Dynamic Group Signature Scheme using Lattices

Fabrice Mouhartem[†] — **Supervisor:** Benoît Libert[†]

[†]École Normale Supérieure de Lyon, Laboratoire de l'Informatique et du Parallélisme

M2 Internship – 2015 – 12th June 2015

Abstract

Lattice-based cryptography is a field of research that has been very active in the last decade. It offers *expressiveness* along with *asymptotic efficiency* and seems to the best of our knowledge to be resistant to quantum computers attacks. *Advanced cryptography* would then benefit from being designed from lattice assumptions. Along those primitives, there is one that grabs our interest: *group signatures*; it is namely a protocol that allows an user to sign on behalf of a group while keeping his anonymity. Nowadays all lattice-based group signatures are built on the static model, where the group is fixed at its creation and cannot be changed without rerunning all the setup again (and redistributing the keys). This is why we construct during this internship a *dynamic group signature* scheme based on lattices that is somewhat efficient (as efficient as the best known algorithms for weaker purpose).

Table of Contents

1 Introduction

Cryptology is the study of methods to ensure the existence of secure communication. Its goals are – among other things – to provide tools to guarantee *data-integrity*, *privacy* and *authentications*. Since Caesar cipher, many studies have been done to achieve this goals and lead to nowadays advanced cryptography primitives.

A so called *advanced primitive* is a method to solve more advanced problems than historical ones. Two important cryptographic primitives are public-key encryption and digital signature schemes. They are overly used in applied cryptography such as opening a trusted channel and ciphering in TLS/SSL. But there are also more complex problems where those simple primitives are not sufficient. For this kind of applications, we sometimes rely on more advanced tools like *identity-based encryption*, *zero-knowledge proofs* [21, 22], *multi-party computation protocols* or *privacy-enhancing cryptographic primitives*.

Group signature [19] is one of those advanced primitive. It aims to provide a way to guarantee that a message was sent by a group member – *authentication and data integrity* – without leaking any information about which member signed it – *privacy* – unless an *opening authority* decides to open the signature [7]. More precisely, group signatures allow members of a population of users (which is administered by a *group manager*) to sign anonymously messages in the name of that population. The signature verifier will be convinced that the signature comes from some group member, but without knowing which one. At the same time, group members remain accountable for the messages they sign since, if necessary, an authority is able to determine the signer's identity using some trapdoor information. The latter operation is known as the *signature opening* operation.

Group signatures have many applications. A basic example is *intelligent cars* [24]. Let us assume that we have cars that can communicate with each other, for instance to transmit traffic information. We want that no malicious opponent is able to say "Hello, I am a car equipped with a transmitter for the system and an accident occurred in this road". A first solution can be to sign every messages, but this suffers from two major issues. At first there is a verification key database of N cars to be kept updated, and then anyone listening the traffic can trace a given car. A solution to this problem is to use *group signatures*, as suggested in [12]: at setup time, the *group manager* – here the car company – builds a group public key. Thereafter, each *member* – the cars – joins interactively the group by communicating with the group manager. Hence, each car can attest its messages without explicitly identify itself, and every car can verify that it is a car who send the message. Moreover, in case of accident, an *opening authority* – the police – can open the signatures to know who sent messages (if there is a fake message for instance).

There are many other possible applications to group signatures, such as trusted computing platforms or protecting the privacy of users in public transportation: by issuing a group signature on the current date/time, we can prove to the **TCL** (*Transports en Commun Lyonnais*, the public transportation company in Lyon) that we have a valid subscription without letting the system link our metro rides together or infer information about their frequency. In a more pragmatic point of view, we can just think the gate opening in a building: employees of a company can prove that they are the right to enter the company's building in the morning without revealing to the managers when they arrive or leave.

In the last five years, the research community has considered the problem of designing group signature schemes whose security relies on the hardness of lattice problems. One of the motivations for this is the threat of quantum computers, which would break all known efficient group signature constructions proposed before 2010. It is then an important task to construct group signatures that will resist in the presence of quantum adversaries.

Nowadays, the most efficient constructions based on lattices are for static group [31, 35, 37] only. This means that, after a setup phase during which the group manager gives private keys to all group members, no new group member can join the group. In other words, the population of the group is frozen after the setup phase. Therefore, we are not able to add a group member without re-starting the setup phase! As a first attempt to deal with dynamically-changing groups, Langlois *et al.* [33] proposed a group signature scheme that supports *verifier-local revocation* [11], which is a revocation mechanism where the revocation messages are only sent to signature verifiers. There are protocols that achieve quite practical dynamic group signatures [12, 23], but they are based on discrete-logarithm-related hardness assumptions, which are known not to resist quantum attacks. We will further discuss about it in section 2.

It is important to notice that *dynamic* group signatures are not just a simple extension of *static* group signatures since dynamic schemes involve two distinct authorities (instead of one in static groups) and should satisfy additional security properties. For example, in dynamic group signatures, the group manager should not be able to sign messages that will be opened to honest group members. We will explain these difficulties in more details in section 2. While we will try to start from a static scheme and adapt it to the dynamic setting, we will see that the problem is really not straightforward and we will discuss on the difficulties.

1.1 Our Contributions

Initially this internship was entitled: "Designing Dynamic Group Signature Scheme and Adaptive Oblivious Transfer using Lattices". *Oblivious Transfer schemes* are used to make anonymous data transfers from a database. Our first idea was that designing a dynamic group signature would gives us techniques to go forward oblivious transfer. In ACNS'15, however, Blazy and Chevalier [9] proposed a partial solution to the oblivious transfer problem. In this internship, we thus focused on dynamic group signature. Our main contribution is a first example of lattice-based group signatures for dynamic groups, where new group members can be added at any time.

One can think that making a *static* group signature scheme into a *dynamic* scheme can easily be done due to the similarities of the schemes and security requirements.

However, designing a lattice-based dynamic group signature scheme has been left as an open problem for 4 years. One reason is that in the dynamic setting, the opening authority (which can identify the author of any signature if needed) and the group manager (which delivers group membership certificates to users) are distinct entities, and they thus need to have their own secrets. This leads to more complex structures to play with in order to keep the scheme secure. Another difficulty is that dynamic group signatures involve additional security notions that are not present in the security models of static group signatures. A requirement of dynamic group signatures is that, even if the group manager and the opening authority collude together and with dishonest group members, they will not be able to create signatures that will be opened to some honest group member who did not sign the corresponding message. For this purpose, each group member needs a *membership secret* sec which is used for generating signatures and which remains unknown to the group manager and the opening authority. When the user joins the system, he chooses a membership secret for which he sends a corresponding public value v to the group manager (for instance the syndrome of a short vector according to a given matrix). The latter public value v is then certified by means of a *membership certificate* cert which is generated by the group manager to certify that the user's public value (and then the underlying membership secret sec) really belongs to some group member. When the user signs a message, he generally proves the knowledge of both cert and sec (as well as v) without revealing them and without revealing anything except that they form a valid credential.

The fact that only the user knows sec is the reason that prevents the group manager and the opening authority from generating a signature for which the opening operation reveals the public value v. In the lattice setting, the difficulty is to find a way to define a membership secret that is properly bound to some membership certificate.

In this internship, we solved this problem by adapting a recent construction proposed by Ling, Nguyen, and Wang [35] in the setting of static groups.

The rest of the report is organized as follows: we start with a state of the art of group signatures and the lattice-based cryptography around it. Then we introduce the concepts we will work with in section 3, and finally we present our protocol and discuss about it.

2 State of the art

As depicted in the title of the section, we will have a look at the state of the art in group signatures, especially in the dynamic setting.

Group signature schemes were first introduced by Chaum and Van Heyst [19] in 1991. In this setting, there are a group manager and a certain number of group members. To this group is associated an unique public key gpk and each group member i is given a secret key $\mathsf{gsk}[i]$ on which he can produce a valid signature with respect to gpk. The requirements in [19] is that the group manager has the knowledge of a secret key gmsk that can be used to extract the identity of the signer of a given signature that is valid on gpk. On the other hand you should not be able to open a signature without the secret key gmsk. Chaum and Van Heyst [19] then describe the notions on *traceability* and *anonymity* respectively. We then had at this time the basic of group signatures as it is common to see.

Since then more requirements were added on top of this basic core. For instance group signature standard requirements as *unforgeability*, and also requirements that are more specific to the group setting like *exculpability* [4], *unlinkeability* or *framing resistance* [20]. Hence it ended in a chaos of different definitions that claim to be better than others and no accepted definition of *group signatures*.

This is why in 2003, Bellare, Micciancio, and Warinschi [7] proposed a clean setting of group signatures to go toward *provable security* that is nowadays admitted as the formal definition of group signatures. They summed up those security requirements into *correctness*, *full-traceability* and *fully-anonymity*. And evocate in the last section [7, sec. 5] the possibility to extend their scheme to the dynamic setting.

In 2005, Bellare, Shi, and Zhang [8] had the same work done for the dynamic setting to give a formalism for dynamic group signatures. The reason is that beyond the fact that it is costly to rerun the setup at each join, it requires a high level of trust to give to the group manager, which one may not want to afford. In the dynamic group setting, there are two authorities: the group manager (the *issuer*) and the opening manager (the *opener*). The only trusted phase is the setup phase were the group public key gpk, and the opening key $\mathscr{S}_{\mathsf{OA}}$. It is mandatory to have two authorities to provides more robustness against *authorities corruption*. They gives three security requirements: anonymity, traceability and non-frameability (the same as in section 1.1).

In 2006, Kiayias and Yung [30] gave an equivalent framework in term of security for dynamic group signature scheme that we will use in this report (only for convenience). The difference lies in the algorithm descriptions and in the security requirements (even if they are close). The requirements in this scheme: anonymity, mis-identification (equivalent to traceability) and non-frameability. In the algorithm description, there is no judge algorithm which goal is to verify the correctness of the opening, because there is no proof of correct opening in this model.

Many group signature schemes has been proposed, based on traditional number-theory assumptions, such as [16, 5, 12]. But if they can be very efficient such as Ateniese *et al.* [5] protocol or Boneh *et al.* [12] scheme, they are sensible to quantum computers' attack [39]. Designing a group signature that is based on lattice assumptions has been a quite active topic lately. In the most significant changes: in 2010, Gordon *et al.* [28] proposed the first group signature scheme based on lattice assumptions where signature size is linear in the number of group member (more precisely $N \cdot \widetilde{\mathscr{O}}(n^2)$ where n is the security parameter and N the size of the group); in 2013, Laguillaumie *et al.* [31] gave a construction where signature size has been shrunk to logarithmic size (namely $\log N \cdot \widetilde{\mathscr{O}}(n)$). Since then, most of the works has been done to reduce the group public key size to constant size [37] (however, it still requires about 1.5 GB for the public key under classical security parameters, which is far for being usable, for instance in embedded systems such as a smart card for public transportation), or reduce the hardness assumptions [35].

As far as I know, today there is no dynamic scheme based on post-quantum assumptions.

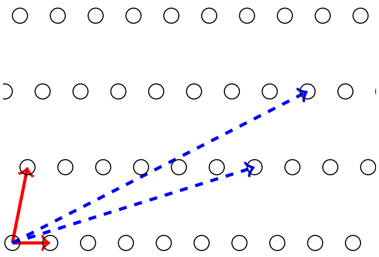

Figure 1: A lattice and two different basis

3 Definitions

In this part, we will introduce the different definitions that will be used all along this report. We will firstly present the lattices and the associated security assumption, then our framework along with its security notions.

Notations. All vectors will be written in bold lower-case letters, bold upper-case letters will be used for matrices. If $\mathbf{b}$ and $\mathbf{c}$ are two vectors of compatible dimensions, their inner product will be denoted by $\langle \mathbf{b}, \mathbf{c} \rangle$. Furthermore, if $\mathbf{b} \in \mathbb{R}^n$, its euclidean norm will be denoted by $\|\mathbf{b}\|$. This notation is extended to any matrix $\mathbf{B} \in \mathbb{R}^{m \times n}$ with columns $(\mathbf{b}_i)_{i \leq n}$ by $\|\mathbf{B}\| = \max_{i \leq n} \|b_i\|$. If $\mathbf{B}$ is full-rank, we let $\widetilde{\mathbf{B}}$ denote its Gram-Schmidt orthogonalisation.

3.1 Lattices

Lattices were first used in cryptography by Regev [38] in 2005. He introduced the *learning with errors* (LWE) problem which has been massively used to construct cryptographic primitives [32].

Lattice based cryptography offers the advantage of an expressive system – which allows us to build advanced primitives – along with asymptotic efficiency.

3.1.1 Lattices Properties

Definition 1 (Lattices)**.** A lattice Λ is the set of all integer linear combinations of some linearly independent vectors, called a basis, $(\mathbf{b}_i)_{i \leq n}$ belonging to $\mathbb{R}^n$.

A basis is not unique, as shown in figure 1, and that is intuitively what makes it expressive and hard as we will later discuss in section 3.1.2.

Classically defined lattices we will use on this report are:

Definition 2 (Classical Lattices)**.** For integer q prime, matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ and vector $\mathbf{u} \in \mathbb{Z}_q^n$, define:

$$\Lambda_q^{\perp}(\mathbf{A}) = \left\{ \mathbf{x} \in \mathbb{Z}^m : \mathbf{x}^T \cdot \mathbf{A} = 0 \bmod q \right\}$$
$$\Lambda_q^{\mathbf{u}}(\mathbf{A}) = \left\{ \mathbf{x} \in \mathbb{Z}^m : \mathbf{x}^T \cdot \mathbf{A} = \mathbf{u}^T \bmod q \right\}$$

Definition 3 (Gaussian distribution over a lattice)**.** For a lattice Λ and a real $\sigma > 0$ we define the *Gaussian distribution* centered in 0 over the lattice Λ and parameter σ by $D_{\Lambda,\sigma}[\mathbf{b}] \sim \exp\left(-\pi \|\mathbf{b}\|^2 / \sigma^2\right)$ for all vector $\mathbf{b}$ in Λ.

We will massively use the following lemma, stating that a vector sampled from $D_{\Lambda,\sigma}$ is short with overwhelming probability:

Lemma 1 ([6, Le. 1.5])**.** *For any lattice* $\Lambda \subseteq \mathbb{R}^n$ *and* $\sigma > 0$, *we have* $\Pr_{\mathbf{b} \hookleftarrow D_{\Lambda,\sigma}} [\|\mathbf{b}\| \leq \sqrt{n}\sigma] \geq 1 - 2^{-\Omega(n)}$.

From a computational point of view, Gentry, Peikert, and Vaikuntanathan [26] show that discrete Gaussian distribution over a lattice can be sampled efficiently given a short basis of the lattice:

Lemma 2 ([14, Le. 2.3])**.** *There exists a probabilistic polynomial time (*PPT*) algorithm* $\mathsf{GPVSample}$ *that takes as inputs a basis* $\mathbf{B}$ *of a lattice* $\Lambda \subseteq \mathbb{Z}^n$ *and a rational* $\sigma \geq \|\widetilde{\mathbf{B}}\| \cdot \Omega(\sqrt{\log n})$*, and outputs vectors* $\mathbf{b} \in \Lambda$ *with distribution* $D_{\Lambda,\sigma}$*.*

We also need an equivalent algorithm stating that the trapdoor $\mathbf{T_A} \in \mathbb{Z}_q^{m\times m}$ of a matrix $\mathbf{A} \in \mathbb{Z}_q^{m\times n}$ allows us to sample a short vector $\mathbf{x} \in \mathbb{Z}_q^m$ such that $\mathbf{x}^T\mathbf{A} = \mathbf{u}^T \mod q$ for a given vector $\mathbf{u} \in \mathbb{Z}_q^n$.

Lemma 3 ([1, Le. 8])**.** *There exists a* PPT *algorithm* $\mathsf{SamplePre}(\mathbf{A}, \mathbf{T_A}, \mathbf{u}, \sigma)$ *that returns a vector* $\mathbf{x} \in \Lambda_q^{\mathbf{u}}(\mathbf{A})$ *sampled from a distribution statistically close to* $D_{\Lambda_q^{\mathbf{u}}(A),\sigma}$ *whenever* $\Lambda_q^{\mathbf{u}}(\mathbf{A})$ *is non empty.*

3.1.2 Security Assumptions

We have already mentioned the LWE problem as a standard lattice hardness assumption.

Definition 4 (Learning With Errors [38])**.** For an integer $p = \mathsf{Poly}(n)$ and a distribution χ on $\mathbb{Z}_p$. The problem is the following: for any vector $\mathbf{s} \in \mathbb{Z}_p^n$, given arbitrarily many samples from the LWE distribution, namely $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s}\rangle + e)$ with $\mathbf{a}$ uniform in $\mathbb{Z}_p^n$ and e samples from χ, the goal is to find $\mathbf{s}$ for the *search LWE problem*, or distinguish this distribution from $\mathscr{U}\left(\mathbb{Z}_p^n \times \mathbb{Z}_p\right)$ for the *decisional LWE problem*.

We can notice that search LWE and decisional LWE are equivalent [38], and that the distribution χ is usually a Gaussian distribution as explained in definition 3.

Remark 1. It is customary in lattice-based cryptography whereas classical cryptography to use as a security parameter the dimension n of the LWE secret instead of the security level in classical cryptography (namely the logarithm of the number of minimum elementary steps to solve the problem of the hardness assumption). The main reason is that the hardness of the LWE problem depends on three parameters: n, q, α where α conditions the standard deviation of χ. Taking $q = \mathsf{Poly}(n)$ and $\alpha^{-1} = \mathsf{Poly}(n)$ leads to best known attacks on LWE to be exponential in n, which means that we can approximate λ by n. This choice of parameters usually gives reasonable size range and are sufficient for most of the purpose in cryptography.

In the following, we select q to be exponentially large in n and not polynomially. This lead to a less efficient protocol, but easier to prove, as discussed in section 4.1. One interesting question could be to go down to polynomially large parameters.

We will mainly rely in this report on the hardness assumption of *Inhomogeneous Short Integer Solution* (ISIS).

Definition 5 (ISIS problem [26, Def. 5.6])**.** The *inhomogeneous short integer solution* problem $\mathsf{ISIS}_{m,q,\beta}$ is defined as follows: given an integer q, a matrix $\mathbf{A} \in \mathbb{Z}_q^n$, a syndrome $\mathbf{u} \in \mathbb{Z}_q^n$ and a real β, find an integer vector $\mathbf{e} \in \mathbb{Z}^m$ such that $\mathbf{Ae} = \mathbf{u} \mod q$ and $\|\mathbf{e}\| \leq \beta$.

Informally it is finding a short vector in the lattice $\Lambda^{\perp}(\mathbf{A})$. Finding a vector in this lattice is just solving a linear algebra system, which is polynomial. But adding the shortness constraints harden the problem. The basic idea is that if we have a short basis of the lattice, finding a short vector in it is easy, we just have to take small linear combinations of the short vectors. But having a long and not orthogonal basis as the blue dashed basis in figure 1 makes this search much longer.

Gentry *et al.* [26, Sec. 9] gave a reduction from worst-case lattices problems to $\mathsf{SIS}_{m,q,\beta}$ (the special case where $\mathbf{u}$ in ISIS is 0) with approximation factor $\gamma = \widetilde{\mathscr{O}}(\beta\sqrt{n})$. In other words it means that solving SIS is an hard problem.

Then we have the following: if we have a *short basis*, we can find a short vector in a lattice, if we don't it is a *hard problem*. This statement means that we have a breeding ground for cryptography: we have a hard problem that becomes easy in the presence of a secret information: a *trapdoor*.

3.1.3 Algorithmic Point of View

We will also need some efficient algorithms to manipulate the matrices as lattices.

First of all, we must be able to construct random matrices along with a short basis for its orthogonal lattice $\Lambda_q^\perp$, which has been described by Alwen and Peikert [2].

Lemma 4 ([2, Th. 3.2]). *There exists a* PPT *algorithm* $\mathsf{TrapGen}$ *that takes as inputs* 1^n, 1^m *and an integer* $q \geq 2$ *with* $m \geq \Omega(n \log q)$, *and outputs a matrix* $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ *and a basis* $\mathbf{T_A}$ *of* $\Lambda_q^\perp(\mathbf{A})$ *such that* $\mathbf{A}$ *is within statistical distance* $2^{-\Omega(n)}$ *to* $U(\mathbb{Z}_q^{m \times n})$, *and* $\|\widetilde{\mathbf{T_A}}\| \leq \mathscr{O}(\sqrt{n \log q})$.

We can notice that Lemma 4 is often combined with the sampler from Lemma 2. Micciancio and Peikert [36] proposed a more efficient approach in 2012 to do this combined task, which should be preferred in practice but, for the sake of simplicity, we will present our scheme using $\mathsf{TrapGen}$.

More algebraically, we may want to be able to manipulate matrices and append more verification information (for instance for the Boyen's signature [13] we use as a building block for our protocol). Which can be seen as an extension of the initial lattice. For this purpose, we may use the following algorithm:

Lemma 5 ([18, Le. 3.2]). *There exists a* PPT *algorithm* $\mathsf{ExtBasis}$ *that takes as inputs a matrix* $\mathbf{B} \in \mathbb{Z}_q^{m' \times n}$ *whose first* m *rows span* $\mathbb{Z}_q^n$, *and a basis* $\mathbf{T_A}$ *of* $\Lambda_q^\perp(\mathbf{A})$ *where* $\mathbf{A}$ *is the top* $m \times n$ *submatrix of* $\mathbf{B}$, *and outputs a basis* $\mathbf{T_B}$ *of* $\Lambda_q^\perp(\mathbf{B})$ *with* $\|\widetilde{\mathbf{T_B}}\| \leq \|\widetilde{\mathbf{T_A}}\|$.

This kind of operation is also called *basis delegation*.

3.2 Zero-knowledge Proofs

Another important notion to have in mind is the *Zero-Knowledge proof* [27] (**ZK** proof) which is an important cryptographic primitive that is used to provide a way to ensure the validity of a statement without revealing any more information.

The basic idea is that there is two interactive algorithms: the prover P (potentially unbounded) and the verifier V (with polynomially bounded computational power). The prover aims to convince the verifier that he knows an element x from a language $\mathscr{L}$ that defines the set of true statements. To ease the task of the verifier, it is common to use a witness w for the element x, and prove that $(x, w) \in \mathscr{R}$ for a certain relation $\mathscr{R}$. We will work on this framework.

This primitive can be used for instance to solve the *identification problem*, which is the problem to prove your identity without letting someone else proving that he is you, even if they see the transcript of the proof. For instance our overly used login/password methods are not solving this problem, because the entity knows your password [29].

We will rely on *non-interactive zero-knowledge proofs* (**NIZK**) in our scheme. To do that we will use the Fiat-Shamir heuristic [25], which relies on Σ-protocols and aims to adapt it into a NIZK.

Definition 6 (Σ-Protocols [22, Def. 1]). A couple of interactive algorithm (P, V) where V is a PPT algorithm is called a Σ-*protocols* for the relation $\mathscr{R}$ if and only if:

- The interaction follows a 3-round move :

  1. P sends a message m, sometimes called a commitment
  2. V sends a *random* challenge c
  3. P sends back an answer s to the verifier, who decides to accept or reject the proof (respectively return 1 or 0)

  Hence a *transcript* is of the form (m, c, s).

- **Completeness:** if P, V follows the protocols on a valid input $(x, w) \in \mathscr{R}$, then the verifier always accepts.

- **Special soundness:** from any pair of accepting transcripts (m,c,s) and (m,c',s') where c and c' are two different challenges, one can efficiently compute w such that $(x,w) \in \mathscr{R}$.
- **Special honest-verifier zero-knowledge:** there exists a PPT simulator M which on input x and a random challenge c outputs an accepting transcript of the form (m,c,s) with the same probability distribution as real conversations between the honest prover and verifier on input x.

Remark 2. We can then work on Σ-protocols instead of zero-knowledge proofs as they are simpler objects to manipulate. For instance we can build an OR-proof, which means that from k Σ-protocols proving a statement, we can build a proof for the disjunction of those statements [21, 22]. We will use this property later on.

We can moreover notice that as we will use a variant of Stern's protocol [40] described in appendix D.1, where we need three transcripts to extract the secret. Then our soundness assumption will not be the described *special soundness*, but a weaker version of it.

3.3 One-time Signatures

For security purpose we will also need another crytographic primitive: *one-time signature*.

Definition 7 (One-time signature)**.** A *one-time signature scheme* consists of a triple of algorithms $\Pi^{\text{ots}} = (\mathscr{G}, \mathscr{S}, \mathscr{V})$ such that, on input of a security parameter 1^n, $\mathscr{G}$ generates a one-time key pair $(\mathsf{SK}, \mathsf{VK})$; $\mathscr{S}$ is a possibly randomized algorithm that outputs a signature $sig \leftarrow \mathscr{S}(\mathsf{SK}, M)$ on input of SK and M; and $\mathscr{V}(\mathsf{VK}, sig, M)$ is a deterministic algorithm that outputs 1 or 0. The standard correctness requirement mandates that $\mathscr{V}$ always accepts the signatures generated by $\mathscr{S}$.

In a strongly unforgeable one-time signature, the adversary is not only unable to forge a signature on a new message but, in addition, no PPT adversary can create a new signature for a previously signed message.

Definition 8 (Strong unforgeability)**.** $\Pi^{\text{ots}} = (\mathscr{G}, \mathscr{S}, \mathscr{V})$ is a strongly unforgeable one-time signature if the probability

$$\begin{aligned}\mathbf{Adv}^{\mathsf{OTS}}(n) = \Pr\big[&(\mathsf{SK},\mathsf{VK}) \leftarrow \mathscr{G}(1^n);\ (M, St) \leftarrow \mathscr{F}(\mathsf{VK});\ sig \leftarrow \mathscr{S}(\mathsf{SK}, M);\\ &(M', sig') \leftarrow \mathscr{F}(\mathsf{VK}, M, sig, St) : \mathscr{V}(\mathsf{VK}', sig', M') = 1 \wedge (M', sig') \neq (M, sig)\ \big],\end{aligned}$$

is negligible for any PPT forger $\mathscr{F}$, where St denotes $\mathscr{F}$'s state information across stages.

3.4 Gentry-Peikert-Vaikuntanathan's IBE scheme

One other scheme we will rely on will be the GPV's Identity Based Encryption scheme [26].

Definition 9 (Identity Based Encryption)**.** An Identity Based Encryption (**IBE**) scheme is a tuple of algorithm $(\mathsf{Setup}, \mathsf{Extract}, \mathsf{Enc}, \mathsf{Dec})$ such that: on input of a security parameter 1^n, Setup outputs a master key pair $(\mathsf{mpk}, \mathsf{msk})$; on input $\mathsf{mpk}, \mathsf{msk}$ and an identity id, $\mathsf{Extract}$ outputs sk_{id} which is the secret key of the identity id. Then Enc from the master public key mpk, a message M and an identity id outputs a ciphertext C; and Dec, from the master public key mpk, the private key sk'_{id} and a ciphertext C outputs a message M or a decryption error symbol $\perp$.

The goal of an IBE scheme is to simplify the key management. To cipher a message to someone, you just need his/her identity, which can be for instance his/her e-mail address.

Definition 10 (GPV's IBE)**.** The GPV's IBE uses a hash function $H : \{0,1\}^* \to \mathbb{Z}_q^n$ modeled as a random oracle. In the following χ denotes the LWE distribution. It is described as follows:

$\mathsf{Setup}(1^n)$: Using $\mathsf{TrapGen}(1^n, 1^m, q)$ we generates a random matrix $\mathbf{A}$ along with its trapdoor $\mathbf{T_A}$. Then the master public key $\mathsf{mpk} := \mathbf{A}$ and the master secret key $\mathsf{msk} := \mathbf{T_A}$

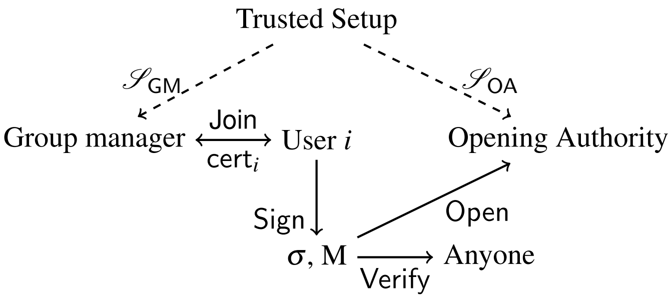

Figure 2: Relations between the protagonists in a dynamic group signature scheme

Extract$(\mathbf{A}, \mathbf{T}_\mathbf{A}, \mathrm{id})$**:** If the identity has already been queried, return the same sk_{id}. Otherwise, let $\mathbf{g} = H(\mathrm{id})$ and choose a decryption key sk_{id} using Lemma 3 to have a preimage of $\mathbf{g}$ with respect to the matrix $\mathbf{A}$. More formally we return $\mathbf{e} = \mathsf{SamplePre}(\mathbf{A}, \mathbf{T}, \mathbf{g}, \sigma)$, and we store the couple $(\mathrm{id}, \mathbf{e})$.

Enc$(\mathbf{A}, \mathrm{id}, b)$**:** To encrypt a bit $b \in \{0, 1\}$ to identity id, let $\mathbf{g} = H(\mathrm{id})$. We sample $\mathbf{s}$ from $\mathscr{U}\left(\mathbb{Z}_q^n\right)$ and we set $\mathbf{p} = \mathbf{A}^T\mathbf{s} + \mathbf{x} \in \mathbb{Z}_q^m$, where $\mathbf{x} \hookleftarrow \chi^m$. Output the ciphertext $(\mathbf{p}, c = \mathbf{g}^T\mathbf{s} + x + b \cdot \lfloor q/2 \rfloor) \in \mathbb{Z}_q^m \times \mathbb{Z}_q$, where $x \hookleftarrow \chi$.

Dec$(\mathbf{e}, (\mathbf{p}, c))$**:** computes $b' = c - \mathbf{e}^T\mathbf{p} \in \mathbb{Z}_q$. Output 0 if b' is closer to 0 than to $\lfloor q/2 \rfloor$ modulo q. Otherwise output 1.

The *multi-bit version* we will use just encrypts $k = \mathsf{Poly}\,(n)$ bits by creating k independent syndromes $\mathbf{g}_i$ in the Enc part. The idea it to use a different hash function mapping id to multiple uniform syndromes in $\mathbb{Z}_q^n$, one for each bit of the message.

Explicitly, H then maps to $\mathbb{Z}_q^{k \times n}$ which are seen as k uniform independent syndromes. Extract thus returns a short matrix $\mathbf{E} \in \mathbb{Z}^{m \times k}$ consisting of row-wise preimages for $\mathbf{G} \in \mathbb{Z}_q^{k \times n}$ with respect to $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$, and the ciphertext is then composed of two vectors. The first one remains the same, but the second one is then $\mathbf{G}\mathbf{s} + \mathbf{x}_2 + \mathbf{b}\lfloor q/2 \rfloor$ where $\mathbf{G} = H(\mathsf{id})$, $\mathbf{x}_2 \hookleftarrow \chi^k$ and $\mathbf{b}$ is the multi-bit message. That corresponds to a bit-wise ciphering using the GPV's IBE. Hence the deciphering is done by computing $\mathbf{b}' = \mathbf{c} - \mathbf{G}^T\mathbf{p}$ and the components of the resulting decryption are 0 if the corresponding component in $\mathbf{b}'$ is closer to 0 than to $\lfloor q/2 \rfloor$, and 1 otherwise.

Another possible extension is to allow non-binary messages. To do this we just change the step $\lfloor q/2 \rfloor$ into $\lfloor q/p \rfloor$.

3.5 Dynamic Group Signature

In this section we will present the definition of group signatures, and the security requirements.

Informally, a *group signature* is a scheme that allows a group member to attest that a message was provided by a member of a *group* without being altered during the process and preserving the *anonymity* of the users. This primitive was introduced by Bellare, Micciancio, and Warinschi [7] in 2003 and was extended to dynamic groups by Bellare, Shi and Zhang (**BSZ**) in 2005 [8].

In the setting of *dynamic groups*, the syntax of group signatures includes an interactive protocol which allows users to register as new members of the group at any time. The syntax and the security model are those defined by Kiayias and Yung [30]. Like the very similar **BSZ** model [8], the Kiayias-Yung (**KY**) model assumes an interactive *join* protocol whereby a prospective user becomes a group member by interacting with the group manager. This protocol provides the user with a membership certificate, cert_i, and a membership secret, sec_i.

We denote by $N \in \mathsf{Poly}(n)$ the maximal number of group members.

Definition 11 (Dynamic Group Signature)**.** A *dynamic group signature* scheme consists of the following algorithms or protocols.

Setup$(1^n, N)$**:** given a security parameter n and a maximal number of group members $N \in \mathbb{N}$, this algorithm is run by a trusted party to generate a group public key $\mathscr{Y}$, the group manager's private key $\mathscr{S}_{\mathsf{GM}}$ and the opening authority's private key $\mathscr{S}_{\mathsf{OA}}$. Each key is given to the appropriate authority while $\mathscr{Y}$ is made public. The algorithm also initializes a public state St comprising a set data structure $St_{\mathsf{users}} = \emptyset$ and a string data structure $St_{\mathsf{trans}} = \varepsilon$.
In the following, all algorithms have access to the public parameters $\mathscr{Y}$.

Join: is an *interactive* protocol between the group manager GM and a user $\mathscr{U}_i$ where the latter becomes a group member. The protocol involves two interactive Turing machines $\mathsf{J}_{\mathsf{user}}$ and J_{GM} that both take $\mathscr{Y}$ as input. The execution, denoted as $[\mathsf{J}_{\mathsf{user}}(n, \mathscr{Y}), \mathsf{J}_{\mathsf{GM}}(n, St, \mathscr{Y}, \mathscr{S}_{\mathsf{GM}})]$, ends with user $\mathscr{U}_i$ obtaining a membership secret sec_i, that no one else knows, and a membership certificate cert_i. If the protocol is successful, the group manager updates the public state St by setting $St_{\mathsf{users}} := St_{\mathsf{users}} \cup \{i\}$ as well as $St_{\mathsf{trans}} := St_{\mathsf{trans}} || \langle i, \mathsf{transcript}_i \rangle$.

Sign($\mathsf{cert}_i, \mathsf{sec}_i, M$**):** given a membership certificate cert_i, a membership secret sec_i and a message M, this *probabilistic* algorithm outputs a signature σ.

Verify(σ, M**):** given a signature σ, a message M and a group public key $\mathscr{Y}$, this *deterministic* algorithm returns either 0 or 1.

Open($\mathscr{S}_{\mathsf{OA}}, M, \sigma$**):** takes as input a message M, a valid signature σ w.r.t. $\mathscr{Y}$, the opening authority's private key $\mathscr{S}_{\mathsf{OA}}$ and the public state St. It outputs $i \in St_{\mathsf{users}} \cup \{\perp\}$, which is the identity of a group member or a symbol indicating an opening failure.

Each membership certificate contains a unique tag that identifies the user.

The correctness requirement basically captures that, if all parties *honestly* run the protocols, all algorithms are correct with respect to their specification described as above.

As mentioned in section 2, the Kiayias-Yung model [30] considers three security notions: the notion of security against *misidentification attacks* requires that, even if the adversary can introduce users under its control in the group, it cannot produce a signature that traces outside the set of dishonest users. The security against *framing attacks* implies that honest users can never be accused of having signed messages that they did not sign, even if the whole system conspired against them. The *anonymity* property is also formalized by granting the adversary access to a signature opening oracle as in the models of [8].

Correctness for Dynamic Group Signatures. Following the Kiayias-Yung terminology [30], we say that a public state St is *valid* if it can be reached from $St = (\emptyset, \varepsilon)$ by a Turing machine having oracle access to J_{GM}. Also, a state St' is said to *extend* another state St if it is within reach from St.

Moreover, as in [30], when we write $\mathsf{cert}_i \leftrightharpoons_{\mathscr{Y}} \mathsf{sec}_i$, it means that there exists coin tosses ϖ for J_{GM} and J_{user} such that, for some valid public state St', the execution of the interactive protocol $[\mathsf{J}_{\mathsf{user}}(n, \mathscr{Y}), \mathsf{J}_{\mathsf{GM}}(n, St', \mathscr{Y}, \mathscr{S}_{\mathsf{GM}})](\varpi)$ provides $\mathsf{J}_{\mathsf{user}}$ with $\langle i, \mathsf{sec}_i, \mathsf{cert}_i \rangle$.

Definition 12 (Correctness)**.** A dynamic group signature scheme is correct if the following conditions are all satisfied:

(1) In a valid state St, $|St_{users}| = |St_{trans}|$ always holds and two distinct entries of St_{trans} always contain certificates with distinct tag.

(2) If $[\mathsf{J}_{\mathsf{user}}(n, \mathscr{Y}), \mathsf{J}_{\mathsf{GM}}(n, St, \mathscr{Y}, \mathscr{S}_{\mathsf{GM}})]$ is run by two honest parties following the protocol and at the end $\langle i, \mathsf{cert}_i, \mathsf{sec}_i \rangle$ is obtained by $\mathsf{J}_{\mathsf{user}}$, then it holds that $\mathsf{cert}_i \leftrightharpoons_{\mathscr{Y}} \mathsf{sec}_i$.

(3) For each $\langle i, \mathsf{cert}_i, \mathsf{sec}_i \rangle$ such that $\mathsf{cert}_i \leftrightharpoons_{\mathscr{Y}} \mathsf{sec}_i$, satisfying condition 2, it always holds that:

$$\mathsf{Verify}\big(\mathsf{Sign}(\mathscr{Y}, \mathsf{cert}_i, \mathsf{sec}_i, M), M, \mathscr{Y}\big) = 1$$

(4) For any outcome $\langle i, \mathsf{cert}_i, \mathsf{sec}_i \rangle$ of the interaction $[\mathsf{J}_{\mathsf{user}}(.,.), \mathsf{J}_{\mathsf{GM}}(.,St,.,.)]$ for some valid state St, if $\sigma = \mathsf{Sign}(\mathscr{Y}, \mathsf{cert}_i, \mathsf{sec}_i, M)$, then

$$\mathsf{Open}(M, \sigma, \mathscr{S}_{\mathsf{OA}}, \mathscr{Y}, St') = i.$$

We formalize security properties via experiments where the adversary interacts with a stateful interface $\mathscr{I}$ that maintains the following variables:

- $\mathsf{state}_{\mathscr{I}}$: is a data structure representing the state of the interface as the adversary invokes the various oracles available in the attack games. It is initialized as $\mathsf{state}_{\mathscr{I}} = (St, \mathscr{Y}, \mathscr{S}_{\mathsf{GM}}, \mathscr{S}_{\mathsf{OA}}) \leftarrow \mathsf{Setup}(n, N)$. It includes the (initially empty) set St_{users} of group members and a dynamically growing database St_{trans} storing the transcripts of previously executed join protocols.
- $n = |St_{users}| < N$ denotes the current cardinality of the group.
- Sigs: is a database of signatures created by the signing oracle. Each entry consists of a triple (i, M, σ) indicating that message M was signed by user i.
- U^a: is the set of users that were introduced by the adversary in the system in an execution of the join protocol.
- U^b: is the set of honest users that the adversary, acting as a dishonest group manager, introduced in the system. For these users, the adversary obtains the transcript of the join protocol but not the user's membership secret.

When mounting attacks, adversaries will be granted access to the following oracles:

- Q_{pub}, Q_{keyGM} and Q_{keyOA}: when these oracles are invoked, the interface looks up $\mathsf{state}_{\mathscr{I}}$ and returns the group public key $\mathscr{Y}$, the GM's private key $\mathscr{S}_{\mathsf{GM}}$ and the opening authority's private key $\mathscr{S}_{\mathsf{OA}}$ respectively.
- $Q_{\mathsf{a\text{-}join}}$: allows the adversary to introduce users under his control in the group. On behalf of the GM, the interface runs J_{GM} in interaction with the $\mathsf{J}_{\mathsf{user}}$-executing adversary who plays the role of the prospective user in the join protocol. If this protocol successfully ends, the interface increments n, updates St by inserting the new user n in both sets St_{users} and U^a. It also sets $St_{\mathsf{trans}} := St_{\mathsf{trans}} || \langle n, \mathsf{transcript}_n \rangle$.
- $Q_{\mathsf{b\text{-}join}}$: allows the adversary, acting as a corrupted group manager, to introduce new honest group members of his/her choice. The interface triggers an execution of $[\mathsf{J}_{\mathsf{user}}, \mathsf{J}_{\mathsf{GM}}]$ and runs $\mathsf{J}_{\mathsf{user}}$ in interaction with the adversary who runs J_{GM}. If the protocol successfully completes, the interface increments n, adds user n to St_{users} and U^b and sets $St_{\mathsf{trans}} := St_{\mathsf{trans}} || \langle n, \mathsf{transcript}_n \rangle$. It stores the membership certificate cert_n and the membership secret sec_n in a *private* part of $\mathsf{state}_{\mathscr{I}}$.
- Q_{sig}: given a message M, an index i, the interface checks whether the private area of $\mathsf{state}_{\mathscr{I}}$ contains a certificate cert_i and a membership secret sec_i. If no such elements $(\mathsf{cert}_i, \mathsf{sec}_i)$ exist or if $i \notin U^b$, the interface returns $\perp$. Otherwise, it outputs a signature σ on behalf of user i and also sets $\mathsf{Sigs} \leftarrow \mathsf{Sigs} || (i, M, \sigma)$.
- Q_{open}: when this oracle is invoked on input of a valid pair (M, σ), the interface runs algorithm Open using the current state St. When S is a set of pairs of the form (M, σ), $Q_{\mathsf{open}}^{\neg S}$ denotes a restricted oracle that only applies the opening algorithm to pairs (M, σ) which are not in S.
- Q_{read} and Q_{write}: are used by the adversary to read and write the content of $\mathsf{state}_{\mathscr{I}}$. Namely, at each invocation, Q_{read} outputs the whole $\mathsf{state}_{\mathscr{I}}$ but the public/private keys and the private part of $\mathsf{state}_{\mathscr{I}}$ where membership secrets are stored after $Q_{\mathsf{b\text{-}join}}$-queries. By using Q_{write}, the adversary can modify $\mathsf{state}_{\mathscr{I}}$ at will as long as it does not remove or alter elements of St_{users}, St_{trans} or invalidate the public state St: for example, the adversary is allowed to create dummy users as long as he/she does not re-use already existing certificate tags.

Using this formalism, we can now properly define the three announced security properties.

Security Against Misidentification Attacks. In a misidentification attack, the adversary can corrupt the opening authority using the Q_{keyOA} oracle. Moreover, he/she can also introduce malicious users in the group via $Q_{\mathsf{a\text{-}join}}$-queries. His/her purpose is to come up with a valid signature $\sigma^\star$. He/she succeeds if the produced signature $\sigma^\star$ does not open to any adversarially-controlled.

Definition 13. A dynamic group signature scheme is secure against *misidentification attacks* if, for any PPT adversary $\mathscr{A}$ involved in the experiment hereunder, we have the advantage of $\mathscr{A}$ to be:

$$\mathbf{Adv}_{\mathscr{A}}^{\text{mis-id}}(n) = \Pr\left[\mathbf{Exp}_{\mathscr{A}}^{\text{mis-id}}(n) = 1\right] \in \mathsf{negl}\left(n\right)$$

Algorithm 1: Experiment $\mathbf{Exp}_{\mathscr{A}}^{\textit{mis-id}}(n)$

```
$\mathsf{state}_{\mathscr{I}} = (St, \mathscr{Y}, \mathscr{S}_{\mathsf{GM}}, \mathscr{S}_{\mathsf{OA}}) \leftarrow \mathsf{Setup}(n, N)$;
$(M^\star, \sigma^\star) \leftarrow \mathscr{A}(Q_{\mathsf{pub}}, Q_{\mathsf{a\text{-}join}}, Q_{\mathsf{read}}, Q_{\mathsf{keyOA}})$;
if $\mathsf{Verify}(\sigma^\star, M^\star, \mathscr{Y}) = 0$ then
    return 0;
$i = \mathsf{Open}(M^\star, \sigma^\star, \mathscr{S}_{\mathsf{OA}}, \mathscr{Y}, St')$;
if $i \notin U^a$ then
    return 1;
return 0;
```

Non-Frameability. Framing attacks consider the situation where the entire system, including the group manager and the opening authority, is colluding against some honest user. The adversary can corrupt the group manager as well as the opening authority (via oracles Q_{keyGM} and Q_{keyOA}, respectively). He/she is also allowed to introduce honest group members (via $Q_{\mathsf{b\text{-}join}}$-queries), observe the system while these users sign messages and create dummy users using Q_{write}. The adversary eventually aims at framing an honest group member.

Definition 14. A dynamic group signature scheme is secure against *framing attacks* if, for any PPT adversary $\mathscr{A}$ involved in the experiment below, it holds that:

$$\mathbf{Adv}_{\mathscr{A}}^{\text{fra}}(n) = \Pr\left[\mathbf{Exp}_{\mathscr{A}}^{\text{fra}}(n) = 1\right] \in \mathsf{negl}\left(n\right)$$

Algorithm 2: Experiment $\mathbf{Exp}_{\mathscr{A}}^{\text{fra}}(n)$

```
$\mathsf{state}_{\mathscr{I}} = (St, \mathscr{Y}, \mathscr{S}_{\mathsf{GM}}, \mathscr{S}_{\mathsf{OA}}) \leftarrow \mathsf{Setup}(n, N)$;
$(M^\star, \sigma^\star) \leftarrow \mathscr{A}(Q_{\mathsf{pub}}, Q_{\mathsf{keyGM}}, Q_{\mathsf{keyOA}}, Q_{\mathsf{b\text{-}join}}, Q_{\mathsf{sig}}, Q_{\mathsf{read}}, Q_{\mathsf{write}})$;
if $\mathsf{Verify}(\sigma^\star, M^\star, \mathscr{Y}) = 0$ then
    return 0;
if $i = \mathsf{Open}(M^\star, \sigma^\star, \mathscr{S}_{\mathsf{OA}}, \mathscr{Y}, St') \notin U^b$ then
    return 0;
if $\left(\bigwedge_{j \in U^b \text{ s.t. } j = i} (j, M^\star, *) \notin \mathsf{Sigs}\right)$ then
    return 1;
return 0;
```

Full Anonymity. The notion of anonymity is formalized by means of a game involving a two-stage adversary. The first stage is called play stage and allows the adversary $\mathscr{A}$ to modify $\mathsf{state}_{\mathscr{I}}$ via Q_{write}-queries and open arbitrary signatures by probing Q_{open}. When the play stage ends, $\mathscr{A}$ chooses a message $M^\star$ as well as two pairs $(\mathsf{sec}_0^\star, \mathsf{cert}_0^\star)$ and $(\mathsf{sec}_1^\star, \mathsf{cert}_1^\star)$, consisting of a valid membership certificate and a corresponding membership secret. Then, the challenger flips a coin $d \leftarrow \{0,1\}$ and computes a challenge signature $\sigma^\star$ using $(\mathsf{sec}_d^\star, \mathsf{cert}_d^\star)$. The adversary is given $\sigma^\star$ with the task of eventually guessing the bit $d \in \{0,1\}$. Before doing so, he/she is allowed further oracle queries throughout the second stage, called guess stage, but is restricted not to query Q_{open} for $(M^\star, \sigma^\star)$.

Definition 15. A dynamic group signature scheme is fully anonymous if, for any PPT adversary $\mathscr{A}$,

$$\mathbf{Adv}_{\mathscr{A}}^{\mathrm{anon}}(n) := |\Pr[\mathbf{Exp}_{\mathscr{A}}^{\mathrm{anon}}(n) = 1] - 1/2| \in \mathsf{negl}(n)$$

Algorithm 3: Experiment $\mathbf{Exp}_{\mathscr{A}}^{\mathrm{anon}}(n)$

```
$\mathsf{state}_{\mathscr{I}} = (St, \mathscr{Y}, \mathscr{S}_{\mathsf{GM}}, \mathscr{S}_{\mathsf{OA}}) \leftarrow \mathsf{Setup}(n)$;
$\big(aux, M^\star, (\mathsf{sec}_0^\star, \mathsf{cert}_0^\star), (\mathsf{sec}_1^\star, \mathsf{cert}_1^\star)\big) \leftarrow \mathscr{A}(\mathsf{play}; Q_{\mathsf{pub}}, Q_{\mathsf{keyGM}}, Q_{\mathsf{open}}, Q_{\mathsf{read}}, Q_{\mathsf{write}})$;
if $\neg(\mathsf{cert}_b^\star \leftrightharpoons_{\mathscr{Y}} \mathsf{sec}_b^\star)$ for $b \in \{0,1\}$ then
    return 0;
if $\mathsf{cert}_0^\star = \mathsf{cert}_1^\star$ then
    return 0;
Picks random $d \leftarrow \{0,1\}$; $\sigma^\star \leftarrow \mathsf{Sign}(\mathscr{Y}, \mathsf{cert}_d^\star, \mathsf{sec}_d^\star, M^\star)$;
$d' \leftarrow \mathscr{A}(\mathsf{guess}; \sigma^\star, aux, Q_{\mathsf{pub}}, Q_{\mathsf{keyGM}}, Q_{\mathsf{open}}^{\neg\{(M^\star,\sigma^\star)\}}, Q_{\mathsf{read}}, Q_{\mathsf{write}})$;
if $d' = d$ then
    return 1;
return 0;
```

One can wonder why the *revocation* is not in the *dynamic group signature scheme* description, the reason is only pragmatic. It is a different problem to build a scheme that allows to revoke a group member than a scheme that allows inserting a group member [23], and it has to be done in a case-by-case fashion.

4 Our Protocol

In this part we will present the scheme that was designed during this internship. As explained in section 1.1 it is an adaptation of the static scheme in [31]. We will first start by a description of the ideas of the construction, along with the encountered difficulties, then we will end with a formal definition of our protocol.

4.1 Main Ideas

As explained in section 1.1, we build our protocol from a static group signature. The idea is to change the key attribution algorithms that gives $\mathsf{gpk}[i]$ to the user $\mathscr{U}_i$ to make it dynamic through an interactive Join protocol.

Another main difference is that the *Opening Manager* OA and the *Group Manager* GM – namely the certificate deliverer – are not the same authority, which lead us to separate the secrets that the key manager and the opening manager have, in such a way that the following hold:

- **Anonymity:** No one except the opening manager can open a signature

- **Non-frameability:** Even if OA and GM collude against you, they cannot build a signature accusing you.

Our first idea was to adapt the Laguillaumie *et al.* [31] protocol, which is recalled in Appendix B. In the latter, each group member receives a unique ℓ-bit identifier $\mathrm{id} \in \{0,1\}^\ell$, with $\ell = \log N$ and where N is the maximal number of group members. The membership certificate is a small-norm integer matrix $\mathbf{T}_{\mathrm{id}} \in \mathbb{Z}^{2m \times 2m}$ such that $\mathbf{T}_{\mathrm{id}} \cdot \mathbf{A}_{\mathrm{id}} = \mathbf{0}^{m \times n}$, where

$$\mathbf{A}_{\mathrm{id}} = \left[\begin{array}{c} \mathbf{A} \\ \hline \mathbf{A}_0 + \sum_{i=1}^{\ell} \mathrm{id}[i]\mathbf{A}_i \end{array}\right] \in \mathbb{Z}_q^{2m \times n}$$

is a matrix that encodes the group member's identity id, similarly as in Boyen's signature [13]. In order to sign a message M, the group member uses the trapdoor $\mathbf{T}_{\mathrm{id}}$ to generate a short integer vector $\mathbf{v} \in \mathbb{Z}^{2m}$ such that $\mathbf{v}^T \cdot \mathbf{A}_{\mathrm{id}} = \mathbf{0}^{1 \times n}$ and generates a proof of knowledge of a pair $(\mathbf{v}, \mathrm{id})$ such that $\mathbf{v}^T \cdot \mathbf{A}_{\mathrm{id}} = \mathbf{0}^{1 \times n}$. To achieve *non-frameability*, our first attempt was to add to the group public key a random matrix $\mathbf{D} \in \mathbb{Z}_p^{m \times n}$ and, when a user joins the system, let that user choose a short integer vector $\mathbf{z}_i \in \mathbb{Z}^m$ for which he sends $\mathbf{v}_i^T = \mathbf{z}_i^T \cdot \mathbf{D} \in \mathbb{Z}_q^{1 \times n}$ to the group manager. Note that computing such a short $\mathbf{z}_i \in \mathbb{Z}^m$ from $\mathbf{v}_i \in \mathbb{Z}_q^n$ is a hard problem, called ISIS, as mentioned in section 3.1.2 (so, the group manager cannot recover the user's membership secret $\mathsf{sec}_i = \mathbf{z}_i$ from what he sees in the joining protocol). Then, the group manager could use some trapdoor information to return to the user a short integer vector $\mathsf{cert}_i = \mathbf{d}_i \in \mathbb{Z}^{2m}$ such that

$$\mathbf{d}_i^T \cdot \mathbf{A}_{\mathrm{id}} = \mathbf{z}_i^T \cdot \mathbf{D}, \tag{1}$$

which certifies the user's membership secret $\mathsf{sec}_i = \mathbf{z}_i$ and the corresponding public value $\mathbf{v}_i^T = \mathbf{z}_i^T \cdot \mathbf{D}$. Our hope was that the user would have been able to sign messages by proving his knowledge of integer vectors $(\mathbf{d}_i, \mathbf{z}_i) \in \mathbb{Z}^{2m} \times \mathbb{Z}^m$ such that (1) is true and without revealing his identity id. The problem is that the Laguillaumie *et al.* protocol Laguillaumie *et al.* [31] requires each user to have a membership certificate consisting of a full matrix $\mathbf{T}_{\mathrm{id}}$ such that $\mathbf{T}_{\mathrm{id}} \cdot \mathbf{A}_{\mathrm{id}} = \mathbf{0}^{m \times n}$, and not just a vector $\mathbf{d}_i \in \mathbb{Z}^{2m}$ such that $\mathbf{d}_i^T \cdot \mathbf{A}_{\mathrm{id}} = \mathbf{0}^{1 \times n}$. This means that, for anonymity reasons, at each new signature, the group member would have to use $\mathbf{T}_{\mathrm{id}}$ to generate a fresh short integer vector $\mathbf{d}_i$ satisfying (1) (for the same vector $\mathbf{z}_i \in \mathbb{Z}^m$ which is the user's membership secret). Unfortunately, we do not know how to do this for a non-zero $\mathbf{z}_i$ (in [31], it was possible since there was no membership secret and we had $\mathbf{z}_i = \mathbf{0}^m$). A tempting solution is to give to the user a full basis $\mathbf{T} \in \mathbb{Z}^{3m \times 3m}$ for the lattice $\Lambda^\perp(\mathbf{A}_{\mathrm{id}})$ associated with the matrix

$$\mathbf{A}_{\mathrm{id}} = \left[\begin{array}{c} A \\ \hline \mathbf{A}_0 + \sum_{i=1}^{\ell} \mathrm{id}[i]\mathbf{A}_i \\ \hline \mathbf{D} \end{array}\right] \in \mathbb{Z}_q^{3m \times n},$$

which would allow to user to sample short vectors $(\mathbf{d}_i, \mathbf{z}_i)$ satisfying (1). Unfortunately, it does not provide a way to attach the user's membership certificate $\mathbf{d}_i$ to a public syndrome $\mathbf{v}_i^T = \mathbf{z}_i^T \cdot \mathbf{D}$ certified by the group manager.

The problem is to bind the user to a unique public value $\mathbf{v}^T = \mathbf{z}^T \cdot \mathbf{D}$, which the user signs using his long-term public key registered in some Public Key Infrastructure and which it cannot deny later on. What we need is a way to: (i) Prevent a cheating user from generating randomized pairs $(\mathsf{cert}_i', \mathsf{sec}_i') = (\mathbf{d}_i', \mathbf{z}_i')$ that satisfies the same verification equations but for which the syndrome $\mathbf{z}_i'^T \cdot \mathbf{D}$ cannot be linked to the user by means of a regular digital signature generated by the user when he joined the group; (ii) Prevent a dishonest group manager from creating group signatures for which the opening operation reveals the public value $\mathbf{v}^T = \mathbf{z}^T \cdot \mathbf{D}$ of the user.

To solve this problem, we modify the construction of [31] in order to have membership certificates consisting of a single integer vector $\mathbf{d} \in \mathbb{Z}^{2m}$ satisfying a relation like (1). For reasons inherent to the security proof, we actually need to add a non-homogeneous term $\mathbf{u} \in \mathbb{Z}_q^n$ and let the user's secret pair $(\mathsf{cert}, \mathsf{sec})$ consist of vectors $(\mathbf{d}_i, \mathbf{z}_i) \in \mathbb{Z}^{2m} \times \mathbb{Z}^m$ satisfying

$$\mathbf{d}_i^T \cdot \mathbf{A}_{\mathrm{id}} = \mathbf{z}_i^T \cdot \mathbf{D} + \mathbf{u}^T. \tag{2}$$

Each group signature is generated by proving the knowledge of a solution to the inhomogeneous short integer solution problem (ISIS) which is basically a proof of knowledge of a valid pair $(\mathbf{d}_i, \mathbf{z}_i) \in \mathbb{Z}^{2m} \times \mathbb{Z}^m$ satisfying (2), with the additional property that the proof hides $\mathrm{id}_i \in \{0,1\}^\ell$. In order to allow the opening authority to open signatures, the signer additionally generates GPV encryptions $(\mathbf{c}_{\mathrm{id}}, \mathbf{c}_{\mathbf{d},1}, \mathbf{c}_{\mathbf{d},2})$ of the complete membership certificate $(\mathbf{d}_i, \mathrm{id}_i)$ (exactly as in most constructions of dynamic group signatures) and proves the knowledge of $(\mathbf{d}_i, \mathbf{z}_i, \mathrm{id}_i) \in \mathbb{Z}^{2m} \times \mathbb{Z}^m \times \{0,1\}^\ell$ that satisfy (2) and are consistent with the values encrypted in $(\mathbf{c}_{\mathrm{id}}, \mathbf{c}_{\mathbf{d},1}, \mathbf{c}_{\mathbf{d},2})$. The reason why we use the GPV identity-based encryption scheme for this reason is that, as in [35], it allows us to prove anonymity in the strong sense (when the adversary has an opening oracle) by applying the Canetti-Halevi-Katz methodology [17].

To sum up, the signature is produced as follows: the signer uses a *one-time signature scheme* to produce a pair $(\mathsf{VK}, \mathsf{SK})$. The signer then encrypts – using the GPV master public key and the one-time verification key as the identity – his membership certificate, and sends it along with NIZK proofs that everything was done correctly. In the challenge transcript of the proofs, the message intervene in order to make the signature dependent of the message. Finally he signs everything using the one-time signature secret key SK and gives the signature along with the verification key and the signed message, namely the ciphertexts of the identity and the certificate and the proofs that everything went correctly.

The proofs and the signature allows the verifier to check that the signature is valid.

The opening authority then has the knowledge of the master secret key for the GPV IBE. That allows him, using the identity VK to decipher the identity and the certificate of the signature Σ. Using this piece of information to verify that the user is indeed in the transcript.

4.2 Description

The parameters are set in such a way that all algorithms described in section 3 can be implemented in polynomial time and are correct, and so that the security properties hold, in the random oracle model (**ROM**) and under LWE and SIS hardness assumptions with known reductions from standard worst-case lattice problems with polynomial approximation factors. In order to set the parameters, we need to take into account the "smudging" technique [3, Lemma 2.1], which is used in our security proof, requires a modulus q that is exponential in the security parameter $\lambda \in \mathbb{N}$. More precisely, we need the following choice of parameters:

- Parameter q is prime and $\mathscr{O}(2^\lambda)$.
- Parameter n is in $\mathscr{O}(\lambda^2)$.
- Parameter m is $\geq 2n \log q$. For example, $m = \mathscr{O}(\lambda^3)$.
- The standard deviation σ of the main discrete gaussian is $\omega(\log m)$.
- The SIS parameter β should satisfy $\beta \leq q/\omega(\sqrt{n \log n})$. For example, we can take $\beta = 2^{\lambda/2}$.
- The parameter p is smaller than $\sigma\sqrt{m} \approx 2^{\lambda/2} \cdot \lambda^{3/2}$.
- The LWE noise distribution χ of the GPV scheme should have a gaussian parameter α such that $\alpha q > 2\sqrt{n}$ (this is necessary for Regev's reduction).
- The correctness of the opening algorithm (which relies on the GPV decryption algorithm) requires that, if $\mathbf{e} \in \mathbb{Z}^m$ denotes a GPV private key and $\mathbf{x}$ is the LWE noise vector, the error term $|\mathbf{e}_1^T \mathbf{x}| \leq \|\mathbf{e}\| \cdot q\alpha\omega(\sqrt{\log m}) + \|\mathbf{e}\| \cdot \sqrt{m}/2$ should be smaller than $q/(2p+1)$.

If we choose $\alpha q = \mathscr{O}(\lambda)$ and if we choose p and q such that $q/(2p+1) \approx 2^{\lambda/2}/\lambda^{3/2}$, the correctness condition becomes

$$\|\mathbf{e}\|\lambda^{5/2} < 2^{\lambda/2}/\lambda^{3/2},$$

which is satisfied for any reasonable choice of the gaussian parameter σ_{GPV} in the GPV system since we usually have $\|\widetilde{\mathbf{T}_\mathbf{B}}\| \leq \mathscr{O}(\sqrt{n \log q})$.

The reason for the choice of q is that we will need two additional gaussian parameters $\sigma_0, \sigma_1 > 0$ such that $\sigma^2 = \sigma_0^2 + \sigma_1^2$. These parameters should be chosen in such a way that the joint distribution $(\mathbf{z}_0, \mathbf{z}_1) \in D_{\mathbb{Z}^m,\sigma_1} \times D_{\mathbb{Z}^m,\sigma_2}$ obtained by sampling $\mathbf{z}_0 \hookleftarrow D_{\mathbb{Z}^m,\sigma_0}$, $\mathbf{z}_1 \hookleftarrow D_{\mathbb{Z}^m,\sigma_1}$ and computing $\mathbf{z} = \mathbf{z}_0 + \mathbf{z}_1$ is statistically indistinguishable from the distribution obtained by sampling $\mathbf{z} \hookleftarrow D_{\mathbb{Z}^m,\sigma}$, $\mathbf{z} \hookleftarrow D_{\mathbb{Z}^m,\sigma_0}$ and computing $\mathbf{z}_1 = \mathbf{z} - \mathbf{z}_0$. One way to do this is to use the smudging technique [3, Lemma 2.1], which is to choose σ_1 exponentially larger than σ_0.

The scheme relies on the GPV's IBE as defined in section 3.4. The idea is that we will already have *chosen-ciphertext* anonymity [17] as the underlying encryption scheme will be CCA-resistant, according to Canetti, Halevi, and Katz [17].

In the following, we denote by χ the distribution of the LWE noise in the GPV encryption scheme. We assume that this distribution is B-bounded (i.e., for any $x \hookleftarrow \chi$, $|x| \leq B$ with overwhelming probability) for some integer $B > 0$. If $D^m_{\mathbb{Z},\sigma}$ denotes the discrete gaussian distribution with standard deviation σ over $\mathbb{Z}^m$, this distribution is β-bounded where $\beta = \sigma\sqrt{m}$.

Setup$(1^\lambda, 1^N)$**:** Given a security parameter $\lambda > 0$ and the maximal number of group members $N = 2^\ell \in \mathsf{poly}(\lambda)$, choose parameters n, q, m, p, α and σ as specified above. Choose a hash function $H : \{0,1\}^* \to \{1,2,3\}^t$ for some $t = \Theta(n)$, which will be modeled as a random oracle in the security analysis. Then, do the following.

1. Choose a uniformly random matrix $\mathbf{D} \hookleftarrow \mathbb{Z}_q^{m \times n}$. Then, run $\mathsf{TrapGen}(1^n, 1^m, q)$ to get $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ and a short basis $\mathbf{T_A}$ of $\Lambda_q^\perp(\mathbf{A})$. This basis allows computing short vectors in $\Lambda^\perp(\mathbf{A})$ with a gaussian parameter $\sigma \geq \|\widetilde{\mathbf{T_A}}\| \cdot \omega(\sqrt{\log m})$. Next, chooses $\ell + 1$ random matrices $\mathbf{A}_0, \mathbf{A}_1, \ldots, \mathbf{A}_\ell \hookleftarrow \mathbb{Z}_q^{m \times n}$.
2. Generate a master key pair for the Gentry-Peikert-Vaikuntanathan IBE scheme in its multi-bit variant. This key pair consists of a public random matrix $\mathbf{B} \in_R \mathbb{Z}_q^{m \times n}$ and a short basis $\mathbf{T_B} \in \mathbb{Z}^{m \times m}$ of $\Lambda^\perp(\mathbf{B})$. This basis will allow us to compute GPV private keys with a gaussian parameter $\sigma_{\mathsf{GPV}} \geq \|\widetilde{T_\mathbf{B}}\| \cdot \sqrt{\log m}$.
3. Choose a uniformly random matrix $\mathbf{D} \hookleftarrow \mathbb{Z}_q^{m \times n}$ and a vector $\mathbf{u} \hookleftarrow \mathbb{Z}_q^n$.
4. Choose a one-time signature scheme $\Pi^{\mathrm{OTS}} = (\mathscr{G}, \mathscr{S}, \mathscr{V})$ and hash functions $H_0 : \{0,1\}^* \to \mathbb{Z}_q^{\ell \times n}$, $H_1 : \{0,1\}^* \to \mathbb{Z}_q^{m \times n}$ that will be modeled as a random oracle in the security analysis.

The group public key is defined as

$$\mathscr{Y} := \big(\mathbf{A},\ \{\mathbf{A}_j\}_{j=0}^{\ell},\ \mathbf{B},\ \mathbf{D},\ \mathbf{u},\ \Pi^{\mathrm{OTS}},\ H,\ H_0,\ H_1\big).$$

The opening authority's private key is $\mathscr{S}_{\mathsf{OA}} := \mathbf{T_B}$ and the private key of the group manager consists of $\mathscr{S}_{\mathsf{GM}} := \mathbf{T_A}$. The algorithm outputs $\big(\mathscr{Y}, \mathscr{S}_{\mathsf{GM}}, \mathscr{S}_{\mathsf{OA}}\big)$.

Join$^{(\mathsf{GM},\mathscr{U}_i)}$**:** the group manager GM and the prospective user $\mathscr{U}_i$ run the following interactive protocol:

1. $\mathscr{U}_i$ samples a discrete gaussian vector $\mathbf{z}_{i,0} \leftarrow D^m_{\mathbb{Z},\sigma_0}$ and compute $\mathbf{v}_{i,0}^T = \mathbf{z}_{i,0}^T \cdot \mathbf{D} \in \mathbb{Z}_q^{1 \times n}$. He sends the vector $\mathbf{v}_{i,0} \in \mathbb{Z}_q^n$ to GM and runs an interactive protocol with GM in order to provide a zero-knowledge proof of knowledge of $\mathbf{z}_{i,0}$ such that $\mathbf{v}_{i,0}^T = \mathbf{z}_{i,0}^T \cdot \mathbf{D}$. Then, GM chooses a fresh ℓ-bit identifier $\mathrm{id}_i = \mathrm{id}_i[1] \ldots \mathrm{id}_i[\ell] \in \{0,1\}^\ell$ samples a short vector $\mathbf{z}_{i,1} \leftarrow D^m_{\mathbb{Z},\sigma_1}$ and sends $(\mathrm{id}_i, \mathbf{z}_{i,1})$ to $\mathscr{U}_i$. The user $\mathscr{U}_i$ computes $\mathbf{z}_i = \mathbf{z}_{i,0} + \mathbf{z}_{i,1} \in \mathbb{Z}^m$ and $\mathbf{v}_i^T = \mathbf{z}_i \cdot \mathbf{D} \bmod q$. Then, he returns $\mathbf{v}_i \in \mathbb{Z}_q^m$ along with a signature $sig_i = \mathrm{Sign}_{\mathsf{usk}[i]}(\mathbf{v}_i, \mathrm{id}_i, \mathbf{z}_{i,1})$ on the triple $(\mathbf{v}_i, \mathrm{id}_i, \mathbf{z}_{i,1})$ to the GM.

2. J_{GM} verifies that sig_i is a valid signature on $(\mathbf{v}_i, \mathrm{id}_i, \mathbf{z}_{i,1})$ w.r.t. $\mathsf{upk}[i]$ and aborts if this is not the case. Otherwise, it uses $\mathscr{S}_{\mathsf{GM}} = \mathbf{T_A}$ to certify $\mathscr{U}_i$ as a new group member. To this end, GM defines the matrix

$$\mathbf{A}_{\mathrm{id}_i} = \left[\begin{array}{c} \mathbf{A} \\ \hline \mathbf{A}_0 + \sum_{j=1}^{\ell} \mathrm{id}_i[j]\mathbf{A}_j \end{array}\right] \in \mathbb{Z}_q^{2m \times n}. \tag{3}$$

Then, GM runs $\mathbf{T}'_{\mathrm{id}_i} \leftarrow \mathsf{ExtBasis}(\mathbf{A}_{\mathrm{id}_i}, \mathbf{T_A})$ to obtain a short delegated basis $\mathbf{T}'_{\mathrm{id}_i}$ of $\Lambda_q^{\perp}(\mathbf{A}_{\mathrm{id}_i}) \in \mathbb{Z}^{2m \times 2m}$. Finally, GM uses the obtained delegated basis $\mathbf{T}'_{\mathrm{id}_i}$ to compute a short vector $\mathbf{d}_i = \left[\begin{array}{c} \mathbf{d}_{i,1} \\ \hline \mathbf{d}_{i,2} \end{array}\right] \in \mathbb{Z}^{2m}$ such that

$$\mathbf{d}_i^T \cdot \mathbf{A}_{\mathrm{id}_i} \quad = \quad \mathbf{d}_i^T \cdot \left[\begin{array}{c} \mathbf{A} \\ \hline \mathbf{A}_0 + \sum_{j=1}^{\ell} \mathrm{id}_i[j]\mathbf{A}_j \end{array}\right] = \mathbf{v}_i^T + \mathbf{u}^T \bmod q \tag{4}$$

User $\mathscr{U}_i$'s membership certificate consists of the pair $\mathsf{cert}_i := (\mathrm{id}_i, \mathbf{d}_i)$ and his membership secret is the integer vector $\mathsf{sec}_i = \mathbf{z}_i \in \mathbb{Z}^m$.

3. J_{GM} stores $\mathsf{transcript}_i = (\mathbf{v}_i, \mathrm{id}_i, i, \mathsf{upk}[i], sig_i)$ in the database St_{trans}, sends cert_i to $\mathsf{J}_{\mathsf{user}}$. $\mathsf{J}_{\mathsf{user}}$ halts if $\mathbf{d}_i$ is not a short vector satisfying (4). Otherwise, $\mathsf{J}_{\mathsf{user}}$ defines the membership certificate as $\mathsf{cert}_i = (\mathrm{id}_i, \mathbf{d}_i)$. The membership secret sec_i is defined to be $\mathsf{sec}_i = \mathbf{z}_i \in \mathbb{Z}^m$.

Sign$(\mathscr{Y}, \mathsf{cert}_i, \mathsf{sec}_i, M)$**:** To sign $M \in \{0,1\}^*$ using the membership certificate $\mathsf{cert}_i = (\mathrm{id}_i, \mathbf{d}_i)$, where $\mathbf{d}_i = \begin{bmatrix} \mathbf{d}_{i,1} \\ \mathbf{d}_{i,2} \end{bmatrix}$ and the membership secret $\mathsf{sec}_i = \mathbf{z}_i \in \mathbb{Z}^m$, the group member $\mathscr{U}_i$ generates a one-time signature key pair $(\mathsf{VK}, \mathsf{SK}) \leftarrow \mathscr{G}(n)$ and conducts the following steps.

1. Compute $\mathbf{G}_0 = H_0(\mathsf{VK}) \in \mathbb{Z}_q^{\ell \times n}$ and use it as an IBE public key to encrypt $\mathrm{id}_i \in \{0,1\}^{\ell}$. Namely, compute

$$(\mathbf{c}_1, \mathbf{c}_2) = \big(\mathbf{B} \cdot \mathbf{s}_0 + \mathbf{e}_1,\ \mathbf{G}_0 \cdot \mathbf{s}_0 + \mathbf{e}_2 + \mathrm{id}_i \cdot \lfloor q/2 \rfloor\big) \in \mathbb{Z}_q^m \times \mathbb{Z}_q^{\ell} \tag{5}$$

for randomly chosen $\mathbf{s}_0 \hookleftarrow \chi^n$, $\mathbf{e}_1 \hookleftarrow \chi^m$, $\mathbf{e}_2 \hookleftarrow \chi^{\ell}$.

2. Let $p > 0$ be an upper bound on entries of $\mathbf{d}_{i,1}$, $\mathbf{d}_{i,2}$ in absolute value. Compute $\mathbf{G}_1 = H_1(\mathsf{VK}) \in \mathbb{Z}_q^{m \times n}$ and use it as an IBE public key to encrypt $\mathbf{d}_{i,1} \in \mathbb{Z}^m$ and $\mathbf{d}_{i,2} \in \mathbb{Z}^m$. Namely, compute

$$(\mathbf{c}_3, \mathbf{c}_4) = \big(\mathbf{B} \cdot \mathbf{s}_1 + \mathbf{e}_3,\ \mathbf{G}_1 \cdot \mathbf{s}_1 + \mathbf{e}_4 + \mathbf{d}_{i,1} \cdot \lfloor q/p \rfloor\big) \in \mathbb{Z}_q^m \times \mathbb{Z}_q^m \tag{6}$$

and

$$(\mathbf{c}_5, \mathbf{c}_6) = \big(\mathbf{B} \cdot \mathbf{s}_2 + \mathbf{e}_5,\ \mathbf{G}_1 \cdot \mathbf{s}_2 + \mathbf{e}_6 + \mathbf{d}_{i,2} \cdot \lfloor q/p \rfloor\big) \in \mathbb{Z}_q^m \times \mathbb{Z}_q^m \tag{7}$$

for randomly chosen $\mathbf{s}_1, \mathbf{s}_2 \hookleftarrow \chi^n$, $\mathbf{e}_3, \mathbf{e}_4, \mathbf{e}_5, \mathbf{e}_6 \hookleftarrow \chi^m$. We can notice that $\mathbf{G}_0, \mathbf{G}_1$ are syndromes for the multi-bit version of Gentry-Peikert-Vaikuntanathan's IBE scheme as described in section 3.4.

3. Run the protocol of [35] in order to prove the knowledge of short vectors $\mathbf{s}_0, \mathbf{s}_1, \mathbf{s}_2 \in \mathbb{Z}^n$, $\mathbf{d}_{i,1}, \mathbf{d}_{i,2}, \mathbf{z}_i \in \mathbb{Z}^m$ and $\mathrm{id}_i \in \{0,1\}^{\ell}$ that satisfy relations (5), (6), (7) as well as

$$\mathbf{d}_{i,1}^T \cdot \mathbf{A} + \mathbf{d}_{i,2}^T \cdot \mathbf{A}_0 + \sum_{i=1} (\mathrm{id}_i[i] \cdot \mathbf{d}_{i,2}^T) \cdot \mathbf{A}_i - \mathbf{z}_i^T \cdot \mathbf{D} = \mathbf{u}^T \in \mathbb{Z}_q^n. \tag{8}$$

If we define the matrices

$$\mathbf{P}_0 = \left(\begin{array}{c|cc} \mathbf{B} & \mathbf{I}_m & \\ \mathbf{G}_0 & & \mathbf{I}_{\ell} \end{array}\right) \in \mathbb{Z}_q^{(m+\ell) \times (m+n+\ell)}, \quad \mathbf{c}_{\mathrm{id}} = \begin{pmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \end{pmatrix} \in \mathbb{Z}_q^{n+\ell}, \quad \mathbf{e}_{\mathrm{id}} = \begin{pmatrix} \mathbf{s}_0 \\ \mathbf{e}_1 \\ \mathbf{e}_2 \end{pmatrix} \in \mathbb{Z}^{m+n+\ell},$$

$$\mathbf{P}_1 = \left(\begin{array}{c|cc} \mathbf{B} & \mathbf{I}_m & \\ \mathbf{G}_1 & & \mathbf{I}_m \end{array} \right) \in \mathbb{Z}_q^{2m\times(2m+n)}, \quad \mathbf{c}_{\mathbf{d},1} = \begin{pmatrix} \mathbf{c}_3 \\ \mathbf{c}_4 \end{pmatrix} \in \mathbb{Z}_q^{2m}, \quad \mathbf{e}_{\mathbf{d},1} = \begin{pmatrix} \mathbf{s}_1 \\ \mathbf{e}_3 \\ \mathbf{e}_4 \end{pmatrix} \in \mathbb{Z}^{2m+n},$$

and

$$\mathbf{c}_{\mathbf{d},2} = \begin{pmatrix} \mathbf{c}_5 \\ \mathbf{c}_6 \end{pmatrix} \in \mathbb{Z}_q^{2m}, \quad \mathbf{e}_{\mathbf{d},2} = \begin{pmatrix} \mathbf{s}_2 \\ \mathbf{e}_5 \\ \mathbf{e}_6 \end{pmatrix} \in \mathbb{Z}^{2m+n},$$

this requires to provide a non-interactive zero-knowledge proof of knowledge π_K of short vectors $\mathrm{id}_i \in \{0,1\}^\ell$, $\mathbf{z}_{i,1}, \mathbf{d}_{i,1}, \mathbf{d}_{i,2} \in \mathbb{Z}^m$, $\mathbf{e}_{\mathrm{id}} \in \mathbb{Z}^{m+n+\ell}$,$\mathbf{e}_{\mathbf{d},1}, \mathbf{e}_{\mathbf{d},2} \in \mathbb{Z}^{2m+n}$ – where id_i is binary and $\|\mathbf{e}_{\mathrm{id}}\|_\infty \leq B$, $\|\mathbf{e}_{\mathbf{d},1}\|_\infty \leq B$, $\|\mathbf{e}_{\mathbf{d},2}\|_\infty \leq B$, $\|\mathbf{z}_{i,1}\|_\infty, \|\mathbf{d}_{i,1}\|_\infty, \|\mathbf{d}_{i,2}\|_\infty \leq \beta$ – that satisfy the relations

$$\mathbf{P}_0 \cdot \mathbf{e}_{\mathrm{id}} + \begin{pmatrix} \mathbf{I}_m & \\ & \lfloor q/2 \rfloor \mathbf{I}_\ell \end{pmatrix} \cdot \begin{pmatrix} \mathbf{0}^m \\ \mathrm{id}_i \end{pmatrix} = \mathbf{c}_{\mathrm{id}} \tag{9}$$

$$\mathbf{P}_1 \cdot \mathbf{e}_{\mathbf{d},1} + \begin{pmatrix} \mathbf{I}_m & \\ & \lfloor q/p \rfloor \mathbf{I}_m \end{pmatrix} \cdot \begin{pmatrix} \mathbf{0}^m \\ \mathbf{d}_{i,2} \end{pmatrix} = \mathbf{c}_{d,1} \tag{10}$$

$$\mathbf{P}_1 \cdot \mathbf{e}_{\mathbf{d},2} + \begin{pmatrix} \mathbf{I}_m & \\ & \lfloor q/p \rfloor \mathbf{I}_m \end{pmatrix} \cdot \begin{pmatrix} \mathbf{0}^m \\ \mathbf{d}_{i,2} \end{pmatrix} = \mathbf{c}_{d,2} \tag{11}$$

and

$$\left[\mathbf{d}_{i,1}^T \;\middle|\; \mathbf{d}_{i,2}^T \;\middle|\; \mathrm{id}_i[1]\mathbf{d}_{i,2}^T \;\middle|\; \ldots \;\middle|\; \mathrm{id}_i[\ell]\mathbf{d}_{i,2}^T \;\middle|\; \mathbf{z}_{i,1}^T \right] \cdot \left[\begin{array}{c} \mathbf{A} \\ \hline \mathbf{A}_0 \\ \hline \mathbf{A}_1 \\ \hline \vdots \\ \hline \mathbf{A}_\ell \\ \hline -\mathbf{D} \end{array} \right] = \mathbf{u}^T \bmod q. \tag{12}$$

Such a proof of knowledge can be generated by extending the proof system of [35] which is based on the decomposition-extension framework of Ling *et al.* [34] we will discuss in Appendix D.1. As in [35], this proof of knowledge is obtained from Stern's protocol [40] by repeating it $t \in \Theta(n)$ times to make the soundness error negligible. This proof of knowledge π_K is made non-interactive using the Fiat-Shamir heuristic [25] and it is made of a triple $\pi_K = (\{\mathsf{Comm}_{K,j}\}_{j=1}^t, \mathsf{Chall}_K, \{\mathsf{Resp}_{K,j}\}_{j=1}^t)$, where the challenge $\mathsf{Chall}_K = H(M, \mathsf{VK}, \mathbf{c}_{\mathbf{id}}, \mathbf{c}_{\mathbf{d},1}, \mathbf{c}_{\mathbf{d},2}, \{\mathsf{Comm}_{K,j}\}_{j=1}^t)$

4. Compute a one-time signature $sig = \mathscr{S}(\mathsf{SK}, (\mathbf{c}_{\mathrm{id}}, \mathbf{c}_{\mathbf{d},1}, \mathbf{c}_{\mathbf{d},2}, \pi_K))$.

Output the signature that consists of

$$\Sigma = \big(\mathsf{VK}, \mathbf{c}_{\mathrm{id}}, \mathbf{c}_{\mathbf{d},1}, \mathbf{c}_{\mathbf{d},2}, \pi_K, sig\big). \tag{13}$$

Verify$(\mathscr{Y}, M, \Sigma)$**:** Parse Σ as in (13). Then, return 1 if and only if: (i) $\mathscr{V}(\mathsf{VK}, (\mathbf{c}_{\mathrm{id}}, \mathbf{c}_{\mathbf{d},1}, \mathbf{c}_{\mathbf{d},2}, \pi_K), sig) = 1$; (ii) The proof of knowledge π_K properly verifies. Otherwise, return 0.

Open$(\mathscr{Y}, \mathscr{S}_{\mathsf{OA}}, M, \Sigma)$**:** Parse $\mathscr{S}_{\mathsf{OA}}$ as $\mathbf{T_B} \in \mathbb{Z}^{m\times m}$ and Σ as in (13). Then, do the following.

1. Compute $\mathbf{G}_0 = H_0(\mathsf{VK}) \in \mathbb{Z}_q^{\ell\times n}$ and $\mathbf{G}_1 = H_1(\mathsf{VK}) \in \mathbb{Z}_q^{m\times n}$. Then, using the master secret key $\mathbf{T_B} \in \mathbb{Z}^{m\times m}$ of the GPV IBE scheme, compute small-norm matrices $\mathbf{E}_{\mathrm{id}} \in \mathbb{Z}^{m\times\ell}$ and $\mathbf{E}_1 \in \mathbb{Z}^{m\times m}$ such that $\mathbf{E}_{\mathrm{id}}^T \cdot \mathbf{B} = \mathbf{G}_0 \bmod q$ and $\mathbf{E}_1^T \cdot \mathbf{B} = \mathbf{G}_1 \bmod q$.

2. Using $\mathbf{E}_{\mathrm{id}}$, decrypt $\mathbf{c}_{\mathrm{id}}$ to obtain the ℓ-bit string $\mathrm{id} \in \{0,1\}^\ell$. Then, use $\mathbf{E}_1$ to decrypt $\mathbf{c}_{\mathbf{d},1}$ (i.e., by computing $\lfloor(\mathbf{c}_4 - \mathbf{E}_1^T\mathbf{c}_3)\cdot(p/q)\rceil$) and $\mathbf{c}_{\mathbf{d},2}$ (i.e., by computing $\lfloor(\mathbf{c}_5 - \mathbf{E}_1^T\mathbf{c}_6)\cdot(p/q)\rceil$) and obtain $\mathbf{d}_{i,1}, \mathbf{d}_{i,2} \in \mathbb{Z}^m$, respectively.
3. Using the information revealed by steps 1 and 2, compute

$$\mathbf{v}^T = \mathbf{d}_{i,1}^T \cdot \mathbf{A} + \mathbf{d}_{i,2}^T \cdot \mathbf{A}_0 + \sum_{j=1}^{\ell} \mathrm{id}_i[j] \cdot \mathbf{d}_{i,2}^T \cdot \mathbf{A}_j - \mathbf{u}^T \bmod q, \tag{14}$$

and determine if the obtained vector $\mathbf{v} \in \mathbb{Z}_q^n$ appears in a record $\mathsf{transcript}_i = (\mathbf{v}, \mathrm{id}, i, \mathsf{upk}[i], sig_i)$ of the database St_{trans} for some index i. If so, output the corresponding i (and, optionally, $\mathsf{upk}[i]$). Otherwise, output $\perp$.

The most difficult part of the security proof is the security against mis-identification attacks. Briefly, we will need to consider two kinds of misidentification attacks. In Type I attacks, the adversary's fake signature $\Sigma^\star$ is opened for a new identifier $\mathrm{id}^\star \in \{0,1\}^\ell$ that was not used in any output of the $\mathscr{Q}_{\textsf{a-join}}$ oracle. Type II attacks are such that: (i) The opening of $\Sigma^\star$ reveals an identity $\mathrm{id}^\star$ for which the $\mathscr{Q}_{\textsf{a-join}}$ oracle returned a membership certificate $\mathsf{cert}_{i^\star}$ containing $\mathrm{id}^\star$, but (ii) The corresponding syndrome $\mathbf{v}^\dagger$ that the adversary chose (at step 1 of the joining oracle) at that $\mathscr{Q}_{\textsf{a-join}}$-query is not the value revealed at step 3 of the opening (14) of the forgery $\Sigma^\star$.

Type I attacks are treated in the same way as in the proofs of [31, 35]. In Type II attacks, we prove the security under the SIS assumption using a technique suggested by Böhl *et al.* [10]. In fact, the membership certificate and the membership secret form a triple $(\mathrm{id}, \mathbf{d}, \mathbf{z})$ that has the same distribution as in a variant of Boyen's signature proposed by Böhl *et al.* [10], where $\mathbf{z}$ plays the role of the signed message. For this reason, we can prove the security against Type II attacks using the same technique. Namely, the reduction has to guess in advance which output of the $\mathscr{Q}_{\textsf{a-join}}$ oracle would contain an $\mathrm{id}^\star$ that the adversary recycles in the fake signature (since N is polynomial, the adversary can guess $\mathrm{id}^\star$ with non-negligible probability). For this identifier $\mathrm{id}^\star$, the reduction computes the matrices $(\mathbf{A}, \{\mathbf{A}_j\}_{j=0}^{\ell})$ in such a way that it can compute exactly one pair $(\mathbf{d}_i^\dagger, \mathbf{z}_i^\dagger)$ of vectors satisfying (2). When running the joining protocol at the $\mathscr{Q}_{\textsf{a-join}}$-query involving $\mathrm{id}^\star$, the reduction $\mathscr{B}$ simulates the view of the adversary (by rewinding the proof of knowledge and computing $\mathbf{z}_{i,1}$ as a function of $\mathbf{z}_{i,0}$, which can be extracted from the proof of knowledge) in such a way that the membership secret becomes the vector $\mathbf{z}_i$ prepared at the beginning. Since the fake signature $\Sigma^\star$ opens to a syndrome $\mathbf{v}^\star \in \mathbb{Z}_q^n$ (at step 3 of the opening algorithm) that differs from the syndrome $\mathbf{v}^\dagger$ of the crucial $\mathscr{Q}_{\textsf{a-join}}$-query, the corresponding short vector $\mathbf{z}_i^\star \in \mathbb{Z}^m$ must also differ from $\mathbf{z}_i^\dagger$. This allows the reduction $\mathscr{B}$ to solve an instance of the SIS problem as in [10, Theorem 6.3].

The security against framing attacks is proved in a very simple manner since the adversary has to break an instance of an ISIS problem to frame a honest user. In fact, we can prove the security of the scheme by having the reduction honestly choose vectors $\mathbf{z}_{i,0}$ and sending $\mathbf{v}_{i,0}^T = \mathbf{z}_{i,0}^T \cdot \mathbf{D}$ to the adversary (which plays the role of the GM). By rewinding the adversary several times using the Improved Forking Lemma of Brickell *et al.* [15], we can extract another short vector $\mathbf{z}_{i,0}^\star$ such that $\mathbf{v}_{i,0}^T = {\mathbf{z}_{i,0}^\star}^T \cdot \mathbf{D}$. Since $\mathbf{z}_{i,0}^\star \neq \mathbf{z}_{i,0}$ with high probability, the reduction obtains a non-trivial short vector $\mathbf{w} = \mathbf{z}_{i,0}^\star - \mathbf{z}_{i,0}$ in $\Lambda^\perp(\mathbf{D})$.

Finally, the anonymity property is proved exactly in the same way as in [35] and there is no specific difficutly to solve here.

5 Conclusion

In a scientific point of view, we still have to improve the exact choice of parameters that will allow the security proof of our scheme to work while making it more efficient. In the current construction, our scheme requires a modulus q of exponential size while the static construction of [35] only requires a modulus q larger than $\log N$.

In a more personal point of view, this internship was the opportunity to get more familiar with lattice-based cryptography and advanced (anonymity-related) cryptographic protocols in particular. Designing such an advanced primitive was the occasion to manipulate objects at different scales and see how they interact to each other. It taught me not to underestimate the challenges that may be encountered in order to correctly tune the parameters of a scheme. For example, the constraints to take into account in a security proof are likely to importantly affect the choice of parameters, even if the scheme seems to work (and remain secure) for a more efficient choice of parameters.

References

[1] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H) IBE in the standard model. In *Advances in Cryptology–EUROCRYPT 2010*, pages 553–572. Springer, 2010.

[2] Joël Alwen and Chris Peikert. Generating shorter bases for hard random lattices. *Theory of Computing Systems*, 48(3):535–553, 2011.

[3] G. Asharov, A. Jain, A. Lopez-Alt, E. Tromer, V. Vaikuntanathan, and D. Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In *Eurocrypt 2012*, volume 7237, pages 483–501, 2012.

[4] Giuseppe Ateniese and Gene Tsudik. Some open issues and new directions in group signatures. In *Financial Cryptography*, pages 196–211. Springer, 1999.

[5] Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *Advances in Cryptology—CRYPTO 2000*, pages 255–270. Springer, 2000.

[6] Wojciech Banaszczyk. New bounds in some transference theorems in the geometry of numbers. *Mathematische Annalen*, 296(1):625–635, 1993.

[7] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *Advances in Cryptology—Eurocrypt 2003*, pages 614–629. Springer Berlin Heidelberg, 2003.

[8] Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of group signatures: The case of dynamic groups. In *Topics in Cryptology–CT-RSA 2005*, pages 136–153. Springer, 2005.

[9] Olivier Blazy and Céline Chevalier. Generic Construction of UC-Secure Oblivious Transfer. In *ACNS*, Lecture Notes in Computer Science, New York, USA, June 2015. Springer.

[10] Florian Böhl, Dennis Hofheinz, Tibor Jager, Jessica Koch, and Christoph Striecks. Confined guessing: New signatures from standard assumptions. *Journal of Cryptology*, 28(1):176–208, 2015.

[11] Dan Boneh and Hovav Shacham. Group signatures with verifier-local revocation. In *Proceedings of the 11th ACM conference on Computer and communications security*, pages 168–177. ACM, 2004.

[12] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *Advances in Cryptology–CRYPTO 2004*, pages 41–55. Springer, 2004.

[13] Xavier Boyen. Lattice mixing and vanishing trapdoors: A framework for fully secure short signatures and more. In *Public Key Cryptography–PKC 2010*, pages 499–517. Springer, 2010.

[14] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 575–584. ACM, 2013.

[15] E. Brickell, D. Pointcheval, S. Vaudenay, and M. Yung. Design validations for discrete logarithm based signature schemes. In *Public Key Cryptography (PKC) 2000*, pages 276–292, 2000.

[16] Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups. In *Advances in Cryptology—CRYPTO'97*, pages 410–424. Springer, 1997.

[17] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In *Advances in Cryptology-Eurocrypt 2004*, pages 207–222. Springer, 2004.

[18] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. *Advances in Cryptology–EUROCRYPT 2010*, pages 523–552, 2010.

[19] David Chaum and Eugène Van Heyst. Group signatures. In *Advances in Cryptology—EUROCRYPT'91*, pages 257–265. Springer, 1991.

[20] Lidong Chen and Torben P Pedersen. New group signature schemes. In *Advances in Cryptology—EUROCRYPT'94*, pages 171–181. Springer, 1995.

[21] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Advances in Cryptology—CRYPTO'94*, pages 174–187. Springer, 1994.

[22] Ivan Damgård. On Σ-protocols. Manuscript, available at `http://www.cs.au.dk/~ivan/Sigma.pdf`, 2010.

[23] Cécile Delerablée and David Pointcheval. Dynamic fully anonymous short group signatures. In *Progress in Cryptology-VIETCRYPT 2006*, pages 193–210. Springer, 2006.

[24] Luca Delgrossi and Tao Zhang. Dedicated short-range communications. *Vehicle Safety Communications: Protocols, Security, and Privacy*, pages 44–51, 2009.

[25] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology—CRYPTO'86*, pages 186–194. Springer, 1987.

[26] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 197–206. ACM, 2008.

[27] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems. In *Proceedings of the seventeenth annual ACM symposium on Theory of computing*, pages 291–304. ACM, 1985.

[28] S Dov Gordon, Jonathan Katz, and Vinod Vaikuntanathan. A group signature scheme from lattice assumptions. In *Advances in Cryptology-ASIACRYPT 2010*, pages 395–412. Springer, 2010.

[29] Club Inutile. The password security checker. Website, available at `http://inutile.club/estatis/password-security-checker/`, 2012.

[30] Aggelos Kiayias and Moti Yung. Secure scalable group signature with dynamic joins and separable authorities. *International Journal of Security and Networks*, 1(1):24–45, 2006.

[31] Fabien Laguillaumie, Adeline Langlois, Benoît Libert, and Damien Stehlé. Lattice-based group signatures with logarithmic signature size. In *Advances in Cryptology-ASIACRYPT 2013*, pages 41–61. Springer, 2013.

[32] Fabien Laguillaumie, Adeline Langlois, and Damien Stehlé. Chiffrement avancé à partir du problème learning with errors. *Informatique Mathématique une photographie en*, pages 179–225, 2014.

[33] Adeline Langlois, San Ling, Khoa Nguyen, and Huaxiong Wang. Lattice-based group signature scheme with verifier-local revocation. In *Public-Key Cryptography–PKC 2014*, pages 345–361. Springer, 2014.

[34] San Ling, Khoa Nguyen, Damien Stehlé, and Huaxiong Wang. Improved zero-knowledge proofs of knowledge for the isis problem, and applications. In *Public-Key Cryptography–PKC 2013*, pages 107–124. Springer, 2013.

[35] San Ling, Khoa Nguyen, and Huaxiong Wang. Group signatures from lattices: Simpler, tighter, shorter, ring-based. In Jonathan Katz, editor, *Public-Key Cryptography – PKC 2015*, volume 9020 of *Lecture Notes in Computer Science*, pages 427–449. Springer Berlin Heidelberg, 2015. ISBN 978-3-662-46446-5. doi: 10.1007/978-3-662-46447-2_19. URL `http://dx.doi.org/10.1007/978-3-662-46447-2_19`.

[36] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *Advances in Cryptology–EUROCRYPT 2012*, pages 700–718. Springer, 2012.

[37] Phong Q Nguyen, Jiang Zhang, and Zhenfeng Zhang. Simpler efficient group signatures from lattices. In *Public Key Cryptography*, 2015.

[38] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM*, 56(6):34, 2009. Preliminary version in STOC'05.

[39] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM journal on computing*, 26(5):1484–1509, 1997.

[40] Jacques Stern. A new paradigm for public key identification. *Information Theory, IEEE Transactions on*, 42 (6):1757–1768, 1996.

Appendix

Table of Contents

A Ling *et al.*'s static group signature

We based our scheme on the Ling *et al.* [35] group signature scheme. It is defined as follows.

Let n be the security parameter, and $N = \mathsf{Poly}\left(()\, n\right)$ be the maximum expected number of group users. Then we choose other scheme parameters such that Boyen's signature scheme (described in C) and the GPV IBE scheme (described in 3.4) function properly and are secure.

We choose hash functions $H_1 : \{0,1\}^* \to \mathbb{Z}_q^{n\times \ell}$ – which goal is to produce ℓ random syndromes for multibit GPV – and $H_2 : \{0,1\}^* \to \{1,2,3\}^t$ which goal is to provide challenges for the Stern's protocol, and select a one time signature scheme $\Pi^{\mathrm{OTS}} = (\mathscr{G}, \mathscr{S}, \mathscr{V})$. Let χ denote the b-bounded LWE distribution over $\mathbb{Z}$.

The group signature is described as follows:

KeyGen$(1^n, 1^N)$**:** 1. Generate verification key $(\mathbf{A}, \mathbf{A}_0, \ldots, \mathbf{A}_\ell, \mathbf{u})$ and signing key $\mathbf{T_A}$ for Boyen's signature scheme. Then for each $\mathrm{id} = (\mathrm{id}_1, \ldots, \mathrm{id}_\ell) \in \{0,1\}^\ell$, use $\mathbf{T_A}$ to generates gsk[id] as Boyen signature on message id.

2. Generate encrypting and decrypting master keys pair $(\mathbf{B}, \mathbf{T_B})$ for the GPV-IBE scheme.
3. Output

$$\mathsf{gpk} = ((\mathbf{A}, \mathbf{A}_0, \ldots, \mathbf{A}_\ell, \mathbf{u}), \mathbf{B}); \quad \mathsf{gmsk} = \mathbf{T_B}; \quad \mathsf{gsk} = \{\mathsf{gsk}[\mathrm{id}]\}_{\mathrm{id}\in\{0,1\}^\ell}.$$

Sign$(\mathsf{gsk}[\mathrm{id}], M)$**:** Given gpk, to sign a message $M \in \{0,1\}^*$ using the secret key $\mathsf{gsk}[\mathrm{id}] = \mathbf{z}$, the user generates a key pair $(\mathsf{VK}, \mathsf{SK}) \leftarrow \mathscr{G}(1^n)$ for Π^{OTS}, and ther perform the following steps:

1. Encrypt the index id with respect to identity VK using the GPV IBE. Namely we compute:

$$(\mathbf{c}_1 = \mathbf{B}^T\mathbf{s} + \mathbf{e}_1, \mathbf{c}_2 = \mathbf{G}^T\mathbf{s} + \mathbf{e}_2 + \lfloor q/2 \rfloor \mathrm{id}) \in \mathbb{Z}_q^m \times \mathbb{Z}_q^\ell$$

with $\mathbf{e}_1, \mathbf{e_2}, \mathbf{s}$ sampled from χ with the right dimension.

2. Generate a NIZKPoK Π to show the possession of a valid message-signature pair $(\mathrm{id}, \mathbf{z})$ for Boyen-s signature and that $(\mathbf{c}_1, \mathbf{c}_2)$ is a correct GPV IBE encryption of d with respect to identity VK. This is done as in 4.2. The message is embed in the challenge of the Fiat-Shamir method to construct NIZK proofs.
3. Compute a one-time signature $sig = \mathscr{S}(\mathsf{VK}; \mathbf{c}_1, \mathbf{c}_2, \Pi)$

4. Output the group signature $\Sigma = (\mathsf{VK}, (\mathbf{c}_1, \mathbf{c}_2), \Pi, sig)$

Verify$(\mathsf{gpk}, M, \Sigma)$**:** Parse Σ as before. Check that $\mathcal{V}(\mathsf{VK}; sig, (\mathbf{c}_1, \mathbf{c}_2), \Pi) = 1$, otherwise return 0.

Verify that Π is a valid proof, otherwise return 0.

If all verifications pass, return 1.

Open$(\mathbf{T_B}, M, \Sigma)$**:** On input $\mathsf{gmsk} = \mathbf{T_B}$ and a signature $\Sigma = (\mathsf{VK}, (\mathbf{c}_1, \mathbf{c}_2), \Pi, sig)$, this algorithm decrypt $(\mathbf{c}_1, \mathbf{c}_2)$ using GPV IBE decrypt with respect to the identity VK and returns the decrypted identity id.

The underlying ideas of this scheme are developed in section 4.1, as our scheme is based on this one.

B Laguillaumie *et al.*'s Group Signature for Static Groups

Before continuing we need in the Laguillaumie *et al.* [31] protocol the following algorithms to complete section 3.1.3.

Lemma 4 was extended by Gordon *et al.* [28] so that the columns of the generated matrix $\mathbf{A}$ is conditioned to lie within a given linear vector space of $\mathbb{Z}_q^n$ (for q prime):

Lemma 6. *There exists a* PPT *algorithm* SuperSamp *that takes as inputs matrices* $\mathbf{B} \in \mathbb{Z}_q^{m\times n}$ *and* $\mathbf{C} \in \mathbb{Z}_q^{n\times n}$ *such that the rows of* $\mathbf{B}$ *span* $\mathbb{Z}_q^n$*,* $m \geq n \geq 1$*, and* $q \geq 2$ *prime such that* $m \geq \Omega(n \log q)$*. It outputs* $\mathbf{A} \in \mathbb{Z}_q^{m\times n}$ *and a basis* $\mathbf{T_A}$ *of* $\Lambda_q^{\perp}(\mathbf{A})$ *such that* $\mathbf{A}$ *is within statistical distance* $2^{-\Omega(n)}$ *to* $U(\mathbb{Z}_q^{m\times n})$ *conditioned on* $\mathbf{B}^T \cdot \mathbf{A} = \mathbf{C}$*, and* $\|\widetilde{\mathbf{T_A}}\| \leq \mathcal{O}(\sqrt{mn \log q \log m})$*.*

It is useful to obtain a randomized basis without changing the underlying lattice, which purpose is to continue working on the same object without knowing on what we are working at, which is important to guarantee privacy for instance. This is why we may want be able to randomize a basis $\mathbf{B}$ to obtain an *independent* equivalent basis $\mathbf{C}$, in order to provide the long blue basis from the short red basis in figure 1.

Lemma 7 (Adapted from [18, Le. 3.3])**.** *There exists a* PPT *algorithm* RandBasis *that takes as inputs a basis* $\mathbf{B}$ *of a lattice* $L \subseteq \mathbb{Z}^n$ *and a rational* $\sigma \geq \|\widetilde{\mathbf{B}}\| \cdot \Omega(\sqrt{\log n})$*, and outputs a basis* $\mathbf{C}$ *of* L *satisfying* $\|\widetilde{\mathbf{C}}\| \leq \sqrt{n}\sigma$ *with probability* $\geq 1 - 2^{-\Omega(n)}$*. Further, the distribution of* $\mathbf{C}$ *is independent of the input basis* $\mathbf{B}$*.*

In this section we will present the Laguillaumie *et al.* [31] protocol we used as a basis to build our protocol.

Keygen$(1^n, 1^N)$**:** Given a security parameter $n > 0$ and the desired number of group members $N = 2^\ell \in \mathsf{Poly}(n)$, choose parameters q, m, p, α and σ as specified in section 4.2 and make them public. Choose a hash function $H : \{0,1\}^* \to \{0,1\}^t$ for some $t = \Theta(n)$, which will be modeled as a random oracle in the security proof. Then, proceed as follows.

1. Run $\mathsf{TrapGen}(1^n, 1^m, q)$ to get $\mathbf{A} \in \mathbb{Z}_q^{m\times n}$ and a short basis $\mathbf{T_A}$ of $\Lambda_q^{\perp}(\mathbf{A})$.
2. For $i = 0$ to ℓ, sample $\mathbf{A}_i \hookleftarrow U(\mathbb{Z}_q^{m\times n})$ and compute $(\mathbf{B}_i, \mathbf{S}'_i) \leftarrow \mathsf{SuperSamp}(\mathbf{A}_i, \mathbf{0})$. Then, randomize $\mathbf{S}'_i$ as $\mathbf{S}_i \leftarrow \mathsf{RandBasis}(\mathbf{S}'_i, \Omega(\sqrt{mn \log q} \log m))$.[1]
3. For $j = 0$ to $N-1$, let $\mathrm{id}_j = \mathrm{id}_j[1] \ldots \mathrm{id}_j[\ell] \in \{0,1\}^\ell$ be the binary representation of id_j and define:
$$\mathbf{A}_{\mathrm{id}_j} = \left[\begin{array}{c}\mathbf{A} \\ \hline \mathbf{A}_0 + \sum_{i=1}^{\ell} \mathrm{id}_j[i]\mathbf{A}_i\end{array}\right] \in \mathbb{Z}_q^{2m\times n}. \tag{15}$$
Then, run $\mathbf{T}'_{\mathrm{id}_j} \leftarrow \mathsf{ExtBasis}(\mathbf{A}_{\mathrm{id}_j}, \mathbf{T_A})$ to get a short delegated basis $\mathbf{T}'_{\mathrm{id}_j}$ of $\Lambda_q^{\perp}(\mathbf{A}_{\mathrm{id}_j})$. Finally, run $\mathbf{T}_{\mathrm{id}_j} \leftarrow \mathsf{RandBasis}(\mathbf{T}'_{\mathrm{id}_j}, \Omega(m\sqrt{\ell n \log q \log m}))$.[1] The j-th member's private key is $\mathsf{gsk}[j] := \mathbf{T}_{\mathrm{id}_j}$.

[1] These randomisation steps are not needed for the correctness of the scheme but are important in the traceability proof.

4. The group manager's private key is $\mathsf{gmsk} := \{\mathbf{S}_i\}_{i=0}^{\ell}$ and the group public key is defined to be $\mathsf{gpk} := \big(\mathbf{A}, \{\mathbf{A}_i, \mathbf{B}_i\}_{i=0}^{\ell}\big)$. The algorithm outputs $\big(\mathsf{gpk}, \mathsf{gmsk}, \{\mathsf{gsk}[j]\}_{j=0}^{N-1}\big)$.

Sign$(\mathsf{gpk}, \mathsf{gsk}[j], M)$**:** To sign a message $M \in \{0,1\}^*$ using the private key $\mathsf{gsk}[j] = \mathbf{T}_{\mathrm{id}_j}$, proceed as follows.

1. Run $\mathsf{GPVSample}(\mathbf{T}_{\mathrm{id}_j}, \sigma)$ to get $(\mathbf{x}_1^T | \mathbf{x}_2^T)^T \in \Lambda_q^{\perp}(\mathbf{A}_{\mathrm{id}_j})$ of norm $\leq \sigma\sqrt{2m}$.
2. Sample $\mathbf{s}_0 \hookleftarrow U(\mathbb{Z}_q^n)$ and encrypt $\mathbf{x}_2 \in \mathbb{Z}_q^m$ as $\mathbf{c}_0 = \mathbf{B}_0 \cdot \mathbf{s}_0 + \mathbf{x}_2 \in \mathbb{Z}_q^m$.
3. Sample $\mathbf{s} \hookleftarrow U(\mathbb{Z}_q^n)$. For $i = 1$ to ℓ, sample $\mathbf{e}_i \hookleftarrow D_{\mathbb{Z}^m, \alpha q}$ and compute $\mathbf{c}_i = \mathbf{B}_i \cdot \mathbf{s} + p \cdot \mathbf{e}_i + \mathrm{id}_j[i] \cdot \mathbf{x}_2$, which encrypts $\mathbf{x}_2 \in \mathbb{Z}_q^m$ (resp. $\mathbf{0}$) if $\mathrm{id}_j[i] = 1$ (resp. $\mathrm{id}_j[i] = 0$).
4. Generate a NIZKPoK π_0 of $\mathbf{s}_0$ so that $(\mathbf{B}_0, \mathbf{c}_0, \sqrt{2}\sigma/q; \mathbf{s}_0) \in R_{\mathsf{LWE}}$ (see Section 3.1.2).
5. For $i = 1$ to ℓ, generate a NIZKPoK $\pi_{\mathrm{OR},i}$ of $\mathbf{s}$ and $\mathbf{s}_0$ so that either:
   (i) $((\mathbf{B}_i|\mathbf{B}_0), p^{-1}(\mathbf{c}_i - \mathbf{c}_0), \sqrt{2}\alpha; (\mathbf{s}^T | -\mathbf{s}_0^T)^T) \in R_{\mathsf{LWE}}$ (the vectors $\mathbf{c}_i$ and $\mathbf{c}_0$ encrypt the same $\mathbf{x}_2$, so that $p^{-1}(\mathbf{c}_i - \mathbf{c}_0)$ is close to the $\mathbb{Z}_q$-span of $(\mathbf{B}_i|\mathbf{B}_0)$);
   (ii) or $(\mathbf{B}_i, p^{-1}\mathbf{c}_i, \alpha; \mathbf{s}) \in R_{\mathsf{LWE}}$ (the vector $\mathbf{c}_i$ encrypts $\mathbf{0}$, so that $p^{-1}\mathbf{c}_i$ is close to the $\mathbb{Z}_q$-span of $\mathbf{B}_i$).

   As explained in remark 2, this can be achieved by OR-ing two proofs for R_{LWE}. The resulting protocol is then made non-interactive using the Fiat-Shamir heuristic.
6. For $i = 1$ to ℓ, set $\mathbf{y}_i = \mathrm{id}_j[i]\mathbf{x}_2 \in \mathbb{Z}^m$ and generate a NIZKPoK π_K of $\{\mathbf{e}_i\}_{i=0}^{\ell}, \{\mathbf{y}_i\}_{i=0}^{\ell}, \mathbf{x}_1$ such that,

$$\mathbf{x}_1^T\mathbf{A} + \sum_{i=0}^{\ell} \mathbf{c}_i^T \mathbf{A}_i = \sum_{i=1}^{\ell} \mathbf{e}_i^T \big(p\mathbf{A}_i\big) \quad \text{and} \quad \mathbf{e}_i^T\big(p\mathbf{A}_i\big) + \mathbf{y}_i^T\mathbf{A}_i = \mathbf{c}_i^T\mathbf{A}_i \qquad \text{for } i \in [1,\ell] \tag{16}$$

   with $\|\mathbf{e}_i\|, \|\mathbf{y}_i\|, \|\mathbf{x}_1\| \leq \max(\sigma, \alpha q)\sqrt{m}$ for all i.
   This is achieved using $\mathsf{Prove}_{\mathsf{ISIS}}$ in order to produce a triple $(\mathsf{Comm}_K, \mathsf{Chall}_K, \mathsf{Resp}_K)$, where the challenge $\mathsf{Chall}_K = H(M, \mathsf{Comm}_K, \{\mathbf{c}_i\}_{i=0}^{\ell}, \pi_0, \{\pi_{\mathrm{OR},i}\}_{i=0}^{\ell})$.

The signature consists of

$$\Sigma = \big(\{\mathbf{c}_i\}_{i=0}^{\ell}, \pi_0, \{\pi_{\mathrm{OR},i}\}_{i=0}^{\ell}, \pi_K\big). \tag{17}$$

Verify$(\mathsf{gpk}, M, \Sigma)$**:** Parse Σ as in (17). Then, return 1 if $\pi_0, \{\pi_{\mathrm{OR},i}\}_{i=0}^{\ell}, \pi_K$ properly verify. Else, return 0.

Open$(\mathsf{gpk}, \mathsf{gmsk}, M, \Sigma)$**:** Parse gmsk as $\{\mathbf{S}_i\}_{i=0}^{\ell}$ and Σ as in (17). Compute $\mathbf{x}_2$ by decrypting $\mathbf{c}_0$ using $\mathbf{S}_0$. For $i = 1$ to ℓ, use $\mathbf{S}_i$ to determine which one of the vectors $p^{-1}\mathbf{c}_i$ and $p^{-1}(\mathbf{c}_i - \mathbf{x}_2)$ is close to the $\mathbb{Z}_q$-span of $\mathbf{B}_i$. Set $\mathrm{id}[i] = 0$ in the former case and $\mathrm{id}[i] = 1$ in the latter. Eventually, output $\mathrm{id} = \mathrm{id}[1] \ldots \mathrm{id}[\ell]$.

C Boyen's signature scheme

In our protocol described in section 4.2, the certificate is a variant of the Boyen's signature scheme [13] that has been first described in [36].

Definition 16 (Signature scheme)**.** A signature scheme is a triple of algorithm $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ such that Gen on security parameter n returns a pair of keys $(\mathsf{SK}, \mathsf{VK})$; Sign on a message M with the signing key SK returns a signature σ; and Verify on a message M a signature σ and the verification key VK returns 1 meaning that the signature is accepted or 0 otherwise.

Definition 17 (Boyen's signature scheme [36, sec 6.2.2])**.** The Boyen's signature variant is defined as follows.

$\textsf{Gen}(1^n)$**:** Generate $(\mathbf{A}, \mathbf{T_A}) \in \mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^{m\times m}$ using $\mathsf{TrapGen}$ algorithm. For $i = 1, \ldots, \ell$, sample uniform matrices $\mathbf{A}_i \hookleftarrow \mathscr{U}(\mathbb{Z}_q^{m\times n})$. And choose a syndrome $\mathbf{u} \hookleftarrow \mathbb{Z}_q^n$.

The public verification key is $\mathsf{VK} = (\mathbf{A}, \{\mathbf{A_i}\}_{i=1}^{\ell}, \mathbf{u})$ and the secret signing key it $\mathbf{T_A}$

$\textsf{Sign}(\mathbf{T_A}, M)$**:** Let $\mathbf{A}_M = \left[\frac{\mathbf{A}}{\mathbf{A}_0 + \sum_{i=1}^{\ell} M_i \mathbf{A}_i}\right] \in \mathbb{Z}_q^{2m\times n}$ where M_i is the i-th bit of the binary decomposition of M. Output $\mathbf{v} \in \mathbb{Z}^m$ sampled from $D_{\Lambda_{\mathbf{u}}^{\perp}(\mathbf{A}_M),\sigma}$ using $\mathsf{GPVSample}(T_{A_M}, \sigma)$ obtaining $\mathbf{T}_{\mathbf{A}_M}$ from $\mathsf{ExtBasis}(\mathbf{A}, \mathbf{A}_0 + \sum_{i=1}^{\ell} M_i \mathbf{A}_i, \mathbf{T_A})$.

$\textsf{Verify}(\mathsf{VK}, M, \mathbf{v})$**:** Let $\mathbf{A}_M$ be as above. Accept if $\|\mathbf{v}\| \leq \sigma\sqrt(n)$ and $\mathbf{v}^T \cdot \mathbf{A}_M = \mathbf{u}$; otherwise reject.

D The Interactive Proof Systems

In this appendix we will present the different proofs system used in our scheme.

D.1 For Boyen's Signature

First of all, we need to prove that the certification of the syndrome in the Join algorithm went well.

The interactive proof system used in sections 3.4 and 4.2 is a variant of the Stern's proof system [40] to prove the knowledge of a valid signature pair $(d, \mathbf{z})$ for the Boyen's signature, or in a more general sense to prove the knowledge of an ISIS solution. Its particularity is to require three transcripts to build back the secret. Then its security is based on a variant of the classical forking lemma [15]. It is described as follows.

S_n is the set of permutations over $1, \ldots, n$, and COM is a statistically hiding commitment scheme.

Commitment: P samples

$$\begin{cases} \mathbf{r_z}^{(1)}, \ldots, \mathbf{r_z}^{p} \hookleftarrow \mathscr{U}(\mathbb{Z}_q^{(2\ell+2)3m});\ \mathbf{r_e}^{(1)}, \ldots, \mathbf{r_e}^{(p')} \hookleftarrow \mathscr{U}(\mathbb{Z}_q^{3k_2});\ \mathbf{r}_d \hookleftarrow \mathscr{U}(\mathbb{Z}_q^{2\ell}) \\ \tau \hookleftarrow \mathsf{S}_{2\ell};\ \pi_1, \ldots, \pi_p, \psi_1, \ldots, \psi_p \hookleftarrow \mathsf{S}_{3m};\ \phi_1, \ldots, \phi_{p'} \hookleftarrow \mathsf{S}_{3k_2} \end{cases}$$

Then P send the commitment CMT $= (\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3)$ to V where

$$\begin{cases} \mathbf{c}_1 = \mathsf{COM}\left(\tau;\ \{\pi_j\}_{j=1}^{p};\ \{\psi_j\}_{j=1}^{p};\ \{\phi_j\}_{j=1}^{p'};\ \mathbf{A}^*\left(\sum_{j=1}^{p} \beta_j \mathbf{r_z}^{(j)}\right);\ \mathbf{P}^*\left(\sum_{j=1}^{p'} b_j \mathbf{r_e}^{(j)}\right) + \mathbf{Q}_{\mathbf{r}_d}\right) \\ \mathbf{c}_2 = \mathsf{COM}\left(\{F_{\pi_j,\psi_j,\tau}(\mathbf{r_z}^{(j)})\}_{j=1}^{p};\ \{\phi_j(\mathbf{r_e}^{(j)})\}_{j=1}^{p'};\ \tau(\mathbf{r}_d)\right) \\ \mathbf{c}_3 = \mathsf{COM}\left(\{F_{\pi_j,\psi_j,\tau}(\mathbf{z}_j + \mathbf{r_z}^{(j)})\}_{j=1}^{p};\ \{\phi_j(\mathbf{e}_j + \mathbf{r_e}^{(j)})\}_{j=1}^{p'};\ \tau(d^* + \mathbf{r}_d)\right) \end{cases}$$

Challenge: V sends a challenge $Ch \hookleftarrow \{1, 2, 3\}$ to P.

Response: Depending on Ch, P computes the response RSP as follows:

- Case $Ch = 1$: For each $j \in 1, \ldots, p$, let $\mathbf{t}_{\mathbf{z}}^{(j)} = F_{\pi_j,\psi_j,\tau}(\mathbf{z}_j)$ and $\mathbf{v_z}^{(j)} = F_{\pi_j,\psi_j,\tau}(\mathbf{r_z}^{(j)})$. For each $j = 1, \ldots, p'$, let $\mathbf{t_e}^{(j)} = \psi(\mathbf{e}_j)$ and $\mathbf{v_e}^{(j)} = \psi(\mathbf{r_e}^{(j)})$. Let $\mathbf{t}_d = \tau(d^*)$ and $\mathbf{v}_d = \tau(\mathbf{r}_d)$. Then the prover sends:

$$\mathsf{RSP} = \left(\{\mathbf{t}_{\mathbf{z}}^{(j)}\}_{i=1}^{p};\ \{\mathbf{v_z}^{(j)}\}_{i=1}^{p};\ \{\mathbf{t_e}^{(j)}\}_{i=1}^{p'};\ \{\mathbf{v_e}^{(j)}\}_{i=1}^{p'};\ \mathbf{t}_d;\ \mathbf{v}_d\right). \tag{18}$$

- Case $Ch = 2$: For each $j = 1, \ldots, p$, let $\mathbf{w_z}^{(j)} = \mathbf{z_j} + \mathbf{r_z}^{(j)}$. For each $j = 1, \ldots, p'$, let $\mathbf{w_e}^{(j)} = \mathbf{e}_j + \mathbf{r}_\mathbf{e}^{(j)}$. Let $\mathbf{w}_d = d^* + \mathbf{r}_d$. Then the prover sends:

$$\mathsf{RSP} = \left(\tau; \{\pi_j\}_{j=1}^{p}; \{\phi_j\}_{j=1}^{p}; \{\psi_j\}_{j=1}^{p'}; \left\{\mathbf{w}_\mathbf{z}^{(j)}\right\}_{j=1}^{p}; \left\{\mathbf{w}_\mathbf{e}^{(j)}\right\}_{j=1}^{p'}; \mathbf{w}_d\right). \qquad (19)$$

- Case $Ch = 3$: For each $j = 1, \ldots, p$, let $\mathbf{y}_\mathbf{z}^{(j)} = \mathbf{r}_\mathbf{z}^{(j)}$. For each $j = 1, \ldots, p'$, let $\mathbf{y}_\mathbf{e}^{(j)} = \mathbf{r}_\mathbf{e}^{(j)}$. Let $y_d = r_d$. Then the prover sends:

$$\mathsf{RSP} = \left(\tau; \{\pi_j\}_{j=1}^{p}; \{\phi_j\}_{j=1}^{p}; \{\psi_j\}_{j=1}^{p'}; \left\{\mathbf{y}_\mathbf{z}^{(j)}\right\}_{j=1}^{p}; \left\{\mathbf{y}_\mathbf{e}^{(j)}\right\}_{j=1}^{p'}; \mathbf{y}_d\right) \qquad (20)$$

Verification: Receiving RSP, the verifier proceeds as follows:

- Case $Ch = 1$: Parse RSP as in (18), check that $\mathbf{t} \in \mathbf{B}_{2\ell}; \mathbf{t}_\mathbf{z}^{(j)} \in \mathsf{VALID}(\mathbf{t_d}), \forall j \in 1, \ldots, p; \mathbf{t}_\mathbf{e}^{(j)} \in \mathbf{B}_{3k_2}, \forall j \in 1, \ldots, p'$; and that:

$$\begin{cases} \mathbf{c}_2 = \mathsf{COM}\left(\{\mathbf{v}_\mathbf{j}^{(j)}\}_{j=1}^{p};\ \{\mathbf{v}_\mathbf{e}^{(j)}\}_{j=1}^{p'};\ \mathbf{v}_d\right) \\ \mathbf{c}_3 = \mathsf{COM}\left(\{\mathbf{t}_\mathbf{z}^{(j)} + \mathbf{v_z}^{(j)}\}_{j=1}^{p};\ \{\mathbf{t}_\mathbf{e}^{(j)} + \mathbf{v}_\mathbf{e}^{(j)}\}_{j=1}^{p'};\ \mathbf{t}_d + \mathbf{v}_d\right). \end{cases}$$

- Case $Ch = 2$: Parse RSP as in (19), check that:

$$\begin{cases} \mathbf{c}_1 = \mathsf{COM}\left(\tau;\ \{\pi_j\}_{j=1}^{p};\ \{\psi_j\}_{j=1}^{p};\ \{\phi_j\}_{j=1}^{p'};\ \mathbf{A}^*\left(\sum_{j=1}^{p} \beta_j \mathbf{w_z}^{(j)} - \mathbf{u}\right);\ \mathbf{P}^*\left(\sum_{j=1}^{p'} b_j \mathbf{w_e}^{(j)}\right) + \mathbf{Q}_{\mathbf{w}_d} - \mathbf{c}\right) \\ \mathbf{c}_3 = \mathsf{COM}\left(\{F_{\pi_j,\psi_j,\tau}(\mathbf{w_z})\}_{j=1}^{p};\ \{\phi_j(\mathbf{w_e}^{(j)})\}_{j=1}^{p'};\ \tau(\mathbf{w}_d)\right) \end{cases}$$

- Case $Ch = 3$: Parse RSP as in (20), check that:

$$\begin{cases} \mathbf{c}_1 = \mathsf{COM}\left(\tau;\ \{\pi_j\}_{j=1}^{p};\ \{\psi_j\}_{j=1}^{p};\ \{\phi_j\}_{j=1}^{p'};\ \mathbf{A}^*\left(\sum_{j=1}^{p} \beta_j \mathbf{y_z}^{(j)}\right);\ \mathbf{P}^*\left(\sum_{j=1}^{p'} b_j \mathbf{y_e}^{(j)}\right) + \mathbf{Q}_{\mathbf{y}_d}\right) \\ \mathbf{c}_2 = \mathsf{COM}\left(\{F_{\pi_j,\psi_j,\tau}(\mathbf{y_z}^{(j)})\}_{j=1}^{p};\ \{\phi_j(\mathbf{y_e}^{(j)})\}_{j=1}^{p'};\ \tau(\mathbf{y}_d)\right) \end{cases}$$

In each case, V output 1 if and only if all the conditions hold. Otherwise, it outputs 0.

Let us explain the previous protocol. We want to prove the knowledge of a pair $(M, \mathbf{v})$ where $\mathbf{v}$ is a Boyen signature. Namely we want to prove that we know $\mathbf{v}$ such that: $\mathbf{v}^T \mathbf{A}_M = \mathbf{u} \bmod q$, with $\mathbf{A}_M$ defined as in appendix C, and $\|\mathbf{v}\| \leq \beta$ for some bound β. We can notice that $(\mathbf{A}_M, \mathbf{v})$ is a valid pair belonging to the ISIS relation defined in section D.2. Thus we can prove it using the *decomposition-extension* framework we will discuss later on.

But this is not enough, we have to prove that $\mathbf{A}_M$ is also well formed. To do this we rewrite the problem $\mathbf{v}^T \mathbf{A}_M = \mathbf{u} \bmod q$ as $\bar{\mathbf{z}}^T \bar{\mathbf{A}} = \mathbf{u} \bmod q$ with $\bar{\mathbf{z}} = (\mathbf{d}_1 || \mathbf{d_2} || \mathrm{id}_0 \mathbf{d}_2 || \ldots || \mathrm{id}_\ell \mathbf{d}_2)$ and $\bar{\mathbf{A}} = \begin{bmatrix} \mathbf{A} \\ \hline \mathbf{A_0} \\ \hline \vdots \\ \hline \mathbf{A}_\ell \end{bmatrix}$ and we prove the knowledge of a binary vector id satisfying the previous relation. To do this we use the same extension method as previously, and then prove that id is indeed binary.

D.2 Decomposition-Extension Framework

The idea of the last proof is to exploit algebraic aspects of the ISIS problem. We want to prove the knowledge of a witness $\mathbf{x}$ in the R_{ISIS} relation. Namely:

$$R_{\mathsf{ISIS}^\infty_{n,m,q,\beta}} = \left\{((\mathbf{A}, \mathbf{y}), \mathbf{x}) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^n \times \mathbb{Z}_q^m : (\|\mathbf{x}\|_\infty \leq \beta) \wedge (\mathbf{x}^T \mathbf{A} = \mathbf{y} \bmod q)\right\}$$

Stern [40] proposed a protocol to solve the *syndrome decoding problem* (SPD). It is a problem from error correcting code theory where the goal is to find a *code vector* $\mathbf{x}$ of given *syndrome* $\mathbf{y}$ and given hamming weight – namely the number of bits set to 1. We can then informally see it as an ISIS problem with respect to hamming weight instead of a norm.

Definition 18 (Syndrome Decoding Problem)**.** Let r,n,w be integers, and let $(\mathbf{H},w,\mathbf{s})$ be a triple consisting of a *parity-check* matrix $\mathbf{H} \in \mathbb{F}_2^{n\times r}$, an integer $w \leq n$ and a vector $\mathbf{s} \in \mathbb{F}_2^r$.

Does there exist a vector $e \in \mathbb{F}_2^n$ of hamming weight $\leq w$ such that $\mathbf{e}^T\mathbf{H} = \mathbf{s}$?

To fit into this setting, we define specific sets of vectors. We set B_{3m} to be the set of vectors in $\{-1,0,1\}^{3m}$ such that there are exactly m coordinates of each value. Similarly we define the set B_{2m} to be the set of vectors in $\{0,1\}$ such that there are exactly m coordinates of each value. Once the initial vector belongs to one of those sets, we are allowed to apply the Stern protocol.

A first restriction is that in the Stern initial protocol, the hamming weight of the witness $\mathbf{x}$ is prescribed initially, which is not the case in the ISIS^∞ problem. To avoid it, the idea is to extend the vector $\mathbf{x}$ with $2m$ coordinates in such a way that the resulting vector $\mathbf{x}'$ belongs to B_{3m} defined as above. Then by adding $2m$ zero-lines to matrix $\mathbf{A}$ to produce $\mathbf{A}'$ we have $\mathbf{x}'^T\mathbf{A}' = \mathbf{y} \bmod q \iff \mathbf{x}^T\mathbf{A} = \mathbf{y} \bmod q$. Which means that if we can convince someone that $\mathbf{x}'$ is a valid witness for the syndrome decoding problem, we have that $\mathbf{x}$ is a valid witness for the relation $R_{\mathsf{ISIS}^\infty_{n,m,q,1}}$. This is the *extension* method.

Once this can be done, we can extend the protocol to reach any bound β and not just 1. To do this we use the binary decomposition technique. Namely let $\mathbf{x}$ be a vector with coordinates bounded by β. We decompose coordinate-wise $\mathbf{x}$ into $k = \lfloor \lg \beta \rfloor + 1$ vectors $\tilde{\mathbf{u}}_0,\ldots,\tilde{\mathbf{u}}_{k-1}$ such that $\mathbf{x} = \sum_{j=0}^{k-1} 2^j \cdot \tilde{\mathbf{u}}_j$. Next we apply the extension in the previous paragraph to have vectors $\mathbf{u}_j$ such that:

$$\left(\sum_{j=0}^{k-1} 2^j \cdot \mathbf{u}_j\right)^T \cdot \mathbf{A}' = \mathbf{y} \bmod q \iff \mathbf{x}^T\mathbf{A} = \mathbf{y} \bmod q$$

We then have a framework to turn k proofs for the syndrome decoding problem into a proof for $R_{\mathsf{ISIS}^\infty_{n,m,q,\beta}}$. We thus have the requested proof system.

D.3 Gentry-Peikert-Vaikuntanathan IBE

To prove that we know the underlying plaintext under a multibit GPV encryption, we will use the method described in [35].

Namely we notice that given $\mathbf{P} = \left(\begin{array}{c|cc} \mathbf{B} & \mathbf{I}_m & \\ \mathbf{G} & & \mathbf{I}_k \end{array}\right) \in \mathbb{Z}_q^{(m+k)\times(m+n+k)}$, $\mathbf{c} = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} \in \mathbb{Z}_q^{n+k}$ be a GPV ciphertext of a given message $\mathbf{b}$ and $\mathbf{e} = \begin{pmatrix} s \\ x \\ x_2 \end{pmatrix} \in \mathbb{Z}^{m+n+\ell}$ be the Gaussian noises used in the GPV encryption, we have:

$$\mathbf{P}\cdot\mathbf{e} + \begin{pmatrix} \mathbf{I}_m & \\ & \lfloor q/2 \rfloor \mathbf{I}_k \end{pmatrix} \cdot \begin{pmatrix} 0^m \\ \mathbf{b} \end{pmatrix} = \mathbf{c}$$

Which is the relation we want to prove. Like (9). The (10), (11) are simply a variant. The witness is $(\mathbf{e},\mathbf{b})$ and the publicly known parameter is $(\mathbf{P},\mathbf{c})$.

This can be achieved using the decomposition-extension framework we described above, as follows:

- To argue that $\mathbf{b} \in \{0,1\}^k$, we extend d to $d^* \in \mathsf{B}_{2k}$, then use a random permutation: if the permutation is in B_{2k}, that means that the initial vector was in B_{2k}
- To argue that $\mathbf{e} \in Z^{m+n+\ell}$, where $\|e\|_\infty \leq b$, we form the vectors $\mathbf{e}_1,\ldots,\mathbf{e}_p \in \mathsf{B}_{3(m+n+\ell)}$ using the decomposition-extension method, then use random permutations to show the membership of the $\mathbf{e}_j$'s.
- We then extend the matrices with 0 in order to have the following relations as in Appendix D.1. To prove the relation we then add random noises to mask the initial vectors.