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, 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, 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.

1 Introduction

The study of group encryption - the encryption analogue of group signatures [17] - was initiated by Kiayias, Tsiounis and Yung (KTY) [29] 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 [29] and subsequent work [15,1,36,32], 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. [29], GE implies hierarchical group signatures [50] - a proper generalization of group signatures [4,5].

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 [21]. Cathalo et al. [15] 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. [32].

Libert et al. [36] enriched the KTY model of GE by introducing a refined tracing mechanism inspired by that of traceable signatures [28]. 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. [26] 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 [17], 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 [25], functional [8] and fully-homomorphic [22] 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 [29], 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 [36], their refined tracing method might pave the way for a mechanism akin to verifier-local revocation [9], 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 [29] 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 [15,1,36] and a Short-Integer-Solution relation is used in [32] 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 [48] is the lattice-based construction from [32]. 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 [10]. Beyond the usual joining algorithm of the KTY model [29], 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 [13] and accumulators [14]. As noted in [10], 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 [29]. 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 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.

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

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

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 [49] to prove knowledge of such $\mathbf{g}, \mathbf{h}$ and we adapt Libert et al.'s technique [32] 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 [37].

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 [49] 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 [46] that satisfies CPA-security and the Naor-Yung transformation [44]. 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 [38]. This amounts to replacing the signature scheme by an accumulator scheme [6] 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. [45].

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 [45,37], (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 [47] from [32] 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 [38], 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 [32] 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 [32], 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 [13,14,9] to address this problem, and efficient pairing-based constructions supporting both dynamic joining and efficient revocation were given in [43,35,34]. Bootle et al. [10] pointed out a few shortcomings of previous models, and put forward robust security notions for fully dynamic group signatures. Here, we adapt the [10] 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 [24] and argument [23,12] 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 [49]. Variants of Stern's protocol have been employed to design privacy-preserving constructions, such as proofs of plaintext knowledge [42], linear-size ring signatures [40,18,41,11], linear-size and sublinear-size group signatures [20,2], proofs of valid openings for commitments and proofs for general relations [27]. Recently, Nguyen et al. [45] 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. In Section 2, we recall the background on Stern-like protocols and previous techniques for designing zero-knowledge protocols in Stern's framework. In Section 3, we present our ZK argument for a quadratic relation. This is crucial for proving the permissive relation in Section 4 - where we also present the strategies for proving the prohibitive relation. Section 5 introduces the model and security requirements of FDGE. Next, we present our code-based instantiation of FDGE in Section 6. Due to space limit, several supplementary materials are deferred to the full version of the paper.

2 Preliminaries

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.

2.1 Stern-like Protocols

The statistical zero-knowledge arguments of knowledge presented in this work are Stern-like [49] protocols. In particular, they are Σ-protocols in the generalized sense defined in [27,7] (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 [31], 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:

$$\begin{cases} \mathbf{w} \in \mathsf{VALID} \iff \Gamma_\phi(\mathbf{w}) \in \mathsf{VALID}, \\ \text{If } \mathbf{w} \in \mathsf{VALID} \text{ and } \phi \text{ is uniform in } \mathcal{S}, \text{ then } \Gamma_\phi(\mathbf{w}) \text{ is uniform in } \mathsf{VALID}. \end{cases} \quad (1)$$

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

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

The conditions in (1) 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 $\mathcal{V}$ can be found in [31] or the full version of the paper. The resulting protocol is a statistical ZKAoK with perfect completeness, soundness error 2/3, and communication cost $\mathcal{O}(L)$.

2.2 Previous Extension and Permutation Techniques

In this section, we first recall the permutation technique that is designed to prove knowledge of a binary vector of fixed hamming weight, which originates from Stern [49].

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 [7].

$$\mathbf{e} \in \mathsf{B}(n, \omega) \iff \sigma(\mathbf{e}) \in \mathsf{B}(n, \omega), \quad (2)$$

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 (2), 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, which were presented in [33].

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

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.

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

– 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 $\mathbf{x} \in \{0,1\}^n$,

$$\mathbf{x} = \mathbf{I}_n^* \cdot \mathsf{Encode}(\mathbf{x}). \tag{3}$$

– 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:

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

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

$$\mathbf{z} = \mathsf{Encode}(\mathbf{x}) \iff F_\mathbf{b}(\mathbf{z}) = \mathsf{Encode}(\mathbf{x} \oplus \mathbf{b}). \tag{4}$$

The above equivalence (4) is useful in the Stern's framework [49] 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 (4), 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.

3 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 the following, we present our ZKAoK for a variant of LPN relation, where we consider secret $\mathbf{A} \in \mathbb{Z}_2^{n \times m}, \mathbf{r} \in \mathsf{B}(m, t_r), \mathbf{e} \in \mathsf{B}(n, t)$. Looking ahead, this protocol is crucial in Section 4.2 that proves a message satisfies the permissive relation.

3.1 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 $\mathsf{ext}'(c_i) \overset{\triangle}{=} \mathsf{ext}'(a_i, r_i) = [\overline{a_i} \cdot r_i | a_i \cdot r_i]^\top \in \{0,1\}^2$. 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

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

to a vector

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

- 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.

$$\mathbf{z} = \mathsf{ext}'(\mathbf{a}, \mathbf{r}) \iff \Psi_{\mathbf{b},\sigma}(\mathbf{z}) = \mathsf{ext}'\big(\sigma(\mathbf{a} \oplus \mathbf{b}), \sigma(\mathbf{r})\big). \tag{5}$$

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

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

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:

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

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 $\mathit{\Psi}'_{\mathbf{b},\sigma}$ that transform vector $\mathbf{z} \in \{0,1\}^{2mn}$ of the following form

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

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

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

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 (5) for $\mathbf{a},\mathbf{b} \in \{0,1\}^{mn}$, $\mathbf{r} \in \mathsf{B}(m,t_r)$, $\sigma_r \in \mathsf{S}_m$.

$$\mathbf{z} = \mathsf{expand}'(\mathbf{a},\mathbf{r}) \Longleftrightarrow \mathit{\Psi}'_{\mathbf{b},\sigma_r}(\mathbf{z}) = \mathsf{expand}'\big(\sigma_r^{(n)}(\mathbf{a}\oplus\mathbf{b}), \sigma_r(\mathbf{r})\big). \tag{6}$$

The zero-knowledge argument. We now transform the relation R_{VLPN} to an instance of R_{abstract} such that the equivalences in (1) hold. Write $\mathbf{A} = [\,\mathbf{a}_1 \mid \cdots \mid \mathbf{a}_m\,] \in \mathbb{Z}_2^{n\times m}$ and $\mathbf{r} = [\,r_1 \mid \cdots \mid r_m\,]^\top \in \mathbb{Z}_2^m$, 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:

$$\mathbf{c} = \mathbf{A} \cdot \mathbf{r} \oplus \mathbf{e} \Longleftrightarrow \mathbf{c} = \mathbf{H}'_{n,m} \cdot \mathsf{expand}'(\mathbf{A}, \mathbf{r}) \oplus \mathbf{e}. \tag{7}$$

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 $\mathbf{c} \overset{\triangle}{=} \mathbf{v}_{\mathrm{VLPN}} = \mathbf{M}_{\mathrm{VLPN}} \cdot \mathbf{w}_{\mathrm{VLPN}} \bmod 2$.

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 $\{\mathit{\Gamma}_\phi : \phi \in \mathcal{S}_{\mathrm{VLPN}}\}$ such that the equivalences in (1) 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 $\mathit{\Gamma}_\phi$ 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 $\mathit{\Gamma}_\phi(\widehat{\mathbf{w}}_{\mathrm{VLPN}}) = (\,\mathit{\Psi}'_{\mathbf{b},\sigma_r}(\widehat{\mathbf{z}}) \,\|\, \sigma_e(\widehat{\mathbf{e}})\,)$.

Based on the equivalence observed in (6 and (2), it can be checked that the conditions in (1) 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 run the Stern-like protocol for the reduced statement R_{VLPN} (see the full version). 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.

4 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 ZK that the underlying messages satisfy the given policies.

4.1 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

$$R_{\text{permit}} = \big\{\big((\mathbf{s}_1, \ldots, \mathbf{s}_k), \mathbf{w}\big) \in (\{0,1\}^t)^k \times \{0,1\}^p : \ \exists i \in [1,k] \ \text{ s.t. } \mathbf{s}_i \sqsubset \mathbf{w}\big\}. \quad (8)$$

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

$$\begin{aligned} R_{\text{prohibit}} = \big\{ \ &\big((\mathbf{s}_1, \ldots, \mathbf{s}_k), \mathbf{w}\big) \in (\{0,1\}^t)^k \times \{0,1\}^p : \\ &d_H(\mathbf{s}_i, \mathbf{y}) \geq d, \forall i \in [1,k], \forall \mathbf{y} \sqsubset \mathbf{w}\big\}. \end{aligned} \quad (9)$$

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

4.2 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 $\big((\mathbf{s}_1, \ldots, \mathbf{s}_k), \mathbf{w}\big) \in R_{\text{permit}}$ 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 $\mathbf{W}$. Details are in the full version of this paper. 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.

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, adapted from [37].

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 $t-(t-d) = d$. Details are in the full version of this paper. As a result, we obtain a ZK protocol for R_{prohibit} with communication cost $\mathcal{O}(t\cdot(p-t+1)\cdot k)$. Similarly to the case of R_{permit}, this protocol can serve as a sub-protocol in our FDGE construction of Section 6, allowing us to realize the "prohibitive" filtering policy.

5 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. [29]. Our model is analogous to the fully dynamic group signature one proposed by Bootle et al. [10]. 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 [29], 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 [15]. 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.

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

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

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

$\mathsf{Setup_{GM}}(\mathsf{pp})$ This algorithm is run by the group manger GM. It takes as input the public parameters pp and outputs a key pair $(\mathsf{pk_{GM}}, \mathsf{sk_{GM}})$. 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 $\mathsf{pk}_{\mathcal{R}}$.

$\langle \mathsf{Join}, \mathsf{Issue}(\mathsf{sk_{GM}}) \rangle (\mathsf{pk_{GM}}, \mathsf{info}_{\tau_{\text{current}}})$ This is an interactive protocol securely run between a user and the GM. Both the Join and Issue algorithms takes as inputs $\mathsf{pk_{GM}}$ and $\mathsf{info}_{\tau_{\text{current}}}$ at current time epoch τ_{current} while the the latter algorithm takes $\mathsf{sk_{GM}}$ 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.

$\mathsf{GUpdate}(\mathsf{sk_{GM}}, \mathcal{S}, \mathsf{info}_{\tau_{\text{current}}}, \mathbf{reg})$ This algorithm is run by the GM who will advance the epoch and update the group information. Given the secret key $\mathsf{sk_{GM}}$, 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.

$\mathsf{Enc}(\mathsf{pk_{GM}}, \mathsf{pk_{OA}}, \mathsf{info}_\tau, w, \mathsf{pk}, L)$ 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 [29], 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.

$\mathcal{P}\big(\mathsf{pp}, \mathsf{pk_{GM}}, \mathsf{pk_{OA}}, \mathsf{info}_\tau, \mathsf{pk}_{\mathcal{R}}, x, \psi, L, w, \mathsf{pk}, \mathsf{coins}_\psi\big)$ 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 $\mathsf{pk_{OA}}$ and can be later revealed using the OA's secret key $\mathsf{sk_{OA}}$.

$\mathcal{V}\big((\mathsf{pp}, \mathsf{pk_{GM}}, \mathsf{pk_{OA}}, \mathsf{info}_\tau, \mathsf{pk}_\mathcal{R}, x, \psi, L), \pi_\psi\big)$ This verification algorithm is run by any verifier who on input the tuple $(\mathsf{pp}, \mathsf{pk_{GM}}, \mathsf{pk_{OA}}, \mathsf{info}_\tau, \mathsf{pk}_\mathcal{R}, x, \psi, L)$ 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.

$\mathsf{Open}(\mathsf{info}_\tau, \mathsf{sk_{OA}}, \psi, L)$ This opening algorithm is run by the OA who holds the key $\mathsf{sk_{OA}}$. 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.

$\mathsf{AddU}(\mathsf{sk_{GM}})$ This oracle adds an honest user to the group at current time τ_{current}. It simulates the interactive protocol $\langle \mathsf{Join}, \mathsf{Issue}(\mathsf{sk_{GM}})\rangle(\mathsf{pk_{GM}}, \mathsf{info}_{\tau_{\text{current}}})$ 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 $\mathsf{GUpdate}(\mathsf{sk_{GM}}, \mathcal{S}, \mathsf{info}_{\tau_{\text{current}}}, \mathbf{reg})$. As the algorithm $\mathsf{GUpdate}$, 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 $\mathbf{Expt}^{\mathsf{corr}}_{\mathcal{A}}(1^\lambda)$
    $\mathsf{pp} \leftarrow \mathsf{Setup_{init}}(1^\lambda)$; $(\mathsf{pk_{OA}}, \mathsf{sk_{OA}}) \leftarrow \mathsf{Setup_{OA}}(\mathsf{pp})$; $(\mathsf{pk_{GM}}, \mathsf{sk_{GM}}) \leftarrow \mathsf{Setup_{GM}}(\mathsf{pp})$.
    $(\mathsf{pk}_\mathcal{R}, \mathsf{sk}_\mathcal{R}) \leftarrow \mathsf{G}_\mathcal{R}(1^\lambda)$; $\mathsf{HUL} \leftarrow \emptyset$.
    $(\mathsf{pk}, \tau, x, w, L) \leftarrow \mathcal{A}^{\mathsf{AddU}, \mathsf{GUp}}(\mathsf{pp}, \mathsf{pk_{OA}}, \mathsf{pk_{GM}}, \mathsf{pk}_\mathcal{R})$.
    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.
    $\psi \leftarrow \mathsf{Enc}(\mathsf{pk_{GM}}, \mathsf{pk_{OA}}, \mathsf{info}_\tau, w, \mathsf{pk}, L)$.
    $\pi_\psi \leftarrow \mathcal{P}\big(\mathsf{pp}, \mathsf{pk_{GM}}, \mathsf{pk_{OA}}, \mathsf{info}_\tau, \mathsf{pk}_\mathcal{R}, x, \psi, L, w, \mathsf{pk}, \mathsf{coins}_\psi\big)$.
```

$w' \leftarrow \mathsf{Dec}(\mathsf{info}_\tau, \mathsf{sk}, \psi, L)$; $\mathsf{pk}' \leftarrow \mathsf{Open}(\mathsf{info}_\tau, \mathsf{sk}_{\mathsf{OA}}, \psi, L)$.
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.

5.1 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 $\mathbf{Expt}_{\mathcal{A}}^{\mathsf{sec}-b}(1^\lambda)$ 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 $\mathbf{Expt}_{\mathcal{A}}^{\mathsf{sec}-b}(1^\lambda)$, 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 RevealU oracle and to update the group information, insofar as info and **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 $\mathbf{Adv}_{\mathcal{A}}^{\mathsf{sec}-b}(1^\lambda) = |\Pr[\mathbf{Expt}_{\mathcal{A}}^{\mathsf{sec}-1}(1^\lambda) = 1] - \Pr[\mathbf{Expt}_{\mathcal{A}}^{\mathsf{sec}-0}(1^\lambda) = 1]|$. A FDGE satisfies message secrecy if, for any PPT adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ is negligible in λ.*

```
Experiment $\mathbf{Expt}_{\mathcal{A}}^{\mathsf{sec}-b}(1^\lambda)$
    $\mathsf{pp} \leftarrow \mathsf{Setup}_{\mathsf{init}}(1^\lambda)$; $(\mathsf{aux}, \mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathsf{OA}}) \leftarrow \mathcal{A}(\mathsf{pp})$; $\mathsf{HUL} \leftarrow \emptyset$, $\mathsf{BUL} \leftarrow \emptyset$.
    Throughout the experiment, if info or 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 $\mathsf{sk}^*$ be the corresponding secret key of $\mathsf{pk}^*$.
    $(\tau^*, \mathsf{pk}^*_{\mathcal{R}}, x^*, w^*, L^*) \leftarrow \mathcal{A}^{\mathsf{DEC}(\mathsf{sk}^*,\cdot)}(\mathsf{aux})$.
    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 $\mathbf{Expt}_{\mathcal{A}}^{\mathsf{anon}-b}(1^\lambda)$ 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 $\mathsf{info}$ 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 $\mathsf{sk}_0^*$ and $\mathsf{sk}_1^*$ be the secret keys of $\mathsf{pk}_0^*$ and $\mathsf{pk}_1^*$, 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 $\eth^* = (\mathsf{pp}, \mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathsf{OA}}, \mathsf{info}_{\tau^*}, \mathsf{pk}_{\mathcal{R}}^*, x^*, \psi^*, L^*, w^*, \mathsf{pk}_b^*, \mathsf{coins}_{\psi^*})$.
    Let $t^* = (\psi^*, \tau^*, L^*)$.
    $b' \leftarrow \mathcal{A}^{\mathcal{P}(\eth^*),\mathsf{DEC}^{\neg t^*}(\mathsf{sk}_0^*,\cdot),\mathsf{DEC}^{\neg t^*}(\mathsf{sk}_1^*,\cdot),\mathsf{OPEN}^{\neg t^*}(\mathsf{sk}_{\mathsf{OA}},\cdot)}(\mathsf{aux}, \psi^*)$.
    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

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

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 $\mathsf{REG}(\mathsf{sk}_{\mathsf{GM}})$, as defined below. In addition, it can remove some users from the group by querying the oracle $\mathsf{GUp}(\cdot)$.

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

6 A Code-Based Fully Dynamic Group Encryption Scheme

To build a code-based FDGE scheme, we require a key-private CCA2-secure encryption scheme [3], 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 [49]. In terms of the encryption scheme, we choose to work with the McEliece cryptosystem [39], specifically the randomized variant from [46]. 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 [44]. Note that there are other CCA2-secure variants of McEliece scheme like [19,16,30]. 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 suggested in [45]. 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 [38], we use an updating algorithm akin to [38] 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 [29], 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.
- Sample a random matrix $\mathbf{B} \xleftarrow{\$} \mathbb{Z}_2^{n \times m}$ that specifies a hash function $h_{\mathbf{B}}$ that will be used build Merkle tree (see the full version of this paper, as well as [45]).
- Pick a statistical hiding and computationally binding commitment scheme $\mathsf{COM} : \{0,1\}^* \to \{0,1\}^n$ like the one in [45, Section 3.1]. This will serve as a building block for the ZK argument systems.
- 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 [21].

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 the full version) 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} (8). 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 the full version) 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 **reg** as $\mathbf{reg}[j][1] := \mathbf{d}; \quad \mathbf{reg}[j][2] := \tau$.
   – 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:

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

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 **reg** from group information $\mathsf{info}_{\tau_{\text{current}}}$ and $\mathsf{info}_{\tau_{\text{new}}}$[8], and vice versa[9].

[8] 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 $\mathbf{reg}[j][3] = \tau_{\text{new}}$.

[9] It is easy to figure out all active users at specific time τ from **reg**, and thus enables verification of well-formedness of info_τ.

$\mathsf{Enc}(\mathsf{pk_{GM}}, \mathsf{pk_{OA}}, \mathsf{info}_\tau, \mathbf{w}, \mathsf{pk}, L)$ $\mathsf{pk_{OA}} = (\mathbf{G}_{\mathsf{oa},0}, \mathbf{G}_{\mathsf{oa},1})$, $\mathsf{pk} = (\mathsf{pk}'_j, \mathsf{bin}(j))$ 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

$$\mathbf{c}_{w,b} = \mathbf{G}_{j,b} \cdot \begin{pmatrix} \mathbf{r}_{w,b} \\ \mathbf{w} \end{pmatrix} \oplus \mathbf{e}_{w,b} \in \mathbb{Z}_2^n. \quad (10)$$

   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 $\mathsf{pk_{OA}} = (\mathbf{G}_{\mathsf{oa},0}, \mathbf{G}_{\mathsf{oa},1})$.
   - 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

$$\mathbf{c}_{\mathsf{oa},b} = \mathbf{G}_{\mathsf{oa},b} \cdot \begin{pmatrix} \mathbf{r}_{\mathsf{oa},b} \\ \mathsf{bin}(j) \end{pmatrix} \oplus \mathbf{e}_{\mathsf{oa},b} \in \mathbb{Z}_2^n. \quad (11)$$

   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 $\mathsf{bin}(j)$. The proof employs a Stern-like interactive ZK protocol on public input $(\mathbf{G}_{\mathsf{oa},0}, \mathbf{G}_{\mathsf{oa},1}, \mathbf{c}_w, \mathbf{c}_{\mathsf{oa}}, L)$ and secret input $(\mathbf{G}_{j,0}, \mathbf{G}_{j,1}, \mathbf{w}, \mathsf{bin}(j), \mathbf{r}_{w,0}, \mathbf{r}_{w,1}, \mathbf{e}_{w,0}, \mathbf{e}_{w,1}, \mathbf{r}_{\mathsf{oa},0}, \mathbf{r}_{\mathsf{oa},1}, \mathbf{e}_{\mathsf{oa},0}, \mathbf{e}_{\mathsf{oa},1})$, described in detail in the full version. The interactive protocol is repeated κ times to achieve negligible soundness error and made non-interactive via Fiat-Shamir transform [21]. The resulting 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

$$\mathsf{coins}_\psi = (\mathbf{r}_{w,0}, \mathbf{r}_{w,1}, \mathbf{e}_{w,0}, \mathbf{e}_{w,1}, \mathbf{r}_{\mathsf{oa},0}, \mathbf{r}_{\mathsf{oa},1}, \mathbf{e}_{\mathsf{oa},0}, \mathbf{e}_{\mathsf{oa},1}). \quad (12)$$

$\mathcal{P}(\mathsf{pp}, \mathsf{pk_{GM}}, \mathsf{pk_{OA}}, \mathsf{info}_\tau, S, \psi, L, \mathbf{w}, \mathsf{pk}, \mathsf{coins}_\psi)$ Let coins_ψ be of the form (12) 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 (10) and (11) 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.

The proof employs a Stern-like interactive ZK protocol on public input $\big(\mathbf{B}, \mathsf{pk}_{\mathsf{OA}}, \mathbf{u}_\tau, S, \psi, L\big)$ and secret input $(\mathbf{w}, \mathsf{pk}, \mathsf{coins}_\psi, w^{(j)})$, provided in the full version. To achieve negligible soundness error, the protocol is repeated κ times. Then the Fiat-Shamir heuristic [21] is applied. 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 the full version), 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 $\mathcal{O}(\lambda^2)$ and size of proof π_ψ is $\omega(\log \lambda)\cdot\mathcal{O}(\lambda^2 + \ell\cdot\lambda)$.

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 5, 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 1, we prove the given FDGE satisfies the proposed security requirements in Section 5.1.

Theorem 1. *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.*

Due to space limit, details of the proof are provided in the full version.

Acknowledgements

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 *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. M. Bellare, A. Boldyreva, A. Desai, and D. Pointcheval. Key-privacy in public-key encryption. In *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 566–582. Springer, 2001.
4. M. Bellare, D. Micciancio, and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 614–629. Springer, 2003.
5. M. Bellare, H. Shi, and C. Zhang. Foundations of group signatures: The case of dynamic groups. In *CT-RSA 2005*, volume 3376 of *LNCS*, pages 136–153. Springer, 2005.
6. J. C. Benaloh and M. de Mare. One-way accumulators: A decentralized alternative to digital sinatures (extended abstract). In *EUROCRYPT 1993*, volume 765 of *LNCS*, pages 274–285. Springer, 1993.
7. 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 *ASIACRYPT 2014*, volume 8873 of *LNCS*, pages 551–572. Springer, 2014.
8. D. Boneh, A. Sahai, and B. Waters. Functional encryption: a new vision for public-key cryptography. *Commun. ACM*, 55(11):56–64, 2012.
9. D. Boneh and H. Shacham. Group signatures with verifier-local revocation. In *CCS 2004*, pages 168–177. ACM, 2004.
10. J. Bootle, A. Cerulli, P. Chaidos, E. Ghadafi, and J. Groth. Foundations of fully dynamic group signatures. In *ACNS 2016*, volume 9696 of *LNCS*, pages 117–136. Springer, 2016.
11. P. Branco and P. Mateus. A code-based linkable ring signature scheme. In *ProvSec 2018*, volume 11192 of *LNCS*, pages 203–219. Springer, 2018.
12. G. Brassard, D. Chaum, and C. Crépeau. Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.*, 37(2):156–189, 1988.
13. E. Bresson and J. Stern. Efficient revocation in group signatures. In *PKC 2001*, volume 1992 of *LNCS*, pages 190–206. Springer, 2001.

14. J. Camenisch and A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *CRYPTO 2002*, volume 2442 of *LNCS*, pages 61–76. Springer, 2002.
15. J. Cathalo, B. Libert, and M. Yung. Group encryption: Non-interactive realization in the standard model. In *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 179–196. Springer, 2009.
16. P. Cayrel, G. Hoffmann, and E. Persichetti. Efficient implementation of a cca2-secure variant of mceliece using generalized srivastava codes. In *PKC 2012*, volume 7293 of *LNCS*, pages 138–155. Springer, 2012.
17. D. Chaum and E. van Heyst. Group signatures. In *EUROCRYPT 1991*, volume 547 of *LNCS*, pages 257–265. Springer, 1991.
18. L. Dallot and D. Vergnaud. Provably secure code-based threshold ring signatures. In *IMACC 2009*, volume 5921 of *LNCS*, pages 222–235. Springer, 2009.
19. 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.
20. M. F. Ezerman, H. T. Lee, S. Ling, K. Nguyen, and H. Wang. A provably secure group signature scheme from code-based assumptions. In *ASIACRYPT 2015*, volume 9452 of *LNCS*, pages 260–285. Springer, 2015.
21. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO 1986*, volume 263 of *LNCS*, pages 186–194. Springer, 1986.
22. C. Gentry. Fully homomorphic encryption using ideal lattices. In *STOC 2009*, pages 169–178. ACM, 2009.
23. O. Goldreich, S. Micali, and A. Wigderson. How to prove all np-statements in zero-knowledge, and a methodology of cryptographic protocol design. In *CRYPTO 1986*, volume 263 of *LNCS*, pages 171–185. Springer, 1986.
24. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.
25. V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *CCS 2006*, pages 89–98. ACM, 2006.
26. M. Izabachène, D. Pointcheval, and D. Vergnaud. Mediated traceable anonymous encryption. In *LATINCRYPT 2010*, volume 6212 of *LNCS*, pages 40–60. Springer, 2010.
27. A. Jain, S. Krenn, K. Pietrzak, and A. Tentes. Commitments and efficient zero-knowledge proofs from learning parity with noise. In *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 663–680. Springer, 2012.
28. A. Kiayias, Y. Tsiounis, and M. Yung. Traceable signatures. In *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 571–589. Springer, 2004.
29. A. Kiayias, Y. Tsiounis, and M. Yung. Group encryption. In *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 181–199. Springer, 2007.
30. K. Kobara and H. Imai. Semantically secure mceliece public-key cryptosystems-conversions for mceliece PKC. In *PKC 2001*, volume 1992 of *LNCS*, pages 19–35. Springer, 2001.
31. B. Libert, S. Ling, F. Mouhartem, K. Nguyen, and H. Wang. Signature schemes with efficient protocols and dynamic group signatures from lattice assumptions. In *ASIACRYPT 2016*, volume 10032 of *LNCS*, pages 373–403, 2016.
32. 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.

33. B. Libert, S. Ling, K. Nguyen, and H. Wang. Zero-knowledge arguments for lattice-based accumulators: Logarithmic-size ring signatures and group signatures without trapdoors. In *EUROCRYPT 2016*, volume 9666 of *LNCS*, pages 1–31. Springer, 2016.
34. B. Libert, T. Peters, and M. Yung. Group signatures with almost-for-free revocation. In *CRYPTO 2012*, volume 7417 of *LNCS*, pages 571–589. Springer, 2012.
35. B. Libert, T. Peters, and M. Yung. Scalable group signatures with revocation. In *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 609–627. Springer, 2012.
36. B. Libert, M. Yung, M. Joye, and T. Peters. Traceable group encryption. In *PKC 2014*, volume 8383 of *LNCS*, pages 592–610. Springer, 2014.
37. S. Ling, K. Nguyen, D. Stehlé, and H. Wang. Improved zero-knowledge proofs of knowledge for the ISIS problem, and applications. In *PKC 2013*, volume 7778 of *LNCS*, pages 107–124. Springer, 2013.
38. 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.
39. R. J. McEliece. A public-key cryptosystem based on algebraic coding theory. *Coding Thv*, 4244:114–116, 1978.
40. C. A. Melchor, P. Cayrel, and P. Gaborit. A new efficient threshold ring signature scheme based on coding theory. In *PQCrypto 2008*, volume 5299 of *LNCS*, pages 1–16. Springer, 2008.
41. 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.
42. K. Morozov and T. Takagi. Zero-knowledge protocols for the mceliece encryption. In *ACISP 2012*, volume 7372 of *LNCS*, pages 180–193. Springer, 2012.
43. T. Nakanishi, H. Fujii, Y. Hira, and N. Funabiki. Revocable group signature schemes with constant costs for signing and verifying. In *PKC 2009*, volume 5443 of *LNCS*, pages 463–480. Springer, 2009.
44. M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *STOC 1990*, pages 427–437. ACM, 1990.
45. K. Nguyen, H. Tang, H. Wang, and N. Zeng. New code-based privacy-preserving cryptographic constructions. In *ASIACRYPT 2019*, volume 11922 of *LNCS*, pages 25–55. Springer, 2019.
46. 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.
47. O. Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6):34:1–34:40, 2009.
48. P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997.
49. J. Stern. A new paradigm for public key identification. *IEEE Transactions on Information Theory*, 42(6):1757–1768, 1996.
50. M. Trolin and D. Wikström. Hierarchical group signatures. In *ICALP 2005*, volume 3580 of *LNCS*, pages 446–458. Springer, 2005.