{B}_1 \cdot \mathsf{RE}(\mathbf{p}^j_{1,1}) \oplus \mathbf{v}_\ell = \mathbf{0}^n; \\ \mathbf{B}_0 \cdot \mathsf{RE}(\mathbf{p}^j_{2,2i}) \oplus \mathbf{B}_1 \cdot \mathsf{RE}(\mathbf{p}^j_{2,2i+1}) \oplus \mathbf{p}^j_{1,i} = \mathbf{0}^n, \ \forall\ i \in [0, 2^1 - 1]; \\ \qquad\qquad\qquad\qquad \cdots\cdots \\ \mathbf{B}_0 \cdot \mathsf{RE}(\mathbf{p}^j_{v-1,2i}) \oplus \mathbf{B}_1 \cdot \mathsf{RE}(\mathbf{p}^j_{v-1,2i+1}) \oplus \mathbf{p}^j_{v-2,i} = \mathbf{0}^n, \ \forall\ i \in [0, 2^{v-2} - 1]; \\ \mathbf{B}_0 \cdot \mathsf{RE}(\mathbf{p}^j_{v,2i}) \oplus \mathbf{B}_1 \cdot \mathsf{RE}(\mathbf{p}^j_{v,2i+1}) \oplus \mathbf{p}^j_{v-1,i} = \mathbf{0}^n, \ \forall\ i \in [0, 2^{v-1} - 1]; \end{cases} \tag{43}$$

in which we have to prove knowledge of

$$\big(\mathbf{v}_\ell, \mathbf{p}^j_{1,0}, \mathbf{p}^j_{1,1}, \mathbf{p}^j_{2,0}, \ldots, \mathbf{p}^j_{2,3}, \ldots, \mathbf{p}^j_{v-1,0}, \ldots, \mathbf{p}^j_{v-1,2^{v-1}-1}, \mathbf{p}^j_{v,0}, \ldots, \mathbf{p}^j_{v,2^v-1}\big).$$

To transform all the secret vectors in (43) so that their constraints are invariant, we apply the Encode function to the following secret input

$$\big(\mathbf{p}^j_{1,0}, \mathbf{p}^j_{1,1}, \mathbf{p}^j_{2,0}, \ldots, \mathbf{p}^j_{2,3}, \ldots, \mathbf{p}^j_{v-1,0}, \ldots, \mathbf{p}^j_{v-1,2^{v-1}-1}\big).$$

For $u \in [1, v-1]$ and $i \in [0, 2^u - 1]$, let $\mathbf{x}^j_{u,i} = \mathsf{Encode}(\mathbf{p}^j_{u,i}) \in \mathbb{Z}_2^{2n}$, which implies $\mathbf{p}^j_{u,i} = \mathbf{I}^*_n \cdot \mathbf{x}^j_{u,i}$. For ease of notation, we also let $\mathbf{q}^j_{u,i} = \mathsf{RE}(\mathbf{p}^j_{u,i}) \in \mathbb{Z}_2^{m/2}$ for $u \in [1, v]$ and $i \in [0, 2^u - 1]$.

In terms of $\mathbf{v}_\ell$, we cannot simply apply Encode function since the permutation F applied to $\mathsf{Encode}(\mathbf{v}_\ell)$ does not preserve the constraint that $\mathbf{v}_\ell$ is non-zero. However, we cannot simply extend it to $\mathbf{v}^*_\ell \in \mathsf{B}(2n-1, n)$ as well since $\mathbf{v}_\ell$ appears in equation (42) in the form of $\mathsf{RE}(\mathbf{v}_\ell)$. Note that the constraint of $\mathbf{v}^*_\ell$ is invariant for a permutation $\pi \in \mathsf{S}_{2n-1}$ while that of $\mathsf{RE}(\mathbf{v}_\ell)$ is invariant for a permutation $E'_{\mathbf{b}_\ell}$ for some $\mathbf{b}_\ell \in \{0,1\}^n$. To solve this issue, we come up with the following novel yet simple approach. We apply Encode function to $\mathbf{v}_\ell$ to obtain $\mathbf{x}_\ell$ in the first equation of (43) and add one more equation $\mathbf{B}_0 \cdot \mathsf{RE}(\mathbf{p}^j_{1,0}) \oplus \mathbf{B}_1 \cdot \mathsf{RE}(\mathbf{p}^j_{1,1}) \oplus \mathbf{v}_{\ell+1} = \mathbf{0}^n$ and extend $\mathbf{v}_{\ell+1}$ to $\mathbf{v}^*_{\ell+1} \in \mathsf{B}(2n-1, n)$. It is obvious that $\mathbf{v}_\ell = \mathbf{v}_{\ell+1}$. With this additional equation (and hence a slightly larger cost), we are able to prove that $\mathbf{v}_\ell$ satisfies multiple constraints. Note that we have $\mathbf{v}_\ell = \mathbf{I}^*_n \cdot \mathbf{x}_\ell$ and $\mathbf{v}_{\ell+1} = \mathbf{P}^*_n \cdot \mathbf{v}^*_{\ell+1}$.

To this end, we obtain equations below that are equivalent to those in (43).

$$\begin{cases} \mathbf{B}_0 \cdot \mathbf{q}^j_{1,0} \oplus \mathbf{B}_1 \cdot \mathbf{q}^j_{1,1} \oplus \mathbf{I}^*_n \cdot \mathbf{x}_\ell = \mathbf{0}^n; \\ \mathbf{B}_0 \cdot \mathbf{q}^j_{1,0} \oplus \mathbf{B}_1 \cdot \mathbf{q}^j_{1,1} \oplus \mathbf{P}^*_n \cdot \mathbf{v}^*_{\ell+1} = \mathbf{0}^n; \\ \mathbf{B}_0 \cdot \mathbf{q}^j_{2,2i} \oplus \mathbf{B}_1 \cdot \mathbf{q}^j_{2,2i+1} \oplus \mathbf{I}^*_n \cdot \mathbf{x}^j_{1,i} = \mathbf{0}^n, \ \forall\ i \in [0, 2^1 - 1]; \\ \qquad\qquad\qquad\qquad \cdots\cdots \\ \mathbf{B}_0 \cdot \mathbf{q}^j_{v-1,2i} \oplus \mathbf{B}_1 \cdot \mathbf{q}^j_{v-1,2i+1} \oplus \mathbf{I}^*_n \cdot \mathbf{x}^j_{v-2,i} = \mathbf{0}^n, \ \forall\ i \in [0, 2^{v-2} - 1]; \\ \mathbf{B}_0 \cdot \mathbf{q}^j_{v,2i} \oplus \mathbf{B}_1 \cdot \mathbf{q}^j_{v,2i+1} \oplus \mathbf{I}^*_n \cdot \mathbf{x}^j_{v-1,i} = \mathbf{0}^n, \ \forall\ i \in [0, 2^{v-1} - 1]. \end{cases} \tag{44}$$

Through careful algebra transformation, equations in (44) are equivalent to

$$\mathbf{M}_{\text{vacc}} \cdot \mathbf{w}_{\text{vacc}} = \mathbf{v}_{\text{vacc}} \bmod 2, \tag{45}$$

for some properly formed public matrix $\mathbf{M}_{\mathrm{vacc}} \in \mathbb{Z}_2^{2k_2 n \times L_{\mathrm{vacc}}}$ and $\mathbf{v}_{\mathrm{vacc}} = \mathbf{0}^{2k_2 n}$ with $L_{\mathrm{vacc}} = (2n-1) + (2^v - 1)\cdot 2n + (2^{v+1}-2)\cdot m/2 = 2k_2 m + 4k_2 n - m - 1$ and $\mathbf{w}_{\mathrm{vacc}}$ being the following form:

$$\begin{aligned}\mathbf{w}_{\mathrm{vacc}} = \big(\mathbf{x}_\ell \| \mathbf{v}^*_{\ell+1} \| \mathbf{x}^j_{1,0} \| \mathbf{x}^j_{1,1} \| \mathbf{x}^j_{2,0} \| \cdots \| \mathbf{x}^j_{2,3} \| \cdots \| \mathbf{x}^j_{v-1,0} \| \cdots \| \mathbf{x}^j_{v-1,2^{v-1}-1} \| \\ \mathbf{q}^j_{1,0} \| \mathbf{q}^j_{1,1} \| \mathbf{q}^j_{2,0} \| \cdots \| \mathbf{q}^j_{2,3} \| \cdots \| \mathbf{q}^j_{v-1,0} \| \cdots \| \mathbf{q}^j_{v-1,2^{v-1}-1} \| \\ \mathbf{q}^j_{v,0} \| \mathbf{q}^j_{v,1} \| \cdots \| \mathbf{q}^j_{v,2^v-1} \big).\end{aligned}$$

The Second Step. Following Section 7.1, we combine the equations (41), (42), (45) together with (35), (37) into a unified form. To this end, denote

$$\mathbf{M}_{\mathrm{GE}} = \begin{pmatrix} \mathbf{M}_{\mathrm{permit}} & & & & \\ & \mathbf{M}_{\mathrm{acc}} & & & \\ & & \mathbf{M}_{\mathrm{vacc}} & & \\ & & & \mathbf{M}_{\mathrm{quad}} & \\ & & & & \mathbf{M}_{\mathrm{oa}} \end{pmatrix} \in \mathbb{Z}_2^{(2k_2+\ell+5)n \times L_{\mathrm{GE}}};$$

$$\begin{aligned}\mathbf{w}_{\mathrm{GE}} &= (\mathbf{w}_{\mathrm{permit}} \| \mathbf{w}_{\mathrm{acc}} \| \mathbf{w}_{\mathrm{vacc}} \| \mathbf{w}_{\mathrm{quad}} \| \mathbf{w}_{\mathrm{oa}}) \in \mathbb{Z}_2^{L_{\mathrm{GE}}}; \\ \mathbf{v}_{\mathrm{GE}} &= (\mathbf{0}^n \| \mathbf{v}_{\mathrm{acc}} \| \mathbf{v}_{\mathrm{vacc}} \| \mathbf{v}_{\mathrm{quad}} \| \mathbf{v}_{\mathrm{oa}}) \in \mathbb{Z}_2^{(2k_2+\ell+5)n}; \\ L_{\mathrm{GE}} &= L_{\mathrm{permit}} + L_{\mathrm{acc}} + L_{\mathrm{vacc}} + L_{\mathrm{quad}} + L_{\mathrm{oa}}; \\ &= 2t(p-t+1) + k + 2k_2 m + 12k_2 n + 2\ell(m+n) + 4k_1 + 2n - m - 2\ell - 1.\end{aligned}$$

Then $\mathbf{M}_{\mathrm{GE}} \cdot \mathbf{w}_{\mathrm{GE}} = \mathbf{v}_{\mathrm{GE}} \bmod 2$.

The Third Step. Lastly, we specify the set $\mathsf{VALID}_{\mathrm{GE}}$, $\mathcal{S}_{\mathrm{GE}}$ and the associated permutations $\{\Gamma_\phi : \phi \in \mathcal{S}_{\mathrm{GE}}\}$. Let $\mathsf{VALID}_{\mathrm{GE}}$ be the set that consists of vector $\widehat{\mathbf{w}}_{\mathrm{GE}} \in \mathbb{Z}_2^{L_{\mathrm{GE}}}$ of the following form

$$\begin{aligned}\widehat{\mathbf{w}}_{\mathrm{GE}} = \big(\widehat{\mathbf{z}} \| \widehat{\mathbf{h}} \| \widehat{\mathbf{y}}_1 \| \cdots \| \widehat{\mathbf{y}}_\ell \| \widehat{\mathbf{z}}_1 \| \cdots \| \widehat{\mathbf{z}}_\ell \| \widehat{\mathbf{x}}_1 \| \cdots \| \widehat{\mathbf{x}}_\ell \| \widehat{\mathbf{v}}^*_{\ell+1} \| \\ \widehat{\mathbf{x}}^j_{1,0} \| \widehat{\mathbf{x}}^j_{1,1} \| \widehat{\mathbf{x}}^j_{2,0} \| \cdots \| \widehat{\mathbf{x}}^j_{2,3} \| \cdots \| \widehat{\mathbf{x}}^j_{v-1,0} \| \cdots \| \widehat{\mathbf{x}}^j_{v-1,2^{v-1}-1} \| \\ \widehat{\mathbf{q}}^j_{1,0} \| \widehat{\mathbf{q}}^j_{1,1} \| \widehat{\mathbf{q}}^j_{2,0} \| \cdots \| \widehat{\mathbf{q}}^j_{2,3} \| \cdots \| \widehat{\mathbf{q}}^j_{v-1,0} \| \cdots \| \widehat{\mathbf{q}}^j_{v-1,2^{v-1}-1} \| \\ \widehat{\mathbf{q}}^j_{v,0} \| \widehat{\mathbf{q}}^j_{v,1} \| \cdots \| \widehat{\mathbf{q}}^j_{v,2^v-1} \| \\ \widehat{\mathbf{k}}^{(0)}_{j,0} \| \widehat{\mathbf{k}}^{(1)}_{j,0} \| \widehat{\mathbf{k}}^{(0)}_{j,1} \| \widehat{\mathbf{k}}^{(1)}_{j,1} \| \\ \widehat{\mathbf{e}}_{w,0} \| \widehat{\mathbf{e}}_{w,1} \| \widehat{\mathbf{h}}_{\mathsf{oa},0} \| \widehat{\mathbf{h}}_{\mathsf{oa},1} \| \widehat{\mathbf{f}} \| \widehat{\mathbf{e}}_{\mathsf{oa},0} \| \widehat{\mathbf{e}}_{\mathsf{oa},1} \big)\end{aligned} \tag{46}$$

such that the following conditions are satisfied.

(i) $\widehat{\mathbf{z}} = \mathsf{expand}'(\widehat{\sigma}^{(t)}(\mathsf{permit}(\widehat{\mathbf{w}})), \widehat{\mathbf{g}}) \in \mathbb{Z}_2^{2t(p-t+1)}$ for some $\widehat{\sigma} \in \mathsf{S}_{p-t+1}$, $\widehat{\mathbf{w}} \in \mathbb{Z}_2^p$, and $\widehat{\mathbf{g}} \in \mathsf{B}(p-t+1, 1)$. $\widehat{\mathbf{h}} \in \mathsf{B}(k, 1)$.

(ii) $\widehat{\mathbf{v}}^*_{\ell+1} \in \mathsf{B}(2n-1, n)$, $\widehat{\mathbf{e}}_{w,0}, \widehat{\mathbf{e}}_{w,1} \in \mathsf{B}(n, t_2)$, $\widehat{\mathbf{e}}_{\mathsf{oa},0}, \widehat{\mathbf{e}}_{\mathsf{oa},1} \in \mathsf{B}(n, t_1)$.

(iii) For $i \in [1, \ell]$, there exists $\widehat{j_i} \in \{0, 1\}$, $\widehat{\mathbf{v}}_i$, $\widehat{\mathbf{w}}_i \in \mathbb{Z}_2^n$ such that

$$\widehat{\mathbf{y}}_i = \mathsf{Ext}(\widehat{j_i}, \mathsf{RE}(\widehat{\mathbf{v}}_i)),\ \widehat{\mathbf{z}}_i = \mathsf{Ext}(\overline{\widehat{j_i}}, \mathsf{RE}(\widehat{\mathbf{w}}_i)) \in \mathbb{Z}_2^m,\ \text{and } \widehat{\mathbf{x}}_i = \mathsf{Encode}(\widehat{\mathbf{v}}_i) \in \mathbb{Z}_2^{2n}.$$

(iv) For all $u \in [1, v-1], i \in [0, 2^u - 1]$, there exists $\widehat{\mathbf{p}}^j_{u,i} \in \mathbb{Z}_2^n$ such that

$$\widehat{\mathbf{x}}^j_{u,i} = \mathsf{Encode}(\widehat{\mathbf{p}}^j_{u,i}) \in \mathbb{Z}_2^{2n}, \text{ and } \widehat{\mathbf{q}}^j_{u,i} = \mathsf{RE}(\widehat{\mathbf{p}}^j_{u,i}) \in \mathbb{Z}_2^{m/2}.$$

(v) For $i \in [0, 2^v - 1]$, there exists $\widehat{\mathbf{p}}^j_{v,i} \in \mathbb{Z}_2^n$ such that $\widehat{\mathbf{q}}^j_{v,i} = \mathsf{RE}(\widehat{\mathbf{p}}^j_{v,i}) \in \mathbb{Z}_2^{m/2}$.

(vi) Denote

$$\widehat{\mathbf{p}}^{(0)}_0 = (\widehat{\mathbf{p}}^j_{v,0} \| \cdots \| \widehat{\mathbf{p}}^j_{v,k_2-p-1}), \quad \widehat{\mathbf{p}}^{(1)}_0 = (\widehat{\mathbf{p}}^j_{v,k_2-p} \| \cdots \| \widehat{\mathbf{p}}^j_{v,k_2-1}),$$
$$\widehat{\mathbf{p}}^{(0)}_1 = (\widehat{\mathbf{p}}^j_{v,k_2} \| \cdots \| \widehat{\mathbf{p}}^j_{v,2k_2-p-1}), \quad \widehat{\mathbf{p}}^{(1)}_1 = (\widehat{\mathbf{p}}^j_{v,2k_2-p} \| \cdots \| \widehat{\mathbf{p}}^j_{v,2k_2-1}).$$

For $b \in \{0,1\}$, there exists $\widehat{\mathbf{r}}_{w,b} \in \mathbb{Z}_2^{k_2-p}$ such that

$$\widehat{\mathbf{k}}^{(0)}_{j,b} = \mathsf{expand}(\widehat{\mathbf{p}}^{(0)}_b, \widehat{\mathbf{r}}_{w,b}) \in \mathbb{Z}_2^{4n(k_2-p)}, \text{ and } \widehat{\mathbf{k}}^{(1)}_{j,b} = \mathsf{expand}(\widehat{\mathbf{p}}^{(1)}_b, \widehat{\mathbf{w}}) \in \mathbb{Z}_2^{4np}.$$

(vii) For $b \in \{0,1\}$, there exits $\widehat{\mathbf{r}}_{\mathsf{oa},b} \in \mathbb{Z}_2^{k_1-\ell}$ such that $\widehat{\mathbf{h}}_{\mathsf{oa},b} = \mathsf{Encode}(\widehat{\mathbf{r}}_{\mathsf{oa},b})$.

(viii) Denote $\widehat{\mathbf{j}} = [\widehat{j_i} | \cdots | \widehat{j_\ell}]^\top$. Then $\widehat{\mathbf{f}} = \mathsf{Encode}(\widehat{\mathbf{j}}) \in \mathbb{Z}_2^\ell$.

Remark 1. If a variable appears multiple times, it is critical that we employ the same permutation for those places. To enhance the understanding, we mark those places in the same color. For example, $\widehat{\mathbf{v}}_i$ appears both in variables $\widehat{\mathbf{y}}_i$ and $\widehat{\mathbf{x}}_i$, we mark those two places in blue.

It is verifiable that our secret vector $\mathbf{w}_{\mathrm{GE}}$ is included in the set $\mathsf{VALID}_{\mathrm{GE}}$. Let

$$\mathcal{S}_{\mathrm{GE}} = \mathbb{Z}_2^p \times \mathsf{S}_{p-t+1} \times \mathsf{S}_k \times \mathsf{S}_{2n-1} \times (\mathsf{S}_n)^4 \times \mathbb{Z}_2^\ell \times (\mathbb{Z}_2^n)^\ell \times (\mathbb{Z}_2^n)^\ell \times (\mathbb{Z}_2^n)^{2^{v+1}-2} \times (\mathbb{Z}_2^{k_2-p})^2 \times (\mathbb{Z}_2^{k_1-\ell})^2.$$

Then for each $\phi \in \mathcal{S}_{\mathrm{GE}}$ of the following form

$$\phi = (\mathbf{t}, \sigma_g, \sigma_h, \sigma_v, \sigma_{w,0}, \sigma_{w,1}, \sigma_{\mathsf{oa},0}, \sigma_{\mathsf{oa},1}, [t_{f,1} | \cdots | t_{f,\ell}]^\top, \mathbf{b}_1, \ldots, \mathbf{b}_\ell, \mathbf{c}_1, \ldots, \mathbf{c}_\ell, \mathbf{t}_{1,0}, \mathbf{t}_{1,1}, \mathbf{t}_{2,0}, \ldots, \mathbf{t}_{2,3}, \ldots, \mathbf{t}_{v-1,0}, \ldots, \mathbf{t}_{v-1,2^{v-1}-1}, \mathbf{t}_{v,0}, \ldots, \mathbf{t}_{v,2^v-1}, \mathbf{t}_{w,0}, \mathbf{t}_{w,1}, \mathbf{t}_{\mathsf{oa},0}, \mathbf{t}_{\mathsf{oa},1}),$$

define an associated permutation Γ_ϕ as below. It transforms a vector $\widehat{\mathbf{w}}_{\mathrm{GE}} \in \mathbb{Z}_2^{L_{\mathrm{GE}}}$ of the form (46) to a vector $\Gamma_\phi(\widehat{\mathbf{w}}_{\mathrm{GE}})$ of the following form

$$\begin{aligned}\Gamma_\phi(\widehat{\mathbf{w}}_{\mathrm{GE}}) = \big(&\mathbf{z}^\star \| \mathbf{h}^\star \| \mathbf{y}^\star_1 \| \cdots \| \mathbf{y}^\star_\ell \| \mathbf{z}^\star_1 \| \cdots \| \mathbf{z}^\star_\ell \| \mathbf{x}^\star_1 \| \cdots \| \mathbf{x}^\star_\ell \| \mathbf{v}^{\star\star}_{\ell+1} \| \\ &\mathbf{x}^{j\star}_{1,0} \| \mathbf{x}^{j\star}_{1,1} \| \mathbf{x}^{j\star}_{2,0} \| \cdots \| \mathbf{x}^{j\star}_{2,3} \| \cdots \| \mathbf{x}^{j\star}_{v-1,0} \| \cdots \| \mathbf{x}^{j\star}_{v-1,2^{v-1}-1} \| \\ &\mathbf{q}^{j\star}_{1,0} \| \mathbf{q}^{j\star}_{1,1} \| \mathbf{q}^{j\star}_{2,0} \| \cdots \| \mathbf{q}^{j\star}_{2,3} \| \cdots \| \mathbf{q}^{j\star}_{v-1,0} \| \cdots \| \mathbf{q}^{j\star}_{v-1,2^{v-1}-1} \| \\ &\mathbf{q}^{j\star}_{v,0} \| \mathbf{q}^{j\star}_{v,1} \| \cdots \| \mathbf{q}^{j\star}_{v,2^v-1} \| \\ &\mathbf{k}^{(0)\star}_{j,0} \| \mathbf{k}^{(1)\star}_{j,0} \| \mathbf{k}^{(0)\star}_{j,1} \| \mathbf{k}^{(1)\star}_{j,1} \| \\ &\mathbf{e}^\star_{w,0} \| \mathbf{e}^\star_{w,1} \| \mathbf{h}^\star_{\mathsf{oa},0} \| \mathbf{h}^\star_{\mathsf{oa},1} \| \mathbf{f}^\star \| \mathbf{e}^\star_{\mathsf{oa},0} \| \mathbf{e}^\star_{\mathsf{oa},1}\big),\end{aligned} \tag{47}$$

such that

(i) $\mathbf{z}^\star = \Psi'_{\mathsf{permit}(\mathbf{t}),\sigma_g}(\widehat{\mathbf{z}})$ and $\mathbf{h}^\star = \sigma_h(\widehat{\mathbf{h}})$.

(ii) $\mathbf{v}^{\star\star}_{\ell+1} = \sigma_v(\widehat{\mathbf{v}}^*_{\ell+1})$. For $b \in \{0,1\}$, $\mathbf{e}^\star_{w,b} = \sigma_{w,b}(\widehat{\mathbf{e}}_{w,b})$ and $\mathbf{e}^\star_{\mathsf{oa},b} = \sigma_{\mathsf{oa},b}(\widehat{\mathbf{e}}_{\mathsf{oa},b})$.

(iii) For $i \in [1,\ell]$, $\mathbf{y}^\star_i = \Phi_{t_{f,i},\mathbf{b}_i}(\widehat{\mathbf{y}}_i)$, $\mathbf{z}^\star_i = \Phi_{\bar{t}_{f,i},\mathbf{c}_i}(\widehat{\mathbf{z}}_i)$, and $\mathbf{x}^\star_i = F_{\mathbf{b}_i}(\widehat{\mathbf{x}}_i)$.

(iv) For $u \in [1, v-1]$, $i \in [0, 2^u - 1]$, $\mathbf{x}^{j\star}_{k,i} = F_{\mathbf{t}_{u,i}}(\widehat{\mathbf{x}}^j_{u,i})$ and $\mathbf{q}^{j\star}_{u,i} = E'_{\mathbf{t}_{u,i}}(\widehat{\mathbf{q}}^j_{u,i})$.

(v) For $i \in [0, 2^v - 1]$, $\mathbf{q}^{j\star}_{v,i} = E'_{\mathbf{t}_{v,i}}(\widehat{\mathbf{q}}^j_{v,i})$.

(vi) Denote

$$\mathbf{t}^{(0)}_0 = (\mathbf{t}_{v,0}\|\cdots\|\mathbf{t}_{v,m-p-1}), \quad \mathbf{t}^{(1)}_0 = (\mathbf{t}_{v,m-p}\|\cdots\|\mathbf{t}_{v,m-1}),$$
$$\mathbf{t}^{(0)}_1 = (\mathbf{t}_{v,m}\|\cdots\|\mathbf{t}_{v,2m-p-1}), \quad \mathbf{t}^{(1)}_1 = (\mathbf{t}_{v,2m-p}\|\cdots\|\mathbf{t}_{v,2m-1}).$$

For $b \in \{0,1\}$, $\mathbf{k}^{(0)\star}_{j,b} = T'_{\mathbf{t}^{(0)}_b,\mathbf{t}_{w,b}}(\widehat{\mathbf{k}}^{(0)}_{j,b})$ and $\mathbf{k}^{(1)\star}_{j,b} = T'_{\mathbf{t}^{(1)}_b,\mathbf{t}}(\widehat{\mathbf{k}}^{(1)}_{j,b})$.

(vii) For $b \in \{0,1\}$, $\mathbf{h}^\star_{\mathsf{oa},b} = F_{\mathbf{t}_{\mathsf{oa},b},}(\widehat{\mathbf{h}}_{\mathsf{oa},b})$.

(viii) Denote $\mathbf{t}_f = [t_{f,1}|\cdots|t_{f,\ell}]^\top$. Then $\mathbf{f}^\star = F_{\mathbf{t}_f}(\widehat{\mathbf{f}})$.

Based on the equivalences observed in (3), (5), (7), (9), (10), (18) and (26), one can verify that for $\phi \in \mathcal{S}_{\mathrm{GE}}$, the equivalence $\widehat{\mathbf{w}}_{\mathrm{GE}} \in \mathsf{VALID}_{\mathrm{GE}} \iff \Gamma_\phi(\widehat{\mathbf{w}}_{\mathrm{GE}}) \in \mathsf{VALID}_{\mathrm{GE}}$ holds. In addition, if ϕ is uniformly chosen from $\mathcal{S}_{\mathrm{GE}}$ and $\widehat{\mathbf{w}}_{\mathrm{GE}} \in \mathsf{VALID}_{\mathrm{GE}}$, then $\Gamma_\phi(\widehat{\mathbf{w}}_{\mathrm{GE}})$ is uniformly distributed in $\mathsf{VALID}_{\mathrm{GE}}$. In other words, the equivalences in (2) are satisfied and we have managed to convert the considered statements to an instance of R_{abstract}. Now $\mathcal{P}$ and $\mathcal{V}$ can interact as described in Figure 2. The resulting protocol is a statistical ZKAoK with perfect completeness, soundness error 2/3, and communication cost $\mathcal{O}(L_{\mathrm{GE}}) = \mathcal{O}(\lambda^2 + \ell \cdot \lambda)$ bits.

8 Security Proofs of Our Group Encryption Scheme

In this section, we prove that our fully dynamic group encryption satisfies message secrecy, anonymity and soundness as defined in Section 2.2.

Theorem 3. *Assume that the* $\mathsf{DMcE}(n, k_2, t_2)$ *and the* $\mathsf{DLPN}(n, k_2 - p, \mathsf{B}(n, t_2))$ *problems are hard, the argument protocol in Section 7.1 is zero-knowledge and simulation-sound, and the argument protocol in Section 7.2 is zero-knowledge, then the given* FDGE *scheme satisfies message secrecy in the random oracle model.*

Proof. We prove the theorem using a sequence of indistinguishable games. The first game corresponds to the experiment $\mathbf{Expt}^{\mathsf{sec}-1}_{\mathcal{A}}(1^\lambda)$ as in Definition 2 while the last game corresponds to $\mathbf{Expt}^{\mathsf{sec}-0}_{\mathcal{A}}(1^\lambda)$. In the former experiment, the adversary obtains a ciphertext that is an encryption of the real witness $\mathbf{w} \in \mathbb{Z}_2^p$ such that $(S, \mathbf{w}) \in R_{\mathrm{permit}}$ and a real proof at each invocation of the oracle $\mathsf{PROVE}^1_{\mathcal{P},\mathsf{P}'}(\cdot)$. In the latter one, the adversary receives a ciphertext that encrypts a random plaintext in $\mathbb{Z}_2^p$ and a simulated proof produced by the zero-knowledge simulator of the argument protocol in Section 7.2 each time it invokes the oracle $\mathsf{PROVE}^0_{\mathcal{P},\mathsf{P}'}(\cdot)$. In Game i, W_i denotes the event that the challenger outputs 1.

Game 0**:** This is the experiment $\mathbf{Expt}^{\mathsf{sec}-1}_{\mathcal{A}}(1^\lambda)$. The challenger sends $\mathcal{A}$ public parameters pp, who in turn generates $\mathsf{pk}_{\mathsf{OA}} = (\mathbf{G}_{\mathsf{oa},0}, \mathbf{G}_{\mathsf{oa},1})$ and $\mathsf{pk}_{\mathsf{GM}}$ on behalf the OA and GM who are both fully corrupted by $\mathcal{A}$. Meanwhile, $\mathcal{A}$ controls the table **reg** and updates group information info at its will. However, if either **reg** or info is not well-formed, the experiment aborts and outputs 0. $\mathcal{A}$ is able to register honest users to the group by interacting with oracle USER and to learn secret key of honest users by querying the oracle $\mathsf{RevealU}$. The challenger maintains two lists HUL and BUL accordingly. At some point, $\mathcal{A}$ targets a receiver $\mathsf{bin}(j^*)$ with public encryption key $(\mathbf{G}_{j^*,0}, \mathbf{G}_{j^*,1})$. It then can make a polynomial number of decryption queries which are faithfully replied using the secret key $\mathsf{sk}^{(j^*,0)}_{\mathsf{ME}}$ of user $\mathsf{bin}(j^*)$. Next, $\mathcal{A}$ specifies a tuple $(\tau^*, S^*, \mathbf{w}^*, L^*)$ such that (1) user j^* is active at epoch τ^*; and (2) $(S^*, \mathbf{w}^*) \in R_{\mathrm{permit}}$, where $S^* = \{\mathbf{s}^*_1, \ldots, \mathbf{s}^*_k\}$ and $\mathbf{s}^*_i \in \{0,1\}^t$ for $i \in [1,k]$. If any of the conditions is not satisfied, the experiment aborts and output 0. Upon receiving the tuple, the challenger sends back to $\mathcal{A}$ a challenge ciphertext $\psi^* = (\mathbf{c}^*_{w,0}, \mathbf{c}^*_{w,1}, \mathbf{c}^*_{\mathsf{oa},0}, \mathbf{c}^*_{\mathsf{oa},1}, \pi^*_{ct})$ computed by running the encryption algorithm $\mathsf{Enc}(\mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathsf{OA}}, \mathsf{info}_{\tau^*}, \mathbf{w}^*, (\mathbf{G}_{j^*,0}, \mathbf{G}_{j^*,1}, \mathsf{bin}(j^*)), L^*)$. Specifically, for $i \in \{0,1\}$, $\mathbf{c}^*_{w,i} = \mathbf{G}_{j^*,i} \cdot (\mathbf{r}^*_{w,i} \| \mathbf{w}^*) \oplus \mathbf{e}^*_{w,i}$ for $\mathbf{r}^*_{w,i} \in \mathbb{Z}_2^{k_2 - p}$ and $\mathbf{e}^*_{w,i} \in \mathsf{B}(n, t_2)$. Then, $\mathcal{A}$ is further allowed to query a polynomial number of proof π_{ψ^*} of validity of the ciphertext π_{ψ^*}. In addition, $\mathcal{A}$ can still query the oracle $\mathsf{DEC}(\mathsf{sk}_{j^*}, \cdot)$ under the restriction specified in Definition 2. Finally, $\mathcal{A}$ halts and outputs b'. The game returns whatever $\mathcal{A}$ outputs. By definition, $\Pr[\mathbf{Expt}^{\mathsf{sec}-1}_{\mathcal{A}}(1^\lambda) = 1] = \Pr[W_0]$.

Game 1**:** This game changes the way we answer the DEC oracle. Let (ψ, τ, L) be a tuple queried to $\mathsf{DEC}(\mathsf{sk}_{j^*}, \cdot)$, parse $\psi = (\mathbf{c}_{w,0}, \mathbf{c}_{w,1}, \mathbf{c}_{\mathsf{oa},0}, \mathbf{c}_{\mathsf{oa},1}, \pi_{ct})$ and let $\mathsf{sk}_{j^*} = (\mathsf{sk}^{(j^*,0)}_{\mathsf{ME}}, \mathsf{sk}^{(j^*,1)}_{\mathsf{ME}})$. Instead of using $\mathsf{sk}^{(j^*,0)}_{\mathsf{ME}}$ to decrypt $\mathbf{c}_{w,0}$, it now employs $\mathsf{sk}^{(j^*,1)}_{\mathsf{ME}}$ to decrypt $\mathbf{c}_{w,1}$. The view of $\mathcal{A}$ is the same as in Game 0 until event F_1, in which π_{ct} is a valid proof yet $\mathbf{c}_{w,0}$ and $\mathbf{c}_{w,1}$ encrypt distinct messages, occurs. Since event F_1 breaks the soundness of the argument system in Section 7.1, we have $|\Pr[W_1] - \Pr[W_0]| \leq \Pr[F_1] \leq \mathbf{Adv}^{\mathsf{sound}}(\lambda) \leq \mathsf{negl}(\lambda)$.

Game 2**:** This game is the same as Game 1 except that we substitute π^*_{ct} with a simulated proof by running the zero-knowledge simulator of the argument system in Section 7.1. Since the argument system is statistical zero-knowledge, this modification does not change the view of $\mathcal{A}$ non-negligibly. We then have $|\Pr[W_2] - \Pr[W_1]| \leq \mathsf{negl}(\lambda)$.

Game 3**:** This game changes Game 2 in a similar way Game 2 does. It produces a simulated proof by resorting to the zero-knowledge simulator of the argument system in Section 7.2 each time $\mathcal{A}$ queries the oracle $\mathsf{PROVE}^1_{\mathcal{P},\mathcal{P}'}(\cdot)$. Due to the statistical zero-knowledge property of the argument system, the view of the adversary in Game 3 and Game 2 are statistically indistinguishable, implying $|\Pr[W_3] - \Pr[W_2]| \leq \mathsf{negl}(\lambda)$.

Game 4**:** This game modifies Game 3 by changing the challenge ciphertext ψ^*. Specifically, this game samples a random element $\mathbf{w}'^* \in \mathbb{Z}_2^p$ and compute $\mathbf{c}^*_{w,0}$ as an encryption of $\mathbf{w}'^*$. By the CPA-security of McEliece's cryptosystem,

which relies on the hardness of the problem $\mathsf{DMcE}(n, k_2, t_2)$ and problem $\mathsf{DLPN}(n, k_2 - p, \mathsf{B}(n, t_2))$, we have $|\Pr[W_4] - \Pr[W_3]| \leq \mathsf{negl}(\lambda)$.

Game 5**:** This game is identical to Game 4 except that we switch back to $\mathsf{sk}_{\mathsf{ME}}^{(j^*,0)}$ for $\mathsf{DEC}(\mathsf{sk}_{j^*}, \cdot)$ queries. We remark that in Game 4 and this game $\mathbf{c}_{w,0}^*$ is an encryption of $\mathbf{w}'^*$ while $\mathbf{c}_{w,1}^*$ is an encryption of the real witness $\mathbf{w}^*$. As a result, π_{ct}^* is a simulated proof of a false statement (since $\mathbf{c}_{w,0}^*, \mathbf{c}_{w,1}^*$ encrypt different messages). This modification remains unnoticed unless event F_2, when $\mathcal{A}$ queries a tuple in which π_{ct} is valid yet $\mathbf{c}_{w,0}, \mathbf{c}_{w,1}$ encrypt different messages, occurs. However, event F_2 violates the simulation-soundness of the argument system in Section 7.1. Therefore, $|\Pr[W_5] - \Pr[W_4]| \leq \Pr[F_2] \leq \mathbf{Adv}^{\mathsf{sim-sound}}(\lambda) \leq \mathsf{negl}(\lambda)$.

Game 6**:** This game is like Game 5 other than that $\mathbf{c}_{w,1}^*$ is an encryption of $\mathbf{w}'^*$ as $\mathbf{c}_{w,0}^*$. This change is negligible to $\mathcal{A}$ due to the CPA-security of McEliece's encryption scheme and that only the key $\mathsf{sk}_{\mathsf{ME}}^{(j,^*0)}$ is used in the $\mathsf{DEC}(\mathsf{sk}_{j^*}, \cdot)$ oracle. Hence, $|\Pr[W_6] - \Pr[W_5]| \leq \mathsf{negl}(\lambda)$.

Game 7**:** When generating the challenge ciphertext ψ^*, this game switches back to a real proof π_{ct}^*. The distance of Game 7 and Game 6 is negligible based on the statistical zero-knowledge of the argument protocol in Section 7.1. Therefore, $|\Pr[W_7] - \Pr[W_6]| \leq \mathsf{negl}(\lambda)$.

In the last game, $\mathcal{A}$ receives a challenge ciphertext that encrypts a random element and sees simulated proofs of ciphertext validity. This is exactly $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{sec}-0}(1^\lambda)$. Therefore, $\Pr[W_7] = \Pr[\mathsf{Expt}_{\mathcal{A}}^{\mathsf{sec}-0}(1^\lambda) = 1]$. Putting everything together, we have

$$\begin{aligned}\mathbf{Adv}_{\mathcal{A}}^{\mathsf{sec}} &= |\Pr[\mathsf{Expt}_{\mathcal{A}}^{\mathsf{sec}-1}(1^\lambda) = 1] - \Pr[\mathsf{Expt}_{\mathcal{A}}^{\mathsf{sec}-0}(1^\lambda) = 1]| \\ &= |\Pr[W_0] - \Pr[W_7]| \\ &\leq \mathsf{negl}(\lambda).\end{aligned}$$

This proves the theorem.

Theorem 4. *Assume that the problems* $\mathsf{DMcE}(n, k_1, t_1)$*,* $\mathsf{DMcE}(n, k_2, t_2)$*, and* $\mathsf{DLPN}(n, k_1 - \ell, \mathsf{B}(n, t_1))$*,* $\mathsf{DLPN}(n, k_2 - p, \mathsf{B}(n, t_2))$ *are hard, the argument protocol in Section 7.1 is zero-knowledge and simulation-sound, and the argument protocol in Section 7.2 is zero-knowledge, then the given fully dynamic group encryption scheme is anonymous in the random oracle model.*

Proof. We proceed via a series of games where the first is $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{anon}-b}(1^\lambda)$ while the last game does not depend on b at all. In Game i, W_i denotes the event that the output of Game i is 1.

Game 0**:** This is the experiment $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{anon}-b}(1^\lambda)$. The challenger generates public parameters pp and key pair $\mathsf{pk}_{\mathsf{OA}} = (\mathbf{G}_{\mathsf{oa},0}, \mathbf{G}_{\mathsf{oa},1})$, $\mathsf{sk}_{\mathsf{OA}} = (\mathsf{sk}_{\mathsf{ME}}^{(\mathsf{oa},0)}, \mathsf{sk}_{\mathsf{ME}}^{(\mathsf{oa},1)})$ on behalf of OA. The adversary $\mathcal{A}$, given $(\mathsf{pp}, \mathsf{pk}_{\mathsf{OA}})$, produces $\mathsf{pk}_{\mathsf{GM}}$ in the name of GM that is completely corrupted. By interacting with the oracles $\mathsf{USER}, \mathsf{RevealU}$, $\mathcal{A}$ is able to register honest receivers to the group and corrupt some of them. It also has access to the OPEN oracle. At some point, $\mathcal{A}$

chooses two receivers $(\mathbf{G}_{j_0^*,0}, \mathbf{G}_{j_0^*,1}, \mathsf{bin}(j_0^*)), (\mathbf{G}_{j_1^*,0}, \mathbf{G}_{j_1^*,1}, \mathsf{bin}(j_1^*))$ as challenged users. It can make a polynomial number of $\mathsf{DEC}(\mathsf{sk}_{j_0^*}, \cdot), \mathsf{DEC}(\mathsf{sk}_{j_1^*}, \cdot)$ queries, where $\mathsf{sk}_{j_0^*}, \mathsf{sk}_{j_1^*}$ are the secret keys of $\mathsf{bin}(j_0^*), \mathsf{bin}(j_1^*)$, respectively. After a while, $\mathcal{A}$ decides a tuple $(\tau^*, S^*, \mathbf{w}^*, L^*)$ such that (1) both challenged users are active at epoch τ^* and (2) $(S^*, \mathbf{w}^*) \in R_{\text{permit}}$, where $S^* = \{\mathbf{s}_1^*, \ldots, \mathbf{s}_k^*\}$ and $\mathbf{s}_i^* \in \{0,1\}^t$ for $i \in [1,k]$. If any of the conditions is not satisfied, the experiment aborts and output 0. The challenger computes $\psi^* = \mathsf{Enc}(\mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathsf{OA}}, \mathsf{info}_{\tau^*}, \mathbf{w}^*, (\mathbf{G}_{j_b^*,0}, \mathbf{G}_{j_b^*,1}, \mathsf{bin}(j_b^*)), L^*)$ and sends $\psi^* = (\mathbf{c}_{w,0}^*, \mathbf{c}_{w,1}^*, \mathbf{c}_{\mathsf{oa},0}^*, \mathbf{c}_{\mathsf{oa},1}^*, \pi_{ct}^*)$ to $\mathcal{A}$. The latter obtains proofs π_{ψ^*} for ψ^* and makes further decryption queries and opening queries with the obvious restrictions. Note that for $i \in \{0,1\}$, $\mathbf{c}_{w,i}^*$ and $\mathbf{c}_{\mathsf{oa},i}^*$ are encryption of $\mathbf{w}^*$ and $\mathsf{bin}(j_b^*)$ under the key $\mathbf{G}_{j_b^*,i}$ and $\mathbf{G}_{\mathsf{oa},i}$, respectively. When $\mathcal{A}$ halts and outputs b', this game outputs 1 if and only if $b' = b$. By definition, $\Pr[\mathsf{Expt}_{\mathcal{A}}^{\mathsf{anon}-b}(1^\lambda) = b] = \Pr[W_0]$.

Game 1: We modify Game 0 by utilizing $\mathsf{sk}_{\mathsf{ME}}^{(\mathsf{oa},1)}$ to decrypt $\mathbf{c}_{\mathsf{oa},1}$ when $\mathcal{A}$ invokes $\mathsf{OPEN}(\mathsf{sk}_{\mathsf{OA}}, \cdot)$ on (ψ, τ, L) with $\psi = (\mathbf{c}_{w,0}, \mathbf{c}_{w,1}, \mathbf{c}_{\mathsf{oa},0}, \mathbf{c}_{\mathsf{oa},1}, \pi_{ct})$. Note that in Game 0, $\mathsf{OPEN}(\mathsf{sk}_{\mathsf{OA}}, \cdot)$ oracle employs $\mathsf{sk}_{\mathsf{ME}}^{(\mathsf{oa},0)}$ to decrypt $\mathbf{c}_{\mathsf{oa},0}$. The view of $\mathcal{A}$ will remain unchanged unless $\mathcal{A}$ comes up with a tuple (ψ, τ, L) such that π_{ct} is valid yet $\mathbf{c}_{\mathsf{oa},0}, \mathbf{c}_{\mathsf{oa},1}$ are encryptions of different strings. However, this would breach the soundness of the argument system in Section 7.1. We thus have $|\Pr[W_1] - \Pr[W_0]| \leq \mathbf{Adv}^{\mathsf{sound}} \leq \mathsf{negl}(\lambda)$.

Game 2: In this game, we appeal to the zero-knowledge simulator of the argument system in Section 7.1, obtaining a simulated proof π_{ct}^*. The statistical zero-knowledge of the argument ensures that the view of $\mathcal{A}$ in this game is statistically close to that in Game 1. In other words, we have $|\Pr[W_2] - \Pr[W_1]| \leq \mathsf{negl}(1^\lambda)$.

Game 3: This game modifies Game 2 by relying on the zero-knowledge simulator of the argument system in Section 7.2 when $\mathcal{A}$ asks for π_{ψ^*}. Since this argument system is statistical zero-knowledge, we obtain $|\Pr[W_3] - \Pr[W_2]| \leq \mathsf{negl}(1^\lambda)$.

Game 4: We now change the challenge ciphertext ψ^* by replacing $\mathbf{c}_{\mathsf{oa},0}^*$ with a random string $\mathbf{c}_{\mathsf{oa},0}^{'*}$ sampled from the space $\mathbb{Z}_2^n$. Since $\mathsf{sk}_{\mathsf{ME}}^{(\mathsf{oa},0)}$ is no longer employed for opening queries, any noticeable change in $\mathcal{A}$'s output implies another adversary $\mathcal{B}$ distinguishing a McEliece ciphertext from random. Due to the hardness of the $\mathsf{DMcE}(n, k_1, t_1)$ and $\mathsf{DLPN}(n, k_1 - \ell, \mathsf{B}(n, t_1))$ problems, the advantage of $\mathcal{B}$ is negligible, so is $|\Pr[W_4 - \Pr[W_3]|$.

Game 5: This game switches back to $\mathsf{sk}_{\mathsf{ME}}^{(\mathsf{oa},0)}$ when replying the $\mathsf{OPEN}(\mathsf{sk}_{\mathsf{OA}}, \cdot)$ oracle. This change will remain unnoticed unless $\mathcal{A}$ queries a tuple (ψ, τ, L) such that π_{ct} is valid yet $\mathbf{c}_{\mathsf{oa},0}, \mathbf{c}_{\mathsf{oa},1}$ are encryptions of different strings. This, however, will break the simulation-soundness of the argument system in Section 7.1 as $\mathcal{A}$ sees a simulated proof π_{ct}^* for a false statement (since $\mathbf{c}_{\mathsf{oa},0}^{'*}$ is random while $\mathbf{c}_{\mathsf{oa},1}^*$ is an encryption of $\mathsf{bin}(j_b^*)$). As a result, $|\Pr[W_5] - \Pr[W_4]| \leq \mathbf{Adv}^{\mathsf{sim}-\mathsf{sound}} \leq \mathsf{negl}(\lambda)$.

Game 6: We further change the challenge ciphertext ψ^* in this game. Instead of being an encryption of $\mathsf{bin}(j_b^*)$, $\mathbf{c}^*_{\mathsf{oa},1}$ is replaced with a random string $\mathbf{c}'^*_{\mathsf{oa},1} \in \mathbb{Z}_2^n$. As $\mathsf{sk}_{\mathsf{ME}}^{(\mathsf{oa},1)}$ is not employed for opening queries, the pseudorandom ciphertexts of the McEliece scheme imply that $|\Pr[W_6] - \Pr[W_5]| \leq \mathsf{negl}(\lambda)$.

Game 7: This game makes a last modification to ψ^*. It replaces $(\mathbf{c}^*_{w,0}, \mathbf{c}^*_{w,1})$ with two random strings $(\mathbf{c}'^*_{w,0}, \mathbf{c}'^*_{w,1}) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n$ by switching between $\mathsf{sk}_{\mathsf{ME}}^{(j_b^*,0)}$ and $\mathsf{sk}_{\mathsf{ME}}^{(j_b^*,1)}$. Based on the hardness of the problems $\mathsf{DMcE}(n, k_2, t_2)$ and $\mathsf{DLPN}(n, k_2 - p, \mathsf{B}(n, t_2))$, and simulation-soundness of the argument system in Section 7.1, it is not hard to see that the view of $\mathcal{A}$ in Game 7 is indistinguishable from that in Game 6. In other words, $|\Pr[W_7] - \Pr[W_6]| \leq \mathsf{negl}(\lambda)$.

In the last game, we see that $\mathcal{A}$'s view does not depend on b any more. Therefore, $\Pr[W_7] = \frac{1}{2}$. Putting everything together, we obtain $|\Pr[\mathsf{Expt}_{\mathcal{A}}^{\mathsf{anon}-b}(1^\lambda) = b] - \frac{1}{2}| \leq \mathsf{negl}(\lambda)$. In addition,

$$\begin{aligned}
\mathbf{Adv}_{\mathcal{A}}^{\mathsf{anon}} &= |\Pr[\mathsf{Expt}_{\mathcal{A}}^{\mathsf{anon}-1}(1^\lambda) = 1] - \Pr[\mathsf{Expt}_{\mathcal{A}}^{\mathsf{anon}-0}(1^\lambda) = 1]| \\
&= |\Pr[\mathsf{Expt}_{\mathcal{A}}^{\mathsf{anon}-1}(1^\lambda) = 1] + \Pr[\mathsf{Expt}_{\mathcal{A}}^{\mathsf{anon}-0}(1^\lambda) = 0] - 1| \\
&\leq |\Pr[\mathsf{Expt}_{\mathcal{A}}^{\mathsf{anon}-1}(1^\lambda) = 1] - \frac{1}{2}| + |\Pr[\mathsf{Expt}_{\mathcal{A}}^{\mathsf{anon}-0}(1^\lambda) = 0] - \frac{1}{2}| \\
&\leq \mathsf{negl}(\lambda).
\end{aligned}$$

This ends the proof.

Theorem 5. *Assume that the* $2\text{-}\mathsf{RNSD}_{n,2n,c}$ *problem is hard, and the argument system in Section 7.2 is sound, then the given fully dynamic group encryption scheme provides soundness in the random oracle model.*

Proof. If an adversary $\mathcal{A}$ breaks soundness as in definition 4 with non-negligible advantage ϵ, then we construct a PPT algorithm $\mathcal{B}$ that uses $\mathcal{A}$ as a subroutine to either break an instance of the $2\text{-}\mathsf{RNSD}_{n,2n,c}$ problem or to break the soundness of the argument system in Section 7.2 with non-negligible probability as well.

Given a matrix $\mathbf{B} \in \mathbb{Z}_2^{n \times m}$, $\mathcal{B}$ simulates the experiment $\mathbf{Expt}_{\mathsf{A}}^{\mathsf{sound}}(1^\lambda)$ as follows. It first generates other parameters except $\mathbf{B}$ as specified in the algorithm $\mathsf{Setup}_{\mathsf{init}}$, produces the key pairs $(\mathsf{pk}_{\mathsf{OA}}, \mathsf{sk}_{\mathsf{OA}})$ and $(\mathsf{pk}_{\mathsf{GM}}, \mathsf{sk}_{\mathsf{GM}})$ on behalf of OA and GM, and initializes the **reg** table and group information info. Next, it invokes $\mathcal{A}$ by sending over public parameters, $\mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathsf{OA}}, \mathsf{sk}_{\mathsf{OA}}$. It then interacts with $\mathcal{A}$ by answers oracle queries REG and GUp honestly. When $\mathcal{A}$ queries the hash function $\mathcal{H}_{\mathsf{FS}}$, $\mathcal{B}$ replies random elements from the set $\{1, 2, 3\}^\kappa$. In the end, $\mathcal{A}$ outputs a tuple $(\tau^*, S^*, \psi^*, L^*, \pi_{\psi^*})$. Parse $\psi^* = (\mathbf{c}^*_{w,0}, \mathbf{c}^*_{w,1}, \mathbf{c}^*_{\mathsf{oa},0}, \mathbf{c}^*_{\mathsf{oa},0}, \pi^*_{ct})$ and $\pi_{\psi^*} = (\{\mathsf{CMT}_i^*\}_{i=1}^\kappa, \mathsf{Ch}^*, \{\mathsf{RSP}_i^*\}_{i=1}^\kappa)$. If $\mathcal{A}$ breaks soundness, then π_{ψ^*} is a valid proof. Let $\delta^* = (\{\mathsf{CMT}_i^*\}_{i=1}^\kappa, \mathbf{B}, \mathsf{pk}_{\mathsf{OA}}, \mathbf{u}_{\tau^*}, S^*, \psi^*, L^*)$. We claim that $\mathcal{A}$ has queried δ^* to the hash oracle $\mathcal{H}_{\mathsf{FS}}$ with overwhelming probability. Otherwise, the probability of guessing a value Ch^* equal to $\mathcal{H}_{\mathsf{FS}}(\delta^*)$ is at most $3^{-\kappa}$, which is negligible. Therefore, with probability at least $\epsilon' = \epsilon - 3^{-\kappa}$, $\mathcal{A}$ has queried δ^* to the hash function. Denote $t^* \in \{1, 2, \cdots, Q_H\}$ the index of this hash query,

where Q_H is the total number of hash queries. Then $\mathcal{B}$ replays $\mathcal{A}$ for at most $32\frac{Q_H}{\epsilon'}$ times. For each replay, $\mathcal{A}$ is given the same random tape and input as in the original invocation. When it queries the hash function, $\mathcal{B}$ replies the same random strings as in the original invocation for the first $t^* - 1$ queries, and independent random strings for the remaining queries. This ensures that for each replay, the input to the t^*-th hash query is the same as the original run yet the reply is an independent random string. By the forking lemma [18], with probability at least $\frac{1}{2}$, $\mathcal{B}$ obtains a 3-fork involving the same tuple δ^* with pairwise distinct hash value $\mathsf{Ch}_{t^*}^{(1)}, \mathsf{Ch}_{t^*}^{(2)}, \mathsf{Ch}_{t^*}^{(3)} \in \{1,2,3\}^\kappa$. With probability $1-(\frac{7}{9})^\kappa$, there exists an index $j \in [1,\kappa]$ such that $\{\mathsf{Ch}_{t^*,j}^{(1)}, \mathsf{Ch}_{t^*,j}^{(2)}, \mathsf{Ch}_{t^*,j}^{(3)}\} = \{1,2,3\}$. In other words, with probability $\frac{1}{2}(1-(\frac{7}{9})^\kappa)$, $\mathcal{B}$ obtains a commitment CMT_j^* and three valid responses $\mathsf{RSP}_{t^*,j}^{(1)}, \mathsf{RSP}_{t^*,j}^{(2)}, \mathsf{RSP}_{t^*,j}^{(3)}$ for all possible challenges $1,2,3$. The soundness of the argument system in Section 7.2 implies that $\mathcal{B}$ can extract witness $\xi^* = (\mathbf{w}^*, \mathsf{bin}(j^*), \mathbf{G}_{j^*,0}^*, \mathbf{G}_{j^*,1}^*, \mathbf{d}_{j^*}^*, w^{(j^*)}, \mathbf{r}_{w,0}^*, \mathbf{r}_{w,1}^*, \mathbf{r}_{\mathsf{oa},0}^*, \mathbf{r}_{\mathsf{oa},1}^*, \mathbf{e}_{w,0}^*, \mathbf{e}_{w,1}^*, \mathbf{e}_{\mathsf{oa},0}^*, \mathbf{e}_{\mathsf{oa},1}^*)$, in which $\mathbf{w}^* \in \mathbb{Z}_2^p$, $\mathbf{d}_{j^*}^* \neq \mathbf{0}$, and

(1) $(S^*, \mathbf{w}^*) \in R_{\text{permit}}$, $\mathsf{Verify}_{\mathbf{B}}(\mathbf{u}_{\tau^*}, \mathbf{d}_{j^*}^*, w^{(j^*)}) = 1$ and $\mathsf{Accu}_{\mathbf{B}}\big((\mathbf{G}_{j^*,0}^*, \mathbf{G}_{j^*,1}^*)\big) = \mathbf{d}_{j^*}^*$;
(2) The ciphertexts $\mathbf{c}_{w,0}^*$, $\mathbf{c}_{w,1}^*$ are encryption of $\mathbf{w}^*$ under the keys $\mathbf{G}_{j^*,0}^*$ and $\mathbf{G}_{j^*,0}^*$, respectively;
(3) The ciphertexts $\mathbf{c}_{\mathsf{oa},0}^*, \mathbf{c}_{\mathsf{oa},1}^*$ are encryption of $\mathsf{bin}(j^*)$ under the keys $\mathbf{G}_{\mathsf{oa},0}$ and $\mathbf{G}_{\mathsf{oa},0}$, respectively.

From the correctness our the underlying encryption scheme, we know that $\mathbf{c}_{\mathsf{oa},0}^*$ will be decrypted to $\mathsf{bin}(j^*)$. At this point, we distinguish the following cases.
Case 1**:** $\mathsf{IsActive}(\mathsf{info}_{\tau^*}, \mathsf{bin}(j^*)) = 0$. This implies that the real leaf value $\mathbf{d}_{j^*}$ of the leaf node $\mathsf{bin}(j^*)$ of the (main-)Merkle tree is $\mathbf{0}^n$. Therefore, we have two different paths from the leaf node $\mathsf{bin}(j^*)$ to the root $\mathbf{u}_{\tau^*}$. Hence, one is able to find a vector $\mathbf{z} \in 2\text{-}\mathsf{regular}(2n,c)$ such that $\mathbf{B}\cdot\mathbf{z} = \mathbf{0} \bmod 2$. This solves the $2\text{-}\mathsf{RNSD}_{n,2n,c}$ problem.
Case 2**:** $\mathsf{IsActive}(\mathsf{info}_{\tau^*}, \mathsf{bin}(j^*)) = 1$ and $(\mathbf{G}_{j^*,0}^*, \mathbf{G}_{j^*,1}^*) \neq (\mathbf{G}_{j^*,0}, \mathbf{G}_{j^*,1})$, where $(\mathbf{G}_{j^*,0}, \mathbf{G}_{j^*,1})$ is the real encryption key of user $\mathsf{bin}(j^*)$. This, however, would solve the $2\text{-}\mathsf{RNSD}_{n,2n,c}$ problem as in **Case** 1.
Case 3**:** $\mathsf{IsActive}(\mathsf{info}_{\tau^*}, \mathsf{bin}(j^*)) = 1$ and $(\mathbf{G}_{j^*,0}^*, \mathbf{G}_{j^*,1}^*) = (\mathbf{G}_{j^*,0}, \mathbf{G}_{j^*,1})$ yet $(\mathbf{G}_{j^*,0}, \mathbf{G}_{j^*,1}) \notin \mathcal{L}_{\mathsf{pk}}^{\mathsf{pp}}$. In other words, there is no corresponding secret keys for $(\mathbf{G}_{j^*,0}, \mathbf{G}_{j^*,1})$. As GM issues an identifier $\mathsf{bin}(j^*)$ for this user, then $\mathbf{G}_{j^*,0}$ and $\mathbf{G}_{j^*,1}$ are full rank. If the user does not know the corresponding secret keys, then outputting correctly the random messages $\mathbf{m}_0, \mathbf{m}_1$ encrypted by GM would require running time at least $k_2^3 \cdot \binom{n}{k_2} / \binom{n-t_2}{k_2}$[13]. This is exponential by the choices of our parameters. In other words, this case only occurs with negligible probability.
Case 4**:** $\mathsf{IsActive}(\mathsf{info}_{\tau^*}, \mathsf{bin}(j^*)) = 1$ and $(\mathbf{G}_{j^*,0}^*, \mathbf{G}_{j^*,1}^*) = (\mathbf{G}_{j^*,0}, \mathbf{G}_{j^*,1}) \in \mathcal{L}_{\mathsf{pk}}^{\mathsf{pp}}$ yet $\psi^* \notin \mathcal{L}_{\mathsf{ciphertext}}^{\mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathsf{OA}}, \tau^*, \mathsf{pk}_{\mathcal{R}}^*, S^*, L^*, \mathsf{bin}(j^*)}$. However, the soundness of the argument system already eliminates this case.

This ends the proof.

[13] Note that this user can generate $\mathbf{G}_{j^*,b}$ honestly and then only needs to guess $\mathbf{m}_{1-b}$.

Acknowledgments

Khoa Nguyen and Huaxiong Wang were supported by Singapore Ministry of Education under Research Grant MOE2019-T2-2-083 and by A*Star, Singapore under research grant SERC A19E3b0099. Reihaneh Safavi-Naini and Yanhong Xu were in part supported by Natural Sciences and Engineering Research Council of Canada Discovery Grant Program and Alberta Innovates Strategic Chair in Information Security Research Program. Neng Zeng was supported by the Singapore Ministry of Education under MOE AcRF Tier 2 grant (MOE2018-T2-1-111).

References

1. L. E. Aimani and M. Joye. Toward practical group encryption. In M. J. J. Jr., M. E. Locasto, P. Mohassel, and R. Safavi-Naini, editors, *ACNS 2013*, volume 7954 of *LNCS*, pages 237–252. Springer, 2013.
2. Q. Alamélou, O. Blazy, S. Cauchie, and P. Gaborit. A code-based group signature scheme. *Des. Codes Cryptography*, 82(1-2):469–493, 2017.
3. B. Applebaum, N. Haramaty, Y. Ishai, E. Kushilevitz, and V. Vaikuntanathan. Low-complexity cryptographic hash functions. In *ITCS 2017*, volume 67 of *LIPIcs*, pages 7:1–7:31. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.
4. D. Augot, M. Finiasz, and N. Sendrier. A fast provably secure cryptographic hash function. *IACR Cryptol. ePrint Arch.*, 2003:230, 2003.
5. D. Augot, M. Finiasz, and N. Sendrier. A family of fast syndrome based cryptographic hash functions. In *Mycrypt 2005*, volume 3715 of *LNCS*, pages 64–83. Springer, 2005.
6. M. Bellare, A. Boldyreva, A. Desai, and D. Pointcheval. Key-privacy in public-key encryption. In C. Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 566–582. Springer, 2001.
7. M. Bellare, D. Micciancio, and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In E. Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 614–629. Springer, 2003.
8. M. Bellare, H. Shi, and C. Zhang. Foundations of group signatures: The case of dynamic groups. In A. Menezes, editor, *CT-RSA 2005*, volume 3376 of *LNCS*, pages 136–153. Springer, 2005.
9. J. C. Benaloh and M. de Mare. One-way accumulators: A decentralized alternative to digital sinatures (extended abstract). In T. Helleseth, editor, *EUROCRYPT 1993*, volume 765 of *LNCS*, pages 274–285. Springer, 1993.
10. F. Benhamouda, J. Camenisch, S. Krenn, V. Lyubashevsky, and G. Neven. Better zero-knowledge proofs for lattice encryption and their application to group signatures. In P. Sarkar and T. Iwata, editors, *ASIACRYPT 2014*, volume 8873 of *LNCS*, pages 551–572. Springer, 2014.
11. D. J. Bernstein, T. Lange, C. Peters, and P. Schwabe. Faster 2-regular information-set decoding. In *IWCC 2011*, volume 6639 of *LNCS*, pages 81–98. Springer, 2011.
12. D. Boneh, A. Sahai, and B. Waters. Functional encryption: a new vision for public-key cryptography. *Commun. ACM*, 55(11):56–64, 2012.

13. D. Boneh and H. Shacham. Group signatures with verifier-local revocation. In V. Atluri, B. Pfitzmann, and P. D. McDaniel, editors, *CCS 2004*, pages 168–177. ACM, 2004.
14. J. Bootle, A. Cerulli, P. Chaidos, E. Ghadafi, and J. Groth. Foundations of fully dynamic group signatures. In M. Manulis, A. Sadeghi, and S. A. Schneider, editors, *ACNS 2016*, volume 9696 of *LNCS*, pages 117–136. Springer, 2016.
15. P. Branco and P. Mateus. A code-based linkable ring signature scheme. In J. Baek, W. Susilo, and J. Kim, editors, *ProvSec 2018*, volume 11192 of *LNCS*, pages 203–219. Springer, 2018.
16. G. Brassard, D. Chaum, and C. Crépeau. Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.*, 37(2):156–189, 1988.
17. E. Bresson and J. Stern. Efficient revocation in group signatures. In K. Kim, editor, *PKC 2001*, volume 1992 of *LNCS*, pages 190–206. Springer, 2001.
18. E. F. Brickell, D. Pointcheval, S. Vaudenay, and M. Yung. Design validations for discrete logarithm based signature schemes. In H. Imai and Y. Zheng, editors, *PKC 2000*, volume 1751 of *LNCS*, pages 276–292. Springer, 2000.
19. J. Camenisch and A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In M. Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 61–76. Springer, 2002.
20. J. Cathalo, B. Libert, and M. Yung. Group encryption: Non-interactive realization in the standard model. In M. Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 179–196. Springer, 2009.
21. P. Cayrel, G. Hoffmann, and E. Persichetti. Efficient implementation of a cca2-secure variant of mceliece using generalized srivastava codes. In M. Fischlin, J. A. Buchmann, and M. Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 138–155. Springer, 2012.
22. D. Chaum and E. van Heyst. Group signatures. In D. W. Davies, editor, *EUROCRYPT 1991*, volume 547 of *LNCS*, pages 257–265. Springer, 1991.
23. L. Dallot and D. Vergnaud. Provably secure code-based threshold ring signatures. In M. G. Parker, editor, *IMACC 2009*, volume 5921 of *LNCS*, pages 222–235. Springer, 2009.
24. N. Döttling, R. Dowsley, J. Müller-Quade, and A. C. A. Nascimento. A CCA2 secure variant of the mceliece cryptosystem. *IEEE Trans. Inf. Theory*, 58(10):6672–6680, 2012.
25. M. F. Ezerman, H. T. Lee, S. Ling, K. Nguyen, and H. Wang. A provably secure group signature scheme from code-based assumptions. In T. Iwata and J. H. Cheon, editors, *ASIACRYPT 2015*, volume 9452 of *LNCS*, pages 260–285. Springer, 2015.
26. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. M. Odlyzko, editor, *CRYPTO 1986*, volume 263 of *LNCS*, pages 186–194. Springer, 1986.
27. C. Gentry. Fully homomorphic encryption using ideal lattices. In M. Mitzenmacher, editor, *STOC 2009*, pages 169–178. ACM, 2009.
28. O. Goldreich, S. Micali, and A. Wigderson. How to prove all np-statements in zero-knowledge, and a methodology of cryptographic protocol design. In A. M. Odlyzko, editor, *CRYPTO 1986*, volume 263 of *LNCS*, pages 171–185. Springer, 1986.
29. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.
30. V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In A. Juels, R. N. Wright, and S. D. C. di Vimercati, editors, *CCS 2006*, pages 89–98. ACM, 2006.

31. M. Izabachène, D. Pointcheval, and D. Vergnaud. Mediated traceable anonymous encryption. In M. Abdalla and P. S. L. M. Barreto, editors, *LATINCRYPT 2010*, volume 6212 of *LNCS*, pages 40–60. Springer, 2010.
32. A. Jain, S. Krenn, K. Pietrzak, and A. Tentes. Commitments and efficient zero-knowledge proofs from learning parity with noise. In X. Wang and K. Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 663–680. Springer, 2012.
33. A. Kiayias, Y. Tsiounis, and M. Yung. Traceable signatures. In C. Cachin and J. Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 571–589. Springer, 2004.
34. A. Kiayias, Y. Tsiounis, and M. Yung. Group encryption. In K. Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 181–199. Springer, 2007.
35. K. Kobara and H. Imai. Semantically secure mceliece public-key cryptosystems-conversions for mceliece PKC. In K. Kim, editor, *PKC 2001*, volume 1992 of *LNCS*, pages 19–35. Springer, 2001.
36. B. Libert, S. Ling, F. Mouhartem, K. Nguyen, and H. Wang. Signature schemes with efficient protocols and dynamic group signatures from lattice assumptions. In J. H. Cheon and T. Takagi, editors, *ASIACRYPT 2016*, volume 10032 of *LNCS*, pages 373–403, 2016.
37. B. Libert, S. Ling, F. Mouhartem, K. Nguyen, and H. Wang. Zero-knowledge arguments for matrix-vector relations and lattice-based group encryption. *Theor. Comput. Sci.*, 759:72–97, 2019.
38. B. Libert, S. Ling, K. Nguyen, and H. Wang. Zero-knowledge arguments for lattice-based accumulators: Logarithmic-size ring signatures and group signatures without trapdoors. In M. Fischlin and J. Coron, editors, *EUROCRYPT 2016*, volume 9666 of *LNCS*, pages 1–31. Springer, 2016.
39. B. Libert, T. Peters, and M. Yung. Group signatures with almost-for-free revocation. In R. Safavi-Naini and R. Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 571–589. Springer, 2012.
40. B. Libert, T. Peters, and M. Yung. Scalable group signatures with revocation. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 609–627. Springer, 2012.
41. B. Libert, M. Yung, M. Joye, and T. Peters. Traceable group encryption. In H. Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 592–610. Springer, 2014.
42. S. Ling, K. Nguyen, D. Stehlé, and H. Wang. Improved zero-knowledge proofs of knowledge for the ISIS problem, and applications. In K. Kurosawa and G. Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 107–124. Springer, 2013.
43. S. Ling, K. Nguyen, H. Wang, and Y. Xu. Accountable tracing signatures from lattices. In M. Matsui, editor, *CT-RSA 2019*, volume 11405 of *LNCS*, pages 556–576. Springer, 2019.
44. S. Ling, K. Nguyen, H. Wang, and Y. Xu. Lattice-based group signatures: Achieving full dynamicity (and deniability) with ease. *Theor. Comput. Sci.*, 783:71–94, 2019.
45. V. Lyubashevsky and D. Micciancio. Asymptotically efficient lattice-based digital signatures. *J. Cryptology*, 31(3):774–797, 2018.
46. V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. *J. ACM*, 60(6):43:1–43:35, 2013.
47. R. J. McEliece. A public-key cryptosystem based on algebraic coding theory. *Coding Thv*, 4244:114–116, 1978.

48. C. A. Melchor, P. Cayrel, and P. Gaborit. A new efficient threshold ring signature scheme based on coding theory. In J. A. Buchmann and J. Ding, editors, *PQCrypto 2008*, volume 5299 of *LNCS*, pages 1–16. Springer, 2008.
49. C. A. Melchor, P. Cayrel, P. Gaborit, and F. Laguillaumie. A new efficient threshold ring signature scheme based on coding theory. *IEEE Trans. Inf. Theory*, 57(7):4833–4842, 2011.
50. K. Morozov and T. Takagi. Zero-knowledge protocols for the mceliece encryption. In W. Susilo, Y. Mu, and J. Seberry, editors, *ACISP 2012*, volume 7372 of *LNCS*, pages 180–193. Springer, 2012.
51. T. Nakanishi, H. Fujii, Y. Hira, and N. Funabiki. Revocable group signature schemes with constant costs for signing and verifying. In S. Jarecki and G. Tsudik, editors, *PKC 2009*, volume 5443 of *LNCS*, pages 463–480. Springer, 2009.
52. M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In H. Ortiz, editor, *STOC 1990*, pages 427–437. ACM, 1990.
53. K. Nguyen, H. Tang, H. Wang, and N. Zeng. New code-based privacy-preserving cryptographic constructions. In S. D. Galbraith and S. Moriai, editors, *ASIACRYPT 2019*, volume 11922 of *LNCS*, pages 25–55. Springer, 2019.
54. R. Nojima, H. Imai, K. Kobara, and K. Morozov. Semantic security for the mceliece cryptosystem without random oracles. *Des. Codes Cryptogr.*, 49(1-3):289–305, 2008.
55. O. Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6):34:1–34:40, 2009.
56. R. L. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In C. Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 552–565. Springer, 2001.
57. P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997.
58. J. Stern. A new paradigm for public key identification. *IEEE Transactions on Information Theory*, 42(6):1757–1768, 1996.
59. M. Trolin and D. Wikström. Hierarchical group signatures. In L. Caires, G. F. Italiano, L. Monteiro, C. Palamidessi, and M. Yung, editors, *ICALP 2005*, volume 3580 of *LNCS*, pages 446–458. Springer, 2005.