Shorter Lattice-Based Group Signatures via “Almost Free” Encryption and Other Optimizations[⋆]

Vadim Lyubashevsky[1], Ngoc Khanh Nguyen[1,2], Maxime Plancon[1,2], and Gregor Seiler[1,2]

[1] IBM Research Europe, Zurich, Switzerland
[2] ETH Zurich, Switzerland

Abstract. We present an improved lattice-based group signature scheme whose parameter sizes and running times are independent of the group size. The signature length in our scheme is around 200KB, which is approximately a 3X reduction over the previously most compact such scheme, based on any quantum-safe assumption, of del Pino et al. (ACM CCS 2018). The improvement comes via several optimizations of some basic cryptographic components that make up group signature schemes, and we think that they will find other applications in privacy-based lattice cryptography.

Keywords: Lattice Cryptography, Group Signatures, Zero-Knowledge

1 Introduction

The eventual coming of quantum computers, combined with the ongoing shift to decentralization, makes designing efficient quantum-safe privacy-based primitives a highly pertinent problem. One of the more elementary privacy-preserving primitives is a group signature, and constructing such schemes has often been seen as an important stepping stone towards constructing more expressive primitives.

In a group signature scheme, the setup authority gives out individual signing keys s_i to users with identities m_i. User m_i can then utilize s_i to create a signature σ on a message μ of his choosing. There is also an entity called the Opener (or Group Manager) who is able to derive the identity m_i of the user who created σ. A basic group signature scheme has the following security properties:

1. Anonymity. The adversary who knows all the signing keys s_i cannot distinguish between signatures produced by user m_i or $m_{i'}$, for i, i' of the adversary's choosing.
2. Traceability. The adversary who possesses signing keys to all users in some set S, and the Opener's secret key, cannot create a valid signature that the Opener will decrypt to some identity $m_i \notin S$ or to $\perp$ (i.e. decryption will fail).

While there exist fairly efficient group signatures based on standard assumptions (e.g. [CS97]), most of the early work in trying to construct lattice-based group signatures were efficient only in an asymptotic sense with concrete signature sizes being around 50MB (e.g. [GKV10, LLNW16]). More recently, some concretely efficient schemes appeared that lowered the signature sizes to a little over 1MB (c.f. [BCN18]), and the scheme with the smallest signature size, in which the parameters and computational complexity of signing and verifying do not depend on the group size, was proposed by del Pino et al. [dPLS18] in 2018, where signature sizes are approximately 580KB.

Starting in 2019, the efficiency of lattice-based zero-knowledge proofs, which are important components of group signatures, has improved by several orders of magnitude (c.f. [YAZ$^+$19, BLS19, ESLL19, ESS$^+$19, ALS20, ENS20, LNS20]). The improvements have been dramatic-enough that one can now create sophisticated systems like lattice-based confidential transactions (i.e. a Monero-like

[⋆] This is the full version of the paper presented at ASIACRYPT 2021.

payment system based on the hardness of lattice problems) where the communication complexity of a transaction is under 30KB [EZS$^+$19, LNS21b].

Using these same techniques, [EZS$^+$19, ESZ21] improved on the efficiency of group signatures for the special case where the group size is not too large. While the signature size is smaller, the signing, verification, and opening times are linear in the number N of group members. When the group size is $N \approx 2^{10}$, the signature size is below 20KB and the scheme is reasonably efficient. The computational complexity of the scheme becomes prohibitive, however, as the group size approaches 2^{20} members. Independently, Beullens et al. [BDK$^+$21] proposed a lattice-based group signature construction which enjoys not only smaller signatures than [EZS$^+$19] for $N > 2^{21}$ but also sublinear verification time. However, the signing complexity is still linear in N.

It is interesting that despite the recent progress in zero-knowledge proofs, there haven't been any improvements in general group signature constructions whose complexities are independent of the group size. One reason for this lack of progress might be that the techniques used in [dPLS18] are quite different than what has been improved upon. For example, a key component in that scheme an ABB-like [ABB10] selectively-secure signature scheme that uses the Micciancio-Peikert trapdoor generation procedure [MP12], which has not been improved upon since it was first introduced. Furthermore, the utilized zero-knowledge proof in [dPLS18] requires proving equations of the form $\boldsymbol{A}\boldsymbol{s} = \boldsymbol{t} \pmod q$, where $\boldsymbol{s}$ has very large coefficients, on the order of $\sqrt{q}$. Most of the improvements in ZK proof constructions, on the other hand, only improved upon proofs of the above equation when $\boldsymbol{s}$ has small (e.g. $-1/0/1$) coefficients. And of course the state of the art of lattice-based encryption also hasn't changed since 2018.

In this work, we improve the group signature scheme in [dPLS18] by approximately a 3X factor in the signature size. Our construction follows the framework from [dPLS18] and the signature size improvement comes from moderate improvements to many parts of that protocol. Similarly as in [dPLS18], we fix an upper-bound on the number of users in the group ($\approx 2^{64}$) and thus the signature size, signing and verifying complexity do not depend on the actual group size. Since some of those parts are quite generic, we believe that our improvements could also find applications in other privacy-based protocols, including group signatures that satisfy stronger security notions (e.g. dynamic, corrupt setup authority, etc.). We will now give a high level overview of our signing algorithm and relate it to what was the state of the art in [dPLS18].

1.1 The Scheme of [dPLS18] and Our Improvements

The master public key of the setup authority consists of a random matrix $\boldsymbol{A} \in \mathcal{R}_q^{\alpha\times 2\alpha}$ and $\boldsymbol{B} = \boldsymbol{A}\boldsymbol{R} \in \mathcal{R}_q^{\alpha\times 3\alpha}$, where $\boldsymbol{R}$ is the master secret key and consists of polynomials in $\mathcal{R}_q$ with small coefficients, and a random polynomial vector $\boldsymbol{u} \in \mathcal{R}_q^{2\alpha}$. In our scheme, the ring $\mathcal{R}_q$ is fixed to be $\mathbb{Z}_q[X]/(X^{128}+1)$, for $q \approx 2^{64}$, and the security of the scheme is based on the Module-SIS / Module-LWE problems and varies with α. This is similar to the setup in [dPLS18], except there the security was based on Ring-LWE.

The secret key of a user with identity $m \in \mathcal{R}_q$ is a low-norm vector $\boldsymbol{s}$ satisfying

$$\left[\boldsymbol{A} \middle|\ \boldsymbol{B} + m\boldsymbol{G}\right] \boldsymbol{s} = \boldsymbol{u}, \tag{1}$$

where $\boldsymbol{G}$ is a "gadget matrix" which, together with the trapdoor matrix $\boldsymbol{R}$, is used to create such an $\boldsymbol{s}$ using the sampling algorithm of [MP12].

When creating a signature on a message μ, the signer needs to prove knowledge of $\boldsymbol{s}$ and m that satisfy this equation and to also to create an encryption to this same m under the public key

1. The prover $\mathcal{P}$ chooses a masking parameter $\boldsymbol{y}$ (from some specified distribution), creates $\boldsymbol{w} = \boldsymbol{A}'\boldsymbol{y}$ and sends $\boldsymbol{w}$ to the verifier
2. The verifier chooses a random challenge c
3. The prover computes $\boldsymbol{z} = c\boldsymbol{s} + \boldsymbol{y}$, performs rejection sampling to make sure $\boldsymbol{z}$ does not leak information about $\boldsymbol{s}$, and sends $\boldsymbol{z}$ to the verifier (or restarts).
4. The verifier accepts if $\|\boldsymbol{z}\|$ is small and $\boldsymbol{A}'\boldsymbol{z} = \boldsymbol{w} + c\boldsymbol{u}$.

Fig. 1. Zero Knowledge Proof of Knowledge of low-norm $\bar{\boldsymbol{s}}, \bar{c}$ that satisfy $\boldsymbol{A}'\bar{\boldsymbol{s}} = \bar{c}\boldsymbol{u}$, when the prover has knowledge of $\boldsymbol{s}$ satisfying $\boldsymbol{A}'\boldsymbol{s} = \boldsymbol{u}$.

of the opening authority.[3] As is often the case in lattice cryptography, it is more efficient to give a relaxed proof of (1) showing knowledge of an $\bar{\boldsymbol{s}}$ with a slightly larger norm, and a small $\bar{c}$ satisfying

$$\left[\boldsymbol{A} | \ \boldsymbol{B} + m\boldsymbol{G}\right] \bar{\boldsymbol{s}} = \bar{c}\boldsymbol{u}. \tag{2}$$

The above is very similar in form to the basic lattice-based ZK relaxed proof that one uses to construct Schnorr-like signature schemes (c.f. [Lyu09, Lyu12, DKL$^+$18]). The idea in those schemes is that a signer with a secret key $\boldsymbol{s}$ satisfying $\boldsymbol{A}'\boldsymbol{s} = \boldsymbol{u}$, where $\boldsymbol{A}'$ and $\boldsymbol{u}$ are public, can prove knowledge of $\bar{\boldsymbol{s}}$ and $\bar{c}$ satisfying $\boldsymbol{A}'\bar{\boldsymbol{s}} = \bar{c}\boldsymbol{u}$ as described in Figure 1. It's a proof of knowledge because by rewinding the prover at step (2), the verifier can create a second equality $\boldsymbol{A}'\boldsymbol{z}' = \boldsymbol{w} + c'\boldsymbol{u}$, and subtract to obtain $\boldsymbol{A}'\bar{\boldsymbol{z}} = \bar{c}\boldsymbol{u}$, where $\bar{\boldsymbol{z}} = \boldsymbol{z} - \boldsymbol{z}'$ and $\bar{c} = c - c'$.

One could hope to use the same approach for creating a proof for (2), but there is an important difference that doesn't allow a direct application of the same proof technique. In (2), the matrix on the left side is not public, as it contains the secret identity m. The verifier would therefore have no way to perform the verification in step (4) of the above procedure. The solution in [dPLS18] was to commit to the vectors $m\boldsymbol{G}$ using a BDLOP commitment [BDL$^+$18] and then replace the value $m\boldsymbol{G}$ with the commitment. Via some homomorphic properties of this commitment scheme, one can combine a zero-knowledge proof of the commitment with the zero-knowledge proof of (2) (with the modified left side) to conclude something similar to (2). The main downside of that approach is that the proof of knowledge for the commitment and (2) are both "relaxed", and therefore there is an additional $\bar{c}$ term that ends up multiplying into an extracted value and increasing the size of the extracted solution.

An improved proof of (2)**.** As part of our improved protocol, we propose a simpler and more efficient technique for proving (2). The verification in step (4) of Figure 1 requires all the involved elements to be known to the verifier. If they are not known (as $\boldsymbol{A}' := \left[\boldsymbol{A} | \ \boldsymbol{B} + m\boldsymbol{G}\right]$ will not be), then what we can instead do is to create a commitment to the part of $\boldsymbol{A}'$ that is unknown, and also a commitment to $\boldsymbol{w}$ (this is necessary to keep the unknown part of $\boldsymbol{A}'$ hidden, and so in the first step, the prover sends the commitment to $\boldsymbol{w}$ instead of $\boldsymbol{w}$ itself), and then instead of the verifier doing the verification check himself, the prover sends him a zero-knowledge proof that $\boldsymbol{A}'\boldsymbol{z} = \boldsymbol{w} + c\boldsymbol{u}$. If we use the BDLOP commitment scheme and the above equation is linear over the ring $\mathcal{R}_q$, then proving this relation does not add anything extra over just proving knowledge of the committed values.

[3] The message μ enters the signature as an input to a hash function that is used to convert the interactive proof into a non-interactive one via the Fiat-Shamir transform.

To go back to our example, if we create a BDLOP commitment of m and $\boldsymbol{w}$, then the equation $\left[\boldsymbol{A} | \ \boldsymbol{B} + m\boldsymbol{G}\right] \boldsymbol{z} = \boldsymbol{w} + c\boldsymbol{u}$ is indeed linear over $\mathcal{R}_q$ in the committed values m and $\boldsymbol{w}$. Thus sending this proof proves knowledge of $\bar{\boldsymbol{s}}$ and $\bar{c}$ satisfying (2) because one can do the extraction exactly in the same manner as for the protocol in Figure 1.

Proving Knowledge that the identity $\boldsymbol{m}$ is in a "Special" Set. It is important for the security of our scheme, which will eventually be shown to be as hard to break as forging the ABB signature scheme, that the identity m comes from a set $S \subset \mathcal{R}_q$ which satisfies that for $m \neq m' \in S$, $m - m'$ is invertible in $\mathcal{R}_q$, and that $|S|$ is small. The security reduction of the ABB signature scheme loses a factor $|S|$, one should ideally not have S be too large.[4] A good compromise is therefore having $|S| = q \approx 2^{64}$ and one can define it to be all polynomials in $\mathcal{R}_q$ of degree 0 (i.e. the integers modulo q). Now one needs to prove that m is indeed of this form. In [dPLS18], this proof was performed by showing that m is fixed under two specific automorphisms, and therefore must be an integer. But these "automorphism stability" proofs increased the size of the BDLOP opening proofs (unlike the linear relation proofs).

In our current construction, we instead use the recent advances in ZK proofs for proving multiplicative relations over $\mathcal{R}_q$ [ALS20], as well as linear relations over the NTT coefficients [ENS20], of polynomials committed using BDLOP commitments. The tools from [ALS20, ENS20] are quite powerful and the proof that a committed value is an integer follows quite easily (there may even be multiple equally good ways of doing it), but we give a sketch of one such approach anyway. If $m \in \mathcal{R}_q$ is an integer, then $\mathsf{NTT}(m)$ contains m in all the slots. In other words,

$$\mathsf{NTT}(m) = \begin{bmatrix} 1 & 2 & 4 & 8 & \dots \\ 1 & 2 & 4 & 8 & \dots \\ \dots & \dots & \dots & \dots & \dots \\ 1 & 2 & 4 & 8 & \dots \end{bmatrix} \cdot \mathsf{NTT}(m_{bin}),$$

where $m_{bin} \in \mathcal{R}_q$ is a polynomial all of whose NTT coefficients are 0/1. The idea is then to include a commitment to the polynomial m_{bin} into the BDLOP commitment that we already use for proving (2), and then the above relation can pe proved using the techniques from [ENS20]. To prove that m_{bin} has 0/1 NTT coefficients, we give a proof that $(m_{bin}) \cdot (1 - m_{bin}) = 0$ by using the multiplicative proof from [ALS20].[5]

Because we were already using a BDLOP commitment, committing to an extra $\mathcal{R}_q$ polynomial and doing the above two proofs only adds a few extra kilobytes to the entire proof system.

Encryption (and Proof) of $\boldsymbol{m}$ almost for free. Our final improvement relates to the encryption procedure. A group signature scheme requires the signer to encrypt his identity m under the opener's public key and give a zero-knowledge proof that the encryption is the same m that was used in the proof of (2). A significant saving in the size of our signature, as compared to [dPLS18], is that we show how the encryption and the proof of knowledge that the encryption is valid can already be mostly included in the commitment of m that we created when proving (2).

[4] While it's insecure for $S = \mathcal{R}_q$, it's unclear whether the size of S actually affects the real security of the scheme or it's just an artefact of the proof.

[5] Observe that we cannot use m_{bin} as our identity because the set of polynomials with 0/1 NTT coefficients is not closed under subtraction – hence this conversion is necessary.

The BDLOP commitment to a message $m \in \mathcal{R}_q$, and other things that need to be included (e.g. the m_{bin} described in the previous section, the $\boldsymbol{w}$ needed for the proof of (2), some "garbage terms" that need to be committed to as part of the proofs, etc.) is of the form

$$\begin{bmatrix} \boldsymbol{A}_0 \\ \boldsymbol{b}_1^T \\ \vdots \end{bmatrix} \cdot \boldsymbol{r} + \begin{bmatrix} \boldsymbol{0} \\ m \\ \vdots \end{bmatrix} = \begin{bmatrix} \boldsymbol{t}_0 \\ t_1 \\ \vdots \end{bmatrix}, \tag{3}$$

where $\boldsymbol{A}_0$ and $\boldsymbol{b}_1$ are random.[6] The important thing to note in the commitment scheme is that if we want to commit to an element in $\mathcal{R}_q$, then t_1 is just an element of the ring. On the other hand, the length of the vector $\boldsymbol{t}_0$ needs to be large for the security (binding property) of the commitment scheme. So if $\mathcal{R}_q$ is a 128-dimensional ring, the size of $\boldsymbol{t}_0$ could be 20 - 30 X larger than t_1.

Another thing to notice is that (3) looks very similar to a Regev-type encryption scheme [Reg09]. In particular, if $\boldsymbol{b}_1^T = \boldsymbol{s}_1^T\boldsymbol{A}_0 + \boldsymbol{e}_1^T$ (where $\boldsymbol{e}_1^T$ has small coefficients) then one could "decrypt" by computing $t_1 - \boldsymbol{s}_1^T\boldsymbol{t}_0 = \boldsymbol{e}_1^T\boldsymbol{r} + m$. So if $\boldsymbol{b}_1^T$ is part of the opener's public key, one can use the BDLOP commitment both for committing to m for the proof of (2) and for encrypting m! The main savings comes from the fact that we do not need to send two polynomial vectors of the form $\boldsymbol{t}_0$, one for the encryption and one for the commitment. If the opener uses $\boldsymbol{A}_0$ as part of his (Module)-LWE public key, then the same $\boldsymbol{t}_0$ can be used for both, which results in a substantial saving. Since the binding property of the commitment scheme only depends on $\boldsymbol{A}_0$, a malicious opener cannot do anything except possibly construct $\boldsymbol{b}_1$ such that it does not hide m – but a malicious opener can anyway always construct a malformed public key that does the same thing. So there is no disadvantage to combining the commitment and the encryption scheme into one.

We are, however, not quite yet done. One issue that needs to be taken care of is that from (3), the opener can recover $t_1 - \boldsymbol{s}_1^T\boldsymbol{t}_0 = \boldsymbol{e}_1^T\boldsymbol{r} + m$, where $\boldsymbol{e}_1^T\boldsymbol{r}$ has small coefficients; but this does not allow him to recover m because m is an arbitrary integer in $\mathbb{Z}_q$. In order for the opener to be able to recover m, we need to employ an encryption scheme implicit in Gentry et al. [GSW13] which allows for encryptions of arbitrary-size messages. In particular, in addition to encrypting m, the prover will also have to encrypt $\sqrt{q}m$ (it's really $\lfloor\sqrt{q}\rceil$, but we will omit the $\lfloor\cdot\rceil$ for the sake of readability) as $\boldsymbol{b}_2^T\boldsymbol{r} + \sqrt{q}m = t_2$, where $\boldsymbol{b}_2^T = \boldsymbol{s}_2^T\boldsymbol{A}_0 + \boldsymbol{e}_2^T$. Then to decrypt, the decryptor uses his secret keys $\boldsymbol{s}_1^T, \boldsymbol{s}_2^T$ as before, to obtain

$$\begin{aligned} u_1 &= \boldsymbol{e}_1^T\boldsymbol{r} + m = \epsilon_1 + m \pmod q \\ u_2 &= \boldsymbol{e}_2^T\boldsymbol{r} + \sqrt{q}m = \epsilon_2 + \sqrt{q}m \pmod q, \end{aligned}$$

and then compute $u_2 - \sqrt{q}u_1 = \epsilon_2 - \sqrt{q}\epsilon_1 \pmod q$. If the size of $\epsilon_1, \epsilon_2 < \sqrt{q}/4$, then no reduction modulo q takes place in the preceding equation. And furthermore, ϵ_1 and ϵ_2 can be easily recovered by computing the previous equation modulo $\sqrt{q}$. And then one can recover m. So in order to have the commitment scheme which commits to arbitrary-sized ring elements also be an encryption scheme, the prover just needs to create an additional commitment to $\sqrt{q}m$ (which is very cheap because it's just one ring element), and do a BDLOP linear proof over $\mathcal{R}_q$ that the commitments to m and $\sqrt{q}m$ are related by a factor of $\sqrt{q}$ (which does not add anything to the proof size).

[6] Sometimes to save on computation time, the vector $\boldsymbol{A}_0$ and $\boldsymbol{b}_1$ can contain some polynomials that are just 0 or 1 (see [BDL+18]), but in our case we will need them to be uniformly random.

1. The prover generates masking vector $\boldsymbol{y}$ and computes $\boldsymbol{w} = \left[\boldsymbol{A} \middle| \ \boldsymbol{B} + m\boldsymbol{G}\right]\boldsymbol{y}$
2. The prover creates a BDLOP commitment f to $(m, m\sqrt{q}, m_{bin}, \boldsymbol{w})$, and sends f to the verifier.
3. The verifier picks a random challenge c and sends it to the prover.
4. The prover computes $\boldsymbol{z} = c\boldsymbol{s} + \boldsymbol{y}$ and performs rejection sampling to make sure that the distribution of $\boldsymbol{z}$ is independent of $\boldsymbol{s}$.
5. The prover creates the following ZK proofs on the values committed in f:
   - BDLOP proof of knowledge of the committed values
   - BDLOP linear proof that $\left[\boldsymbol{A} \middle| \ \boldsymbol{B} + m\boldsymbol{G}\right]\boldsymbol{z} = \boldsymbol{w} + c\boldsymbol{u}$.
   - BDLOP proof that $\sqrt{q}$ times the first committed value equals the second
   - Multiplicative proof that $m_{bin} \cdot (m_{bin} - 1) = 0$
   - Linear proof that $\begin{bmatrix} 1 & 2 & 4 & 8 & \dots \\ 1 & 2 & 4 & 8 & \dots \\ \dots & \dots & \dots & \dots & \dots \\ 1 & 2 & 4 & 8 & \dots \end{bmatrix} \cdot \mathsf{NTT}(m_{bin}) = \mathsf{NTT}(m)$.
6. The prover sends $\boldsymbol{z}$ and all the proofs to the verifier.
7. The verifier checks that $\|\boldsymbol{z}\|$ is small and that all the proofs are valid.

Fig. 2. The interactive protocol allowing a prover with identity m and low-norm polynomial vector $\boldsymbol{s}$ satisfying (1) to prove knowledge of low-norm $\bar{\boldsymbol{s}}$ and $\bar{c}$ satisfying (2). Additionally, the BLDOP proof of knowledge of the committed values implies a proof of knowledge of $\bar{\boldsymbol{r}}$, $\bar{c}$ satisfying (4), which can be used by the opener to recover m. The commitments to some "garbage terms" and other extraneous terms that are required for the scheme to work are omitted from this high level description. To convert this interactive protocol to a signing algorithm for the group signature, one applies the Fiat-Shamir transform and puts the message μ to be signed as an input to the hash function.

There is still a second issue. When doing a proof of knowledge for the BDLOP commitment as in (3) (with the additional $\boldsymbol{b}_2^T$ line), the prover does not actually prove this equation. Instead, he gives a "relaxed" proof (analogously to (2)) showing the existence of a low-norm vector $\bar{\boldsymbol{r}}$ and polynomial $\bar{c}$ satisfying

$$\begin{bmatrix} \boldsymbol{A}_0 \\ \boldsymbol{b}_1^T \\ \boldsymbol{b}_2^T \\ \vdots \end{bmatrix} \cdot \bar{\boldsymbol{r}} + \bar{c} \begin{bmatrix} \boldsymbol{0} \\ m \\ \sqrt{q}m \\ \vdots \end{bmatrix} = \bar{c} \begin{bmatrix} \boldsymbol{t}_0 \\ t_1 \\ t_2 \\ \vdots \end{bmatrix}. \tag{4}$$

Because the opener does not know $\bar{c}$, he cannot perform decryption as above. He can, however, perform a decryption of the type described in [LN17] where decryption involves guessing an element c' from the challenge space and then trying to decrypt using it and the proof produced by the prover by constructing a $\bar{c} = c - c'$ and essentially testing whether (4) is satisfied. This is also the decryption algorithm that was used in the group signatures of [dPLS18] and [EZS$^+$19]. The encryption scheme used in [LN17] was the Regev scheme where the messages were small, but we prove that the same technique is also applicable in our case where the message is arbitrary in $\mathcal{R}_q$.

We summarize the high-level signing algorithm in Figure 2. The real algorithm described in Figure 5 includes the concrete "garbage terms" that one needs to include as part of the proof of all the parts we described, and also a modification to the public key that is necessary for the security proof to go through. Specifically, instead of the public key being $[\boldsymbol{A}| \ \boldsymbol{B} = \boldsymbol{AR}\ ]$, it is of the form $[\boldsymbol{A}| \ \boldsymbol{AR} \mid \boldsymbol{B}']$ where $\boldsymbol{B}'$ is a random matrix and serves no real purpose in the signing procedure. The reason for its inclusion is that proving security of the scheme requires doing game hops between the public key being $[\boldsymbol{A}| \ \boldsymbol{AR}]$ and $[\boldsymbol{A}| \ \boldsymbol{AR} + m^*\boldsymbol{G}]$ for an arbitrary message m. While these two public keys are indistinguishable based on the Module-LWE assumption, the game hops also require the extractor to be able to produce valid signatures – and so some trapdoor needs to always be present.

The only way that we know how to do such game hops is to embed a second trapdoor into $\boldsymbol{B}'$ so that the extractor can always sign even when he loses access to the trapdoor $\boldsymbol{AR}$. It's interesting to note that if the parameters were set such that $\boldsymbol{AR}$ were statistically-close to uniform, then we would not need to use a computational assumption and could simply replace $\boldsymbol{AR}$ with $\boldsymbol{AR} + m^*\boldsymbol{G}$. But imposing that $\boldsymbol{AR}$ is statistically-close to uniform would make the overall parameters significantly worse than just adding the useless $\boldsymbol{B}'$ to the public key (and thus also increasing the dimension of the vector $\boldsymbol{s}$ in (1)). If one chooses to remove this matrix $\boldsymbol{B}'$ from the public key, one could save approximately 15% in the size of the signature from the parameter computation in Section 4.2. Removing the need for such a $\boldsymbol{B}'$ in the security proof (without affecting parameters) is a very good open problem.

1.2 Reducing the Public Key Size by Using Multiple Rings

For optimal efficiency of the protocol in Figure 2, we would like to create commitments of elements in a small ring, as certain parts of the proof are linear in the ring size. Working over small rings, however, has a negative effect on the public key size of the group signature scheme. The matrix $\boldsymbol{B}$ comprising the public key contains a trapdoor, and therefore, unlike the $\boldsymbol{A}$ and the $\boldsymbol{A}_0$ in (4), it cannot be generated from a small seed. One therefore needs to store the entire matrix $\boldsymbol{B}$ as part of the public key. In our sample instantiation (Table 2), the matrix $\boldsymbol{B}$ consists of $\alpha \times 3\alpha$ d-dimensional polynomials. Since the modulus we're working with is $\approx 2^{64}$, $\alpha = 24$, and $d = 128$, storing this matrix requires $128 \cdot 3\alpha^2 \cdot 64$ bits, which is more than 1.7MB.

Since the security of the scheme is determined by the total dimension over $\mathbb{Z}$ of the matrix $\boldsymbol{B}$, which is αd, it would be more advantageous to work over a larger ring, while having a smaller α. For example, if we instead set $d = 1024$, and $\alpha = 3$, the total dimension over $\mathbb{Z}$ of the matrix remains the same, yet the cost of storing it goes down to 216KB. And if we wanted to increase security to have $\alpha d = 4096$, we could set $\alpha = 1$ and $d = 4096$, and end up needing under 100KB to represent $\boldsymbol{B}$. Then, if we allow ternary coefficiens for $\boldsymbol{R}$, the secret key size becomes at most 6KB. Having a larger d, though, would increase the signature size.[7] In short, we want d to be small in order for the proofs to be more compact, but we want d to be large in order to have a small public key.

It turns out that we can have the best of both worlds. That is, we can still use small (e.g. 128-degree) rings for the commitment scheme in (3), while using larger rings in equations that use the non-compressible public key (1). The interaction between the committed elements in (3) and the equation in (1) is through the BDLOP proof of $\left[\boldsymbol{A} \middle| \, \boldsymbol{B} + m\boldsymbol{G}\right] \boldsymbol{z} = \boldsymbol{w} + c\boldsymbol{u}$, where m and $\boldsymbol{w}$ are in committed form and all the other variables are public. If the smaller ring S is a sub-ring of the larger one R (e.g. $S = \mathbb{Z}[X]/(X^d + 1)$ and $R = \mathbb{Z}[X]/(X^{dk} + 1)$), then one can show that there is a ring homomorphism between R and S^k, for an appropriately-defined multiplication over S^k. In other words, whatever relation that we need to prove over R can be proved by showing that some corresponding relations over S hold true. Therefore we can use BDLOP commitments over S to prove relations over R at no extra cost. For simplicity, we describe our protocols in this paper entirely over the small ring S, and give details about how one can express relations over R in S in Section 2.8.

[7] In principle, d does not need to be a power-of-2, but then we could not work with the very convenient polynomial rings $\mathbb{Z}[X]/(X^d + 1)$. We think that the slight saving in the public key size is not worth the extra hassle of working aver different rings, and so we only consider power-of-2 d.

Public key	Secret key	Signature
96KB	6KB	203KB

Table 1. Public key, user secret key and signature sizes of our group signature scheme.

2 Preliminaries

2.1 Notation

Let q be an odd prime. We write $x \leftarrow S$ when $x \in S$ is sampled uniformly at random from the finite set S and similarly $x \leftarrow D$ when x is sampled according to the distribution D. For $a < b$ and $n \in \mathbb{N}$, we define $[a, b] := \{a, a+1 \ldots, b\}$ and $[n] := [1, n]$. Given two functions $f, g : \mathbb{N} \to [0, 1]$, we write $f(\mu) \approx g(\mu)$ if $|f(\mu) - g(\mu)| < \mu^{-\omega(1)}$. A function f is negligible if $f \approx 0$. We write $\mathsf{negl}(n)$ to denote an unspecified negligible function in n.

For a power of two d, denote $\mathcal{R}$ and $\mathcal{R}_q$ respectively to be the rings $\mathbb{Z}[X]/(X^d + 1)$ and $\mathbb{Z}_q[X]/(X^d + 1)$. Unless stated otherwise, lower-case letters denote elements in $\mathcal{R}$ or $\mathcal{R}_q$ and bold lower-case letters represent column vectors with coefficients in $\mathcal{R}$ or $\mathcal{R}_q$. We also write bold upper-case letters for matrices in $\mathcal{R}$ or $\mathcal{R}_q$.

For an element $w \in \mathbb{Z}_q$, we write $\|w\|_\infty$ to mean $|w \bmod^{\pm} q|$. Define the ℓ_∞ and ℓ_p norms for $w = w_0 + w_1 X + \ldots + w_{d-1} X^{d-1} \in \mathcal{R}$ as follows:

$$\|w\|_\infty = \max_j \|w_j\|_\infty, \quad \|w\|_p = \sqrt[p]{\|w_0\|_\infty^p + \ldots + \|w_{d-1}\|_\infty^p}.$$

If $w = (w_1, \ldots, w_m) \in \mathcal{R}^k$, then

$$\|\boldsymbol{w}\|_\infty = \max_j \|w_j\|_\infty, \quad \|\boldsymbol{w}\|_p = \sqrt[p]{\|w_1\|^p + \ldots + \|w_k\|^p}.$$

By default, we denote $\|\boldsymbol{w}\| := \|\boldsymbol{w}\|_2$.

2.2 Cyclotomic Rings

Suppose q splits into l prime ideals of degree d/l in $\mathcal{R}$. This means $X^d + 1 \equiv \varphi_1 \ldots \varphi_l \pmod{q}$ with irreducible polynomials φ_j of degree d/l modulo q. We assume that $\mathbb{Z}_q$ contains a primitive $2l$-th root of unity $\zeta \in \mathbb{Z}_q$ but no elements whose order is a higher power of two, i.e. $q - 1 \equiv 2l \pmod{4l}$. Therefore, we have

$$X^d + 1 \equiv \prod_{j \in \mathbb{Z}_l} \left(X^{\frac{d}{l}} - \zeta^{2j+1} \right) \pmod{q}. \tag{5}$$

Let $\mathcal{M}_q := \{p \in \mathbb{Z}_q[X] : \deg(p) < d/l\}$ be the $\mathbb{Z}_q$-module of polynomials of degree less than d/l. We define the Number Theoretic Transform (NTT) of a polynomial $p \in \mathcal{R}_q$ as follows:

$$\mathsf{NTT}(p) := \begin{bmatrix} \hat{p}_0 \\ \vdots \\ \hat{p}_{l-1} \end{bmatrix} \in \mathcal{M}_q^l \text{ where } \mathsf{NTT}(p)_j = \hat{p}_j = p \bmod (X^{\frac{d}{l}} - \zeta^{2j+1}).$$

Furthermore, we expand the definition of NTT to vectors of polynomials $\boldsymbol{p} \in \mathcal{R}_q^k$, where the NTT operation is applied to each coefficient of $\boldsymbol{p}$, resulting in a vector in $\mathcal{M}_q^{kl}$.

We also define the inverse NTT operation. Namely, for a vector $\vec{v} \in \mathcal{M}_q^l$, $\mathsf{NTT}^{-1}(\vec{v})$ is the polynomial $p \in \mathcal{R}_q$ such that $\mathsf{NTT}(p) = \vec{v}$.

Let $\vec{v} = (v_0, \ldots, v_{l-1}), \vec{w} = (w_0, \ldots, w_{l-1}) \in \mathcal{M}_q^l$. Then, we define the component-wise product $\vec{v} \circ \vec{w}$ to be the vector $\vec{u} = (u_0, \ldots, u_{l-1}) \in \mathcal{M}_q^l$ such that

$$u_j = v_j w_j \bmod (X^{\frac{d}{l}} - \zeta^{2j+1})$$

for $j \in \mathbb{Z}_l$. By definition, we have the following property of the inverse NTT operation:

$$\mathsf{NTT}^{-1}(\vec{v}) \cdot \mathsf{NTT}^{-1}(\vec{w}) = \mathsf{NTT}^{-1}(\vec{v} \circ \vec{w}).$$

Similarly, we define the *inner product* as in [LNS21b]:

$$\langle \vec{v}, \vec{w} \rangle = \sum_{j=0}^{l-1} \left(v_j w_j \bmod (X^{\frac{d}{l}} - \zeta^{2j+1}) \right).$$

We point out that this operation is not an inner product in the strictly mathematical sense (e.g. it is not linear). Nevertheless, it has a few properties which are characteristic for an inner product. For instance, given arbitrary vectors $\vec{x}, \vec{y}, \vec{z} \in \mathcal{M}_q^l$ and scalar $c \in \mathbb{Z}_q$ we have: $\langle \vec{x}, \vec{y} \rangle = \langle \vec{y}, \vec{x} \rangle$ (symmetry), $\langle \vec{x} + \vec{y}, \vec{z} \rangle = \langle \vec{x}, \vec{z} \rangle + \langle \vec{y}, \vec{z} \rangle$ (distributive law) and $\langle c\vec{x}, \vec{y} \rangle = c\langle \vec{x}, \vec{z} \rangle$. We also remark that the definition of $\langle \cdot, \cdot \rangle$ depends on the factors of $X^d + 1$ modulo q.

We generalise the newly introduced operations to work for vectors $\vec{v} = (\vec{v}_1, \ldots, \vec{v}_k)$ and $\vec{w} = (\vec{w}_1, \ldots, \vec{w}_k) \in \mathcal{M}_q^{kl}$ of length being a multiple of l in the usual way. In particular $\langle \vec{v}, \vec{w} \rangle = \sum_{i=1}^{k} \langle \vec{v}_i, \vec{w}_i \rangle$.

Eventually, for a matrix $A \in \mathcal{M}_q^{n \times kl}$ with rows $\vec{a}_1, \ldots, \vec{a}_n \in \mathcal{M}_q^{kl}$ and a vector $\vec{v} \in \mathcal{M}_q^{kl}$, we define the matrix-vector operation:

$$A\vec{v} = \begin{pmatrix} \langle \vec{a}_1, \vec{v} \rangle \\ \vdots \\ \langle \vec{a}_n, \vec{v} \rangle \end{pmatrix} \in \mathcal{M}_q^n.$$

In proving linear relations, we will need the following two lemmas.

Lemma 2.1 ([LNS21b]). *Let $n, k \in \mathbb{N}$. Then, for any $A \in \mathcal{M}_q^{nl \times kl}, \vec{v} \in \mathcal{M}_q^{nl}$ and $\vec{s} \in \mathbb{Z}_q^{kl}$ we have*

$$\langle A\vec{s}, \vec{v} \rangle = \langle \vec{s}, A^T \vec{v} \rangle.$$

Lemma 2.2 ([ENS20]). *Let $p = p_0 + p_1 X + \ldots + p_{d-1} X^{d-1} \in \mathcal{R}_q$. Then,*

$$\frac{1}{l} \sum_{i=0}^{l} \mathsf{NTT}(p)_i = \sum_{i=0}^{d/l-1} p_i X^i.$$

2.3 Challenge Space

Let $\mathcal{C} := \{-1, 0, 1\}^d \subset \mathcal{R}_q$ be the challenge set of ternary polynomials with coefficients $-1, 0, 1$. We define the following probability distribution $C : \mathcal{C} \to [0, 1]$. The coefficients of a challenge $c \leftarrow C$ are independently identically distributed with $\Pr(0) = 1/2$ and $\Pr(1) = \Pr(-1) = 1/4$. We write ω such that $\Pr_{c \leftarrow C}(\|c\|_1 \leqslant \omega) \leqslant 2^{-\lambda}$.

Consider the coefficients of the polynomial $c \bmod (X^{d/l} - \zeta^{2j+1})$ for $c \leftarrow C$. Then, all coefficients follow the same distribution over $\mathbb{Z}_q$. Let us write Y for the random variable over $\mathbb{Z}_q$ that follows this distribution. Attema et al. [ALS20] give an upper bound on the maximum probability of Y.

Lemma 2.3. *Let the random variable Y over $\mathbb{Z}_q$ be defined as above. Then for all $x \in \mathbb{Z}_q$,*

$$\Pr(Y = x) \leqslant \frac{1}{q} + \frac{2l}{q} \sum_{j=0}^{l-1} \prod_{i=0}^{l-1} \left| \frac{1}{2} + \frac{1}{2} \cos\left(2\pi(2j+1)y\zeta^i/q\right) \right|. \tag{6}$$

In particular, [ALS20, ENS20] computed that for $q \approx 2^{32}$, the maximum probability for each coefficient of $c \bmod X^{d/l} - \zeta^{2j+1}$ is around $2^{-31.4}$. In general, we will call this probability p.

An immediate consequence of Lemma 2.3 is that polynomial $c \leftarrow C$ is invertible in $\mathcal{R}_q$ with overwhelming probability as long as parameters q, d, l are selected so that $q^{-d/l}$ is negligible.

2.4 Module-SIS and Module-LWE Problems

Security of the [BDL+18] commitment scheme used in our protocols relies on the well-known computational lattice problems, namely Module-LWE (MLWE) and Module-SIS (MSIS) [LS15]. Both problems are defined over $\mathcal{R}_q$.

Definition 2.4 ($\mathsf{MSIS}_{\kappa,m,B}$)**.** *Given $\boldsymbol{A} \leftarrow \mathcal{R}_q^{\kappa\times m}$, the* Module-SIS *problem with parameters $\kappa, m > 0$ and $0 < B < q$ asks to find $\boldsymbol{z} \in \mathcal{R}_q^m$ such that $\boldsymbol{Az} = \boldsymbol{0}$ over $\mathcal{R}_q$ and $0 < \|\boldsymbol{z}\| \leqslant B$. An algorithm Adv is said to have advantage ϵ in solving $\mathsf{MSIS}\kappa, m, B$ if*

$$\Pr\left[0 < \|\boldsymbol{z}\| \leqslant B \ \wedge\ \boldsymbol{Az} = \boldsymbol{0} \,\middle|\, \boldsymbol{A} \leftarrow \mathcal{R}_q^{\kappa\times m};\ \boldsymbol{z} \leftarrow \mathsf{Adv}(\boldsymbol{A})\right] \geqslant \epsilon.$$

Definition 2.5 ($\mathsf{MLWE}_{m,\lambda,\chi}$)**.** *The* Module-LWE *problem with parameters $m, \lambda > 0$ and an error distribution χ over $\mathcal{R}$ asks the adversary Adv to distinguish between the following two cases: 1) $(\boldsymbol{A}, \boldsymbol{As} + \boldsymbol{e})$ for $\boldsymbol{A} \leftarrow \mathcal{R}_q^{m\times\lambda}$, a secret vector $\boldsymbol{s} \leftarrow \chi^\lambda$ and error vector $\boldsymbol{e} \leftarrow \chi^m$, and 2) $(\boldsymbol{A}, \boldsymbol{b}) \leftarrow \mathcal{R}_q^{m\times\lambda} \times \mathcal{R}_q^m$. Then, Adv is said to have advantage ϵ in solving $\mathsf{MLWE}_{m,\lambda,\chi}$ if*

$$\begin{aligned} \Big|\Pr\left[b = 1 \,\middle|\, \boldsymbol{A} \leftarrow \mathcal{R}_q^{m\times\lambda};\ \boldsymbol{s} \leftarrow \chi^\lambda;\ \boldsymbol{e} \leftarrow \chi^m;\ b \leftarrow \mathsf{Adv}(\boldsymbol{A}, \boldsymbol{As} + \boldsymbol{e})\right] \\ - \Pr\left[b = 1 \,\middle|\, \boldsymbol{A} \leftarrow \mathcal{R}_q^{m\times\lambda};\ \boldsymbol{b} \leftarrow \mathcal{R}_q^m;\ b \leftarrow \mathsf{Adv}(\boldsymbol{A}, \boldsymbol{b})\right]\Big| \geqslant \epsilon. \end{aligned} \tag{7}$$

2.5 Probability Distributions

In this paper we sample the coefficients of the random polynomials in the commitment scheme using the distribution χ on $\{-1, 0, 1\}$ where ±1 both have probability $5/16$ and 0 has probability $6/16$ identically as in e.g. [BLS19, ALS20, ENS20]. We also write S_μ the uniform distribution over the set $\{x \in \mathcal{R}_q \mid \|x\|_\infty \leqslant \mu\}$.

Discrete Gaussian distribution. We now define the discrete Gaussian distribution used for the rejection sampling.

Definition 2.6. *The* discrete Gaussian distribution *on $\mathbb{Z}^\ell$ centered around $\vec{v} \in \mathbb{Z}^\ell$ with standard deviation $\mathfrak{s} > 0$ is given by*

$$D^\ell_{\vec{v},\mathfrak{s}}(\vec{z}) = \frac{e^{-\|\vec{z}-\vec{v}\|^2/2\mathfrak{s}^2}}{\sum_{\vec{z}'\in\mathbb{Z}^\ell} e^{-\|\vec{z}'\|^2/2\mathfrak{s}^2}}.$$

When it is centered around $\boldsymbol{0} \in \mathbb{Z}^\ell$ we write $D^\ell_{\mathfrak{s}} = D^\ell_{\vec{0},\mathfrak{s}}$

We will use the following tail bound, which follows from [Ban93, Lemma 1.5(i)].

Lemma 2.7. *Let $\boldsymbol{z} \leftarrow D_{\mathfrak{s}}^{\ell d}$. Then*

$$\Pr\left[\|\boldsymbol{z}\|_2 \leqslant \mathfrak{s}\sqrt{2\ell d}\right] \geqslant 1 - 2^{-\log(e/2)\ell d/4}.$$

2.6 Rejection Sampling

In lattice-based zero-knowledge proofs, the prover will want to output a vector $\boldsymbol{z}$ whose distribution should be independent of a secret randomness vector $\boldsymbol{r}$, so that $\boldsymbol{z}$ cannot be used to gain any information on the prover's secret. During the protocol, the prover computes $\boldsymbol{z} = \boldsymbol{y} + c\boldsymbol{r}$ where $\boldsymbol{r}$ is the randomness used to commit to the prover's secret, $c \leftarrow C$ is a challenge polynomial, and $\boldsymbol{y}$ is a "masking" vector. In order to remove the dependency of $\boldsymbol{z}$ on $\boldsymbol{r}$, one applies *rejection sampling* [Lyu12].

Lemma 2.8 (Rejection Sampling). *Let $V \subseteq \mathcal{R}^\ell$ be a set of polynomials with norm at most T and $\rho\colon V \to [0,1]$ be a probability distribution. Now, sample $\boldsymbol{v} \leftarrow \rho$ and $\boldsymbol{y} \leftarrow D_{\mathfrak{s}}^{\ell d}$, set $\boldsymbol{z} = \boldsymbol{y}+\boldsymbol{v}$, and run $b \leftarrow \mathsf{Rej}_0(\boldsymbol{z}, \boldsymbol{v}, \mathfrak{s})$ as defined in Fig. 3. Then, the probability that $b = 0$ is at least $(1-2^{-100})/M$ and the distribution of $(\boldsymbol{v}, \boldsymbol{z})$, conditioned on $b = 0$, is within statistical distance of $2^{-100}/M$ of the product distribution $\rho \times D_{\mathfrak{s}}^{\ell d}$.*

```
Rej_0($\vec{z}, \vec{v}, \mathfrak{s}$)
01 $u \leftarrow [0,1)$
02 If $u > \frac{1}{M} \cdot \exp\left(\frac{-2\langle \vec{z},\vec{v}\rangle + \|\vec{v}\|^2}{2\mathfrak{s}^2}\right)$
03     return 1
04 Else
05     return 0

Rej_1($\vec{z}, \vec{v}, \mathfrak{s}$)
01 If $\langle \vec{z},\vec{v}\rangle < 0$
02     return 1
03 $u \leftarrow [0,1)$
04 If $u > \frac{1}{M} \cdot \exp\left(\frac{-2\langle \vec{z},\vec{v}\rangle + \|\vec{v}\|^2}{2\mathfrak{s}^2}\right)$
05     return 1
06 Else
07     return 0
```

Fig. 3. Two rejection sampling algorithms: the one used generally in previous works [Lyu12] (left) and the one proposed recently in [LNS21a] (right).

We recall how parameters $\mathfrak{s}$ and M in Lemma 2.8 are selected. Concretely, the repetition rate M is chosen to be an upper-bound on:

$$\frac{D_{\mathfrak{s}}^{\ell d}(\boldsymbol{z})}{D_{\boldsymbol{v},\mathfrak{s}}^{\ell d}(\boldsymbol{z})} = \exp\left(\frac{-2\langle \boldsymbol{z}, \boldsymbol{v}\rangle + \|\boldsymbol{v}\|^2}{2\mathfrak{s}^2}\right) \leqslant \exp\left(\frac{24\mathfrak{s}\|\boldsymbol{v}\| + \|\boldsymbol{v}\|^2}{2\mathfrak{s}^2}\right) = M^8. \tag{8}$$

For the inequality we used the tail bound which says that with probability at least $1 - 2^{100}$ we have $|\langle \boldsymbol{z}, \boldsymbol{v}\rangle| < 12\mathfrak{s}\|\boldsymbol{v}\|$ for $\boldsymbol{z} \leftarrow D_{\mathfrak{s}}^{\ell d}$ [Ban93, Lyu12]. Hence, by setting $\mathfrak{s} = 11\|\boldsymbol{v}\|$ we obtain $M \approx 3$.

[8] Here, the inner product is over $\mathbb{Z}$, i.e. $\langle \boldsymbol{z}, \boldsymbol{v}\rangle = \langle \vec{z}, \vec{v}\rangle$ where vectors $\vec{z}, \vec{v}$ are polynomial coefficients of $\boldsymbol{z}$ and $\boldsymbol{v}$ respectively.

2.7 BDLOP Commitment Scheme

We recall the BDLOP commitment scheme from [BDL$^+$18]. Suppose that we want to commit to a message vector $\boldsymbol{m} = (m_1, \ldots, m_n) \in \mathcal{R}_q^n$ for $n \geqslant 1$ and that module ranks of κ and λ are required for MSIS and MLWE security, respectively. Then, in the key generation, a matrix $\boldsymbol{A}_0 \leftarrow \mathcal{R}_q^{\kappa \times (\kappa+\lambda+n)}$ and vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n \leftarrow \mathcal{R}_q^{\kappa+\lambda+n}$ are generated and output as public parameters. Note that one could choose to generate $\boldsymbol{A}_0, \boldsymbol{a}_1, \ldots, \boldsymbol{a}_n$ in a more structured way as in [BDL$^+$18] since it saves some computation. However, for readability, we write the commitment matrices in the "Knapsack" form as above. In our case, the hiding property of the commitment scheme is established via the duality between the Knapsack and MLWE problems. We refer to [EZS$^+$19, Appendix C] for a more detailed discussion.

To commit to the message $\boldsymbol{m}$, we first sample $\boldsymbol{r} \leftarrow \chi^{d\cdot(\kappa+\lambda+n)}$. Now, there are two parts of the commitment scheme: the binding part and the message encoding part. In particular, we compute

$$\begin{aligned} \boldsymbol{t}_0 &= \boldsymbol{A}_0 \boldsymbol{r} \bmod q, \\ t_i &= \boldsymbol{a}_i^T \boldsymbol{r} + m_i \bmod q, \end{aligned}$$

for $i \in [n]$, where $\boldsymbol{t}_0$ forms the binding part and each t_i encodes a message polynomial m_i. In this paper, when we write that we compute a BDLOP commitment to a vector $\vec{m} = (\vec{m}_1, \ldots, \vec{m}_n) \in \mathcal{M}_q^{nl}$, we mean that we commit to the vector of polynomials $\boldsymbol{m} = (\mathsf{NTT}^{-1}(\vec{m}_1), \ldots, \mathsf{NTT}^{-1}(\vec{m}_n)) \in \mathcal{R}_q^n$ as above.

Next, we define the notion of a weak opening of the commitment [ALS20].

Definition 2.9. *A* weak opening *for the commitment $\boldsymbol{t} = \boldsymbol{t}_0 \parallel t_1 \parallel \cdots \parallel t_n$ consists of a polynomial $\bar{c} \in \mathcal{R}_q$, a randomness vector $\boldsymbol{r}^*$ over $\mathcal{R}_q$ and messages $m_1^*, \ldots, m_n^* \in \mathcal{R}_q$ such that*

$$\begin{aligned} &\|\bar{c}\|_1 \leqslant 2d \text{ and } \bar{c} \text{ is invertible over } \mathcal{R}_q \\ &\|\bar{c}\boldsymbol{r}^*\|_2 \leqslant 2\beta, \\ &\boldsymbol{A}_0 \boldsymbol{r}^* = \boldsymbol{t}_0, \\ &\boldsymbol{a}_i^T \boldsymbol{r}^* + m_i^* = t_i \text{ for } i \in [n]. \end{aligned}$$

Attema et al. [ALS20] show that the commitment scheme is still binding with respect to weak openings if $\mathsf{MSIS}\kappa, 8d\beta$ is hard.

2.8 Working Over Subrings

Let $R = \mathbb{Z}[X]/(X^{dk}+1)$ and $S = \mathbb{Z}[X]/(X^d+1)$, where both d and k are powers of two. The goal of this section is to show how one can perform operations in R by first mapping the elements to S^k, performing operations over S, and then (if needed) mapping back to R. We mention that the results in this section are basic algebra, and we are merely stating them for completeness.

The operations that we are interested in is addition and multiplication. Notice that addition between two polynomials is exactly the same whether they are in R, S, or $\mathbb{Z}[X]$, and so we only focus on multiplication, which is defined differently for each ring. We will denote the multiplication operations by $\underset{R}{\otimes}, \underset{S}{\otimes}$, and $\otimes$ for multiplication in R, S, and $\mathbb{Z}[X]$, respectively. In what follows, rather than formally working with quotient rings R and S, we define the elements of R (resp. S) do simply be all polynomials in $\mathbb{Z}[X]$ of degree less than dk (resp. d). And thus the multiplication $\underset{R}{\otimes}$ (resp.

$\underset{S}{\otimes}$) is a multiplication of two polynomials in $\mathbb{Z}[X]$ followed by a reduction modulo $X^{dk}+1$ (resp. X^d+1) to obtain a polynomial of degree at most dk (resp. d).

The concrete relationship between S and R hinges on the way that one chooses to represent elements in R by a combination of elements in S. There are, of course, many ways to do so, but a particularly useful one is writing any element of $r \in R$ as

$$r = \sum_{i=0}^{k-1} s_i(X^k) \underset{R}{\otimes} X^i, \tag{9}$$

where $s_i \in S$. The algebraic perspective (which is not necessary in the sequel) is that $s_i(X^k)$ is an embedding of S into R and the above equation shows how to write any element in R as a combination of elements in S with basis $1, X, \ldots, X^{k-1}$.

The key property of the embedding $s(X^k)$ that we need is that it is a homomorphism.

Lemma 2.10. *For elements $s, \bar{s} \in S$,*

$$s(X^k) \underset{R}{\otimes} \bar{s}(X^k) = (s \underset{S}{\otimes} \bar{s})(X^k)$$

Proof. We write out the multiplication $(s \underset{S}{\otimes} \bar{s})$ over $\mathbb{Z}[X]$ as $(s \underset{S}{\otimes} \bar{s}) = s \otimes \bar{s} + g \otimes (X^d+1)$, where g is some polynomial in $\mathbb{Z}[X]$. Then

$$\begin{aligned}(s \underset{S}{\otimes} \bar{s})(X^k) &= (s \otimes \bar{s} + g \otimes (X^d+1))(X^k) \\ &= (s(X^k) \otimes \bar{s}(X^k)) + g(X^k) \otimes (X^{kd}+1) \\ &= s(X^k) \underset{R}{\otimes} \bar{s}(X^k),\end{aligned}$$

where the last equality holds because $(X^{kd}+1) = 0$ in R. □

We now define a bijection

$$\begin{aligned}\sigma : & R \leftrightarrow S^k \\ & r \leftrightarrow (s_0, s_1, \ldots, s_{k-1})\end{aligned}$$

where the $s_i \in S$ are from (9). And we will finally show that multiplication between two elements $r, \bar{r} \in R$ can be performed by mapping them to S^k and then performing additions and multiplications in the ring S. In other words, the function σ is a homomorphism.

Lemma 2.11. *Let $r = \sum_{i=0}^{k-1} s_i(X^k) \underset{R}{\otimes} X^i$ and $\bar{r} = \sum_{j=0}^{k-1} \bar{s}_j(X^k) \underset{R}{\otimes} X^j$, where $s_i, \bar{s}_j \in S$. Thus $\sigma(r) = (s_0, s_1, \ldots, s_{k-1}) \in S^k$ and $\sigma(\bar{r}) = (\bar{s}_0, \bar{s}_1, \ldots, \bar{s}_{k-1}) \in S^k$.*

Define the operation $\underset{S^k}{\otimes}$ between two elements in S^k as

$$(s_0, \ldots, s_{k-1}) \underset{S^k}{\otimes} (\bar{s}_0, \ldots, \bar{s}_{k-1}) = (t_0, \ldots, t_{k-1})$$

where for all $0 \leqslant \ell < k$,

$$t_\ell = \sum_{\substack{0 \leqslant i,j < k \\ i+j \equiv \ell \bmod k}} s_i \underset{S}{\otimes} \bar{s}_j \underset{S}{\otimes} X^{\lfloor \frac{i+j}{k} \rfloor}.$$

Then $\sigma(r \underset{R}{\otimes} \bar{r}) = \sigma(r) \underset{S^k}{\otimes} \sigma(\bar{r})$.

Proof.

$$\begin{aligned}
r \underset{R}{\otimes} \bar{r} &= \left(\sum_{i=0}^{k-1} s_i(X^k) \underset{R}{\otimes} X^i\right) \underset{R}{\otimes} \left(\sum_{j=0}^{k-1} \bar{s}_j(X^k) \underset{R}{\otimes} X^j\right) \\
&= \sum_{0\leqslant i,j<k} s_i(X^k) \underset{R}{\otimes} \bar{s}_j(X^k) \underset{R}{\otimes} X^{i+j} \\
&= \sum_{\substack{0\leqslant i,j<k \\ i+j<k}} s_i(X^k) \underset{R}{\otimes} \bar{s}_j(X^k) \underset{R}{\otimes} X^{i+j} \\
&\qquad\qquad + \sum_{\substack{0\leqslant i,j<k \\ i+j\geqslant k}} s_i(X^k) \underset{R}{\otimes} \bar{s}_j(X^k) \underset{R}{\otimes} X^k \underset{R}{\otimes} X^{i+j-k} \\
&= \sum_{\substack{0\leqslant i,j<k \\ i+j<k}} (s_i \underset{S}{\otimes} \bar{s}_j)(X^k) \underset{R}{\otimes} X^{i+j} + \sum_{\substack{0\leqslant i,j<k \\ i+j\geqslant k}} (s_i \underset{S}{\otimes} \bar{s}_j \underset{S}{\otimes} X)(X^k) \underset{R}{\otimes} X^{i+j-k} \qquad (10) \\
&= \sum_{0\leqslant i,j<k} \left(s_i \underset{S}{\otimes} \bar{s}_j \underset{S}{\otimes} X^{\lfloor \frac{i+j}{k} \rfloor}\right)(X^k) \underset{R}{\otimes} X^{(i+j)-\lfloor \frac{i+j}{k} \rfloor k}, \qquad (11)
\end{aligned}$$

where (10) uses Lemma 2.10. The claim in the lemma follows by noticing that the exponent $(i + j) - \left\lfloor \frac{i+j}{k} \right\rfloor k$ in (11) will be equal to ℓ exactly when $i + j \equiv \ell \pmod{k}$. □

Notice that the multiplication operation over $\underset{S^k}{\otimes}$ in the Lemma only uses multiplication over S. Therefore performing multiplication in R can be done by performing multiplications in S, which is what we wanted to show in this section.

We should mention that in our application, we only have a very special case of multiplication involving committed elements. That is, we have a multiplication between a public element in R and m, whose representation in R is simply $m(X^k)$. So the $\bar{r}$ in Lemma 2.11 is simply $(m, 0, \ldots, 0)$. And therefore

$$r \underset{R}{\otimes} m(X^k) = \left(\sum_{i=0}^{k-1} s_i(X^k) \underset{R}{\otimes} X^i\right) \underset{R}{\otimes} m(X^k) = \sum_{i=0}^{k-1} (s_i \underset{S}{\otimes} m)(X^k) \underset{R}{\otimes} X^i,$$

and so the $t_\ell = s_\ell \underset{S}{\otimes} m$.

As mentioned in Section 1.2, we will not formally describe our construction with different rings S and R, but simply stick with the version using the smaller ring S throughout.

3 The group signature

A group signature is composed of four algorithms. The first one, which we write KeyGen is run by the group manager and is described in Figure 4. In the end of this algorithm, the group manager generated a group public key, his own group manager secret and secret keys for all group members. The second one is the signature. The group member of identity m was given his secret key, which he will prove knowledge of (among other statements) in the signature. The signature is a non-interactive version of the zero-knowledge proof π described on Figure 5. Third, the verification is simply the verification of π. Finally, the last algorithm GSdec described in Algorithm 1 allows the group manager to reveal the identity at the origin of a signature. We will write $\sqrt{q}$ (respectively $\sqrt[3]{q}$) the integer $\lfloor\sqrt{q}\rceil$ (respectively $\lfloor\sqrt[3]{q}\rceil$) for the sake of readability.

3.1 All-in-one interactive zero-knowledge proof

In this subsection, we introduce a single zero-knowledge proof that encompasses all the proofs needed for our group signature scheme. From a high level, π proves the following statements

1. Knowledge of an identity m
2. Knowledge of the secret key of m
3. The decryption of the identity of the prover is m.

Intuitively, these three statements are required to capture the security notions of a group signature.

Each of these statements is proven by gathering more elementary zero-knowledge proofs from [LNS20, ALS20, BDL$^+$18]. For completeness, we present an overview of the techniques in A.1. More specifically, we take $\mathcal{I}$ the set of identities to be $\{0, 1, \ldots, 2^d - 1\} \subset \mathcal{R}_q$ the set of degree zero polynomials of $\mathcal{R}_q$ i.e $\mathbb{Z}_q$; such that the binary representation of m also fits in length as an element of $\mathcal{R}_q$. The prover will commit to m, but also to $m_{\text{bin}} = \mathsf{NTT}^{-1}(\mathsf{binary}(m))$ the inverse NTT of the binary representation of m. This way, we need to prove two things : 1) m_{bin}'s NTT is binary 2) we have the linear relation

$$\mathbf{Q}\ \mathsf{NTT}(m_{\text{bin}}) = \mathsf{NTT}(m), \text{ where } \mathbf{Q} = \begin{bmatrix} 1\ 2\ 4 \ldots & 2^{d-1} \\ 1\ 2\ 4 \ldots & 2^{d-1} \\ \vdots\ \vdots\ \vdots & \vdots \\ 1\ 2\ 4 \ldots & 2^{d-1} \end{bmatrix}.$$

For 1), this proof is done using the product proof from [ALS20]. For 2), we use the so called unstructured linear proof from [LNS20]. Notice that since all the NTT coefficients of m are proven to be equal, m has to be an integer. On top of that, since its binary representation has length d, we indeed prove that $m \in \mathcal{I}$.

Proving knowledge of the short $\mathbf{s}_1^m, \mathbf{s}_2^m, \mathbf{s}_3^m$ is done in two steps. The relation that these secret vectors verify is $\mathbf{A}\mathbf{s}_1^m + (\mathbf{B} + m\mathbf{G})\mathbf{s}_2^m + \mathbf{B}'\mathbf{s}_3^m$, where the identity m is multiplied with the so called *gadget matrix* :

$$\mathbf{G} = \mathbf{I}_\alpha \otimes [1\ \sqrt[3]{q}\ \sqrt[3]{q}^2] = \begin{bmatrix} 1\ \sqrt[3]{q}\ \sqrt[3]{q}^2 & & \\ & 1\ \sqrt[3]{q}\ \sqrt[3]{q}^2 & \\ & & \ddots \end{bmatrix}$$

The matrix $[\mathbf{A}|\mathbf{B}+m\mathbf{G}|\mathbf{B}']$ depends on the committed identity m and we can therefore not directly use the linear proof from [BDL$^+$18]. To circumvent this problem, instead of sending some $\mathbf{w} = \mathbf{A}\mathbf{y}_1 + (\mathbf{B} + m\mathbf{G})\mathbf{y}_2 + \mathbf{B}'\mathbf{y}_3$ as in the BDLOP linear proof, we commit to this $\mathbf{w}$ and give a BDLOP linear proof that $\mathbf{A}\mathbf{z}_1 + (\mathbf{B} + m\mathbf{G})\mathbf{z}_2 + \mathbf{B}'\mathbf{z}_3 = \mathbf{w} + c\mathbf{u}$. This statement is indeed linear in the two committed values m and $\mathbf{w}$.

The encryption of the identity m is part of the commitments. In a nutshell, the group manager plants his decryption key in the public commitment matrix during the key generation. This way, it allows the commitment to m by the prover to also be (part of) a ciphertext for the group manager to decrypt. The encryption involves two *commitments*, one to m and one to $\sqrt{q}m$. To prove that the ciphertext is valid[9] reduces to proving the knowledge of the short randomness $\mathbf{r}$ in the commitment scheme and the linear relation between the committed m and $\sqrt{q}m$. The latter proofs almost come for free : the opening proof is anyway necessary for the other proofs, and the linear proof is very cheap since these committed values are polynomials from $\mathcal{R}_q$.

Fig. 4. KeyGen() :

Decryption secret keys

$\mathbf{h}_1, \mathbf{h}_2 \leftarrow S_\mu^\kappa,\ \mathbf{e}_1, \mathbf{e}_2 \leftarrow S_\mu^{\kappa+\lambda+\alpha+5}$

Commitment public parameters

$(\mathbf{A}_0, \mathbf{A}_1, \mathbf{a}_2, \mathbf{a}_3, \mathbf{a}_4) \leftarrow \mathcal{R}_q^{\kappa\times(\kappa+\lambda+\alpha+5)} \times \mathcal{R}_q^{\alpha\times(\kappa+\lambda+\alpha+5)} \times (\mathcal{R}_q^{\kappa+\lambda+\alpha+5})^3$

$\mathbf{t}_1 = \mathbf{A}_0^T\mathbf{h}_1 + \mathbf{e}_1,\ \mathbf{t}_2 = \mathbf{A}_0^T\mathbf{h}_2 + \mathbf{e}_2$

Group signature group manager key and public parameters

$\mathbf{R} \leftarrow \{x \in \mathcal{R}_q^{\alpha\times\alpha} \mid \|x\|_\infty \leqslant 1\}^{2\times 3}$

$\mathbf{A} \leftarrow \mathcal{R}_q^{\alpha\times 2\alpha}, \mathbf{B} = \mathbf{AR}, \mathbf{B}' \leftarrow \mathcal{R}_q^{\alpha\times 3\alpha}$

$(\mathbf{s}_1^{\mathsf{gm}}, \mathbf{s}_2^{\mathsf{gm}}, \mathbf{s}_3^{\mathsf{gm}}) \leftarrow D_\sigma^{2d\alpha} \times D_\sigma^{3d\alpha} \times D_\sigma^{3d\alpha}$

$$\mathbf{u} = \begin{bmatrix}\mathbf{A} | \ \mathbf{B}\ | \mathbf{B}'\end{bmatrix}\begin{bmatrix}\mathbf{s}_1^{\mathsf{gm}} \\ \mathbf{s}_2^{\mathsf{gm}} \\ \mathbf{s}_3^{\mathsf{gm}}\end{bmatrix}$$

Group members secret keys

$\forall m \in \mathcal{I}$, use GPV trapdoor to sample $(\mathbf{s}_1^m, \mathbf{s}_2^m, \mathbf{s}_3^m)$ such that

$$\begin{bmatrix}\mathbf{A} | \ \mathbf{B} + m\mathbf{G}\ | \mathbf{B}'\end{bmatrix}\begin{bmatrix}\mathbf{s}_1^m \\ \mathbf{s}_2^m \\ \mathbf{s}_3^m\end{bmatrix} = \mathbf{u}$$

Theorem 3.1. *The interactive proof π from Figure 5 is complete, sound and zero-knowledge.*

More precisely, if the prover follows Figure 5 and does not abort, an honest verifier will output 1 *with overwhelming probability.*

There exists a simulator $\mathcal{S}$ that without access to secret information outputs a distribution that is, under the MLWE *assumption for parameters $(\kappa, \kappa+\lambda+\alpha+5, S_\mu)$ and $(\kappa+\alpha+5, \lambda, \chi)$, indistinguishable from the actual interaction.*

Let $B_2 \geqslant 8\omega^2\sigma'\sqrt{2(\kappa+\lambda+\alpha+5)d}$. If ϵ is the success probability of the prover and T its runtime, then there exists an extractor $\mathcal{E}$ that with rewindable blackbox access to this prover finds with probability $\geqslant 1/8$, in time $O(T/\epsilon)$, either a solution to $\mathsf{MSIS}_{\kappa,\kappa+\lambda+\alpha+5,B_2}$ or $\bar{m} \in \mathcal{I}$, $\bar{\mathbf{s}}_1, \bar{\mathbf{s}}_2, \bar{\mathbf{s}}_3$ of norms lower than respectively $4\sigma\sqrt{\alpha d}$, $2\sigma\sqrt{6\alpha d}$, $2\sigma\sqrt{6\alpha d}$ and $\bar{c} \in \bar{\mathcal{C}}$ such that

$$\begin{bmatrix}\mathbf{A} | \ \mathbf{B} + \bar{m}\mathbf{G} | \ \mathbf{B}'\end{bmatrix}\begin{bmatrix}\bar{\mathbf{s}}_1 \\ \bar{\mathbf{s}}_2 \\ \bar{\mathbf{s}}_3\end{bmatrix} = \bar{c}\mathbf{u}.$$

Proof. Completeness. Completeness follows from equations in the soundness proof. More precisely, if the prover follows honestly his part in the protocol Figure 5, then it follows from Equations (27) to (30), (33) and (34)[10] that the verifier shall always accept all conditions on Equations (14) to (17). Moreover, from Lemma 3.2 of [BLS19], the verifier will accept the conditions on Equations (12) and (13) with overwhelming probability.

Soundness. We construct an extractor $\mathcal{E}$ that with rewindable blackbox access to the prover recovers short vectors $\bar{\mathbf{s}}_1, \bar{\mathbf{s}}_2, \bar{\mathbf{s}}_3, \bar{\mathbf{z}}_4$ and polynomials $\bar{m}, \bar{c}, \bar{e}$ such that :

[9] I.e that the group manager can decrypt it and recover the identity m.

[10] Equation (34) holds because g's first d/l coefficients are set to be 0.

Fig. 5. Interactive proof π

Prover | Verifier

Identity $m, \mathsf{sk} = (\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3)$

$(\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, \mathbf{y}_4) \leftarrow D_\sigma^{2d\alpha} \times D_\sigma^{3d\alpha} \times D_\sigma^{3d\alpha} \times D_{\sigma'}^{(\kappa+\lambda+\alpha+5)d}$

$$\mathbf{w} = \begin{bmatrix}\mathbf{A} | \ \mathbf{B} + m\mathbf{G} | \ \mathbf{B}'\end{bmatrix} \begin{bmatrix}\mathbf{y}_1 \\ \mathbf{y}_2 \\ \mathbf{y}_3\end{bmatrix}$$

$g \leftarrow (\mathbf{0}_{d/l} | \mathbb{Z}_q^{d/l \times (l-1)})$

$$\mathbf{f} = \begin{bmatrix}\mathbf{A}_0 \\ \mathbf{A}_1 \\ \mathbf{a}_2 \\ \mathbf{a}_3 \\ \mathbf{a}_4 \\ \mathbf{t}_1 \\ \mathbf{t}_2\end{bmatrix} \mathbf{r} + \begin{bmatrix} 0 \\ \mathbf{w} \\ \mathbf{a}_3^T \mathbf{y}_4 (2m_{\text{bin}} - 1) \\ m_{\text{bin}} \\ g \\ m \\ \sqrt{q} m \end{bmatrix}$$

$\mathbf{t} = \mathbf{A}_0 \mathbf{y}_4$

$v_2 = \sqrt{q}\mathbf{t}_1^T \mathbf{y}_4 - \mathbf{t}_2^T \mathbf{y}_4$

$v_3 = (\mathbf{a}_3^T \mathbf{y}_4)^2 + \mathbf{a}_2^T \mathbf{y}_4$

$\xrightarrow{\mathbf{f},\ \mathbf{t},\ v_2,\ v_3} c \leftarrow C,\ \phi \leftarrow \mathcal{M}_q^l$

$\xleftarrow{c, \phi}$

$\mathbf{z}_1 = \mathbf{y}_1 + c\mathbf{s}_1$

$\mathbf{z}_2 = \mathbf{y}_2 + c\mathbf{s}_2$

$\mathbf{z}_3 = \mathbf{y}_3 + c\mathbf{s}_3$

$\mathsf{Rej}_0(\mathbf{z}_1, c\mathbf{s}_1, \sigma), \mathsf{Rej}_0(\mathbf{z}_2, c\mathbf{s}_2, \sigma), \mathsf{Rej}_0(\mathbf{z}_3, c\mathbf{s}_3, \sigma)$

$\mathbf{v}_1 = \mathbf{A}_1^T \mathbf{y}_4 - \mathbf{G}\mathbf{z}_2 \mathbf{t}_1^T \mathbf{y}_4$

$v_4 = \left(\mathbf{a}_4^T + \mathbf{a}_3^T \mathsf{NTT}^{-1}\left(\mathbf{Q}^T \phi\right) + \mathbf{t}_1^T \mathsf{NTT}^{-1}(\phi)\right) \mathbf{y}_4$

$j = g + m_{\text{bin}} \mathsf{NTT}^{-1}\left(\mathbf{Q}^T \phi\right) - m \mathsf{NTT}^{-1}(\phi)$

$\xrightarrow{\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, v_1, j, v_4} e \leftarrow C$

$\xleftarrow{e}$

$\mathbf{z}_4 = \mathbf{y}_4 + e\mathbf{r}$

$\mathsf{Rej}_0(\mathbf{z}_4, e\mathbf{r}, \sigma')$ $\xrightarrow{\mathbf{z}_4}$

return $\mathsf{verify}()$

Fig. 6. $\mathsf{verify}(\pi)$ where π is the proof on Figure 5

Accept if :

$$\|\mathbf{z}_1\|_2 \overset{?}{\leqslant} \sigma\sqrt{4\alpha d},\ \|\mathbf{z}_2\|_2 \overset{?}{\leqslant} \sigma\sqrt{6\alpha d},\ \|\mathbf{z}_3\|_2 \overset{?}{\leqslant} \sigma\sqrt{6\alpha d} \tag{12}$$

$$\text{and } \|\mathbf{z}_4\|_2 \overset{?}{\leqslant} \sigma'\sqrt{2(\kappa+\lambda+\alpha+5)d} \tag{13}$$

$$\text{and } \mathbf{A}_0\mathbf{z}_4 \overset{?}{=} \mathbf{t} + e\mathbf{f}_0 \tag{14}$$

$$\text{and } \sqrt{q}\mathbf{t}_1^T\mathbf{z}_4 - \mathbf{t}_2^T\mathbf{z}_4 \overset{?}{=} v_2 + e(\sqrt{q}f_5 - f_6) \tag{15}$$

$$\text{and } \begin{bmatrix} e\mathbf{A} | \ e\mathbf{B} - \mathbf{t}_1^T\mathbf{z}_4\mathbf{G} | \ e\mathbf{B}' \end{bmatrix} \begin{bmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \\ \mathbf{z}_3 \end{bmatrix} + \mathbf{A}_1^T\mathbf{z}_4 \overset{?}{=} \mathbf{v}_1 + ce\mathbf{u} + e(\mathbf{f}_1 - \mathbf{G}\mathbf{z}_2 f_5) \tag{16}$$

$$\text{and } (\mathbf{a}_3^T\mathbf{z}_4 - ef_3)(\mathbf{a}_3^T\mathbf{z}_4 - ef_3 - e) + \mathbf{a}_2^T\mathbf{z}_4 - ef_2 \overset{?}{=} v_3 \tag{17}$$

$$\begin{aligned} &\text{and } e(j - f_4 + \mathsf{NTT}^{-1}\left(\mathbf{Q}^T\phi\right) f_3 - \mathsf{NTT}^{-1}(\phi) f_5) \\ &\quad \overset{?}{=} \mathbf{a}_4^T\mathbf{z}_4\mathbf{a}_3^T\mathbf{z}_4\mathsf{NTT}^{-1}\left(\mathbf{Q}^T\phi\right) - \mathbf{t}_1^T\mathbf{z}_4\mathsf{NTT}^{-1}(\phi) - v_4 \end{aligned} \tag{18}$$

$$\text{and } j_0, \ldots, j_{d/l-1} \overset{?}{=} 0 \tag{19}$$

$$\begin{bmatrix} \mathbf{A} | \ \mathbf{B} + \bar{m}\mathbf{G} | \ \mathbf{B}' \end{bmatrix} \begin{bmatrix} \bar{\mathbf{s}}_1 \\ \bar{\mathbf{s}}_2 \\ \bar{\mathbf{s}}_3 \end{bmatrix} = \bar{c}\mathbf{u} \tag{20}$$

$$\bar{m} \in \mathcal{I} \tag{21}$$

$$\bar{e}\mathbf{f}_0 = \mathbf{A}_0\bar{\mathbf{z}}_4 \tag{22}$$

$$\bar{e}f_5 = \mathbf{t}_1^T\bar{\mathbf{z}}_4 + \bar{e}\bar{m} \tag{23}$$

$$\bar{e}f_6 = \mathbf{t}_2^T\bar{\mathbf{z}}_4 + \bar{e}\sqrt{q}\bar{m} \tag{24}$$

First, we prove that from two transcripts $(\mathbf{f}, j, g, \mathbf{t}, \mathbf{v}, c, \phi, \mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, e, \mathbf{z}_4)$ and $(\mathbf{f}, j, g, \mathbf{t}, \mathbf{v}, c, \phi, \mathbf{z}_1, \mathbf{z}_2, , \mathbf{z}_3, e', \mathbf{z}_4')$, the extractor can recover vectors $\bar{\mathbf{r}}, \bar{\mathbf{w}}$ and a polynomial $\bar{m}$ such that in addition to Equations (23) and (24), we have :

$$\begin{bmatrix} \mathbf{A} | \ \mathbf{B} + \bar{m}\mathbf{G} | \ \mathbf{B}' \end{bmatrix} \begin{bmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \\ \mathbf{z}_3 \end{bmatrix} = \bar{\mathbf{w}} + c\mathbf{u}. \tag{25}$$

Let $\bar{\mathbf{z}}_4 = \mathbf{z}_4 - \mathbf{z}_4'$, $\bar{e} = e - e'$ and $\bar{r} = \bar{e}^{-1}\bar{\mathbf{z}}_4$. The extractor defines the messages in the commitment as follows :

$$\begin{bmatrix} \bar{\mathbf{w}} \\ \bar{\mathsf{garb}} \\ \bar{m} \\ \bar{m}' \\ \overline{m_{\text{bin}}} \\ \bar{g} \end{bmatrix} = \begin{bmatrix} \mathbf{f}_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \end{bmatrix} - \begin{bmatrix} \mathbf{A}_1^T \\ \mathbf{a}_2^T \\ \mathbf{a}_3^T \\ \mathbf{a}_4^T \\ \mathbf{t}_1^T \\ \mathbf{t}_2^T \end{bmatrix} \bar{\mathbf{r}}. \tag{26}$$

The extractor further defines $\bar{\mathbf{y}}_4 = \mathbf{z}_4 - e\bar{\mathbf{r}}$ and $\bar{\mathbf{y}}_4' = \mathbf{z}_4 - e'\bar{\mathbf{r}}$. Equation (22) follows from taking the difference of Equation (14) for both transcripts and Equation (23) follows from the definition of $\bar{m}$ in Equation (26). We substitute $f_5, f_6, \mathbf{z}_4$ in Equation (15) for both transcripts and we obtain :

$$v_2 - (\sqrt{q}\mathbf{t}_1^T\bar{\mathbf{y}}_4 - \mathbf{t}_2^T\bar{\mathbf{y}}_4) + e(\sqrt{q}\bar{m} - \bar{m}') = 0 \tag{27}$$
$$v_2 - (\sqrt{q}\mathbf{t}_1^T\bar{\mathbf{y}}_4 - \mathbf{t}_2^T\bar{\mathbf{y}}_4) + e'(\sqrt{q}\bar{m} - \bar{m}') = 0. \tag{28}$$

We take the difference of both equations and we have $\bar{m}' = \sqrt{q}\bar{m}$, and hence Equation (24). Now, we plug in the expressions of $\mathbf{f}_1, f_5, \mathbf{z}_4$ in Equation (16) for both transcripts, and we obtain :

$$e\left[\mathbf{A}|\mathbf{B} + \bar{m}\mathbf{G}|\ \mathbf{B}'\right]\begin{bmatrix}\mathbf{z}_1\\ \mathbf{z}_2\\ \mathbf{z}_3\end{bmatrix} = e(\bar{\mathbf{w}} + c\mathbf{u}) + \mathbf{v}_1 - \mathbf{A}_1^T\bar{\mathbf{y}}_4 - \mathbf{t}_1^T\bar{\mathbf{y}}_4\mathbf{G}\mathbf{z}_2 \tag{29}$$
$$e'\left[\mathbf{A}|\mathbf{B} + \bar{m}\mathbf{G}|\ \mathbf{B}'\right]\begin{bmatrix}\mathbf{z}_1\\ \mathbf{z}_2\\ \mathbf{z}_3\end{bmatrix} = e'(\bar{\mathbf{w}} + c\mathbf{u}) + \mathbf{v}_1 - \mathbf{A}_1^T\bar{\mathbf{y}}_4 - \mathbf{t}_1^T\bar{\mathbf{y}}_4\mathbf{G}\mathbf{z}_2. \tag{30}$$

Again, we take the difference and we conclude Equation (25).

We will now prove that with overwhelming probability, $\overline{m_{\text{bin}}}, \bar{m}$ are such that

$$\mathbf{Q}\ \mathsf{NTT}\left(\overline{m_{\text{bin}}}\right) = \mathsf{NTT}\left(\bar{m}\right) \tag{31}$$
$$\overline{m_{\text{bin}}}(\overline{m_{\text{bin}}} - 1) = 0. \tag{32}$$

First, we claim that the prover is committed to unique $\bar{\mathbf{y}}_4$ and $\bar{\mathbf{r}}$, that are therefore independent of the challenge. From Lemma 4.1 of [ALS20], if the prover breaks this commitment[11], then $\mathcal{B}$ finds an $\mathsf{MSIS}_{\kappa,\kappa+\alpha+\lambda+5,B_2}$ solution for $\mathbf{A}_1$. Otherwise, we have that

$$\mathbf{z}_4 = \bar{\mathbf{y}}_4 + e\bar{\mathbf{r}} \text{ and } \mathbf{z}_4' = \bar{\mathbf{y}}_4 + e'\bar{\mathbf{r}},$$

and $\bar{\mathbf{y}}_4, \bar{m_{\text{bin}}}, \bar{\mathsf{garb}}$ are independent of the challenge.

Next, we show that $\overline{m_{\text{bin}}}$ verifies Equation (32). We substitute f_2, f_3 and $\mathbf{z}_4$ with respectively $\mathbf{a}_3^T\bar{\mathbf{r}} + \overline{m_{\text{bin}}}$, $\mathbf{a}_2^T\bar{\mathbf{r}} + \bar{\mathsf{garb}}$ and $\bar{\mathbf{y}}_4 + e\bar{\mathbf{r}}$ in Equation (17), and we obtain :

$$\mathbf{a}_2^T\bar{\mathbf{y}}_4 - v_3 + (\mathbf{a}_3^T\bar{\mathbf{y}}_4)^2 + e(\mathbf{a}_3^T\bar{\mathbf{y}}_4(2\overline{m_{\text{bin}}} - 1) - \overline{\mathsf{garb}}) + e^2\overline{m_{\text{bin}}}(\overline{m_{\text{bin}}} - 1) = 0. \tag{33}$$

Since we claimed that (unless the extractor finds an MSIS solution for $\mathbf{A}_1$) $\bar{m_{\text{bin}}}, \bar{\mathsf{garb}}$ and $\bar{\mathbf{y}}_4$ do not depend on e, we can claim that the expression on the left of Equation (33) is a degree 2 polynomial in e. If Equation (32) does not hold, then there exists a prime ideal $(X^{d/l} + \zeta)$ such that Equation (33) mod $(X^{d/l} + \zeta)$ is a degree 2 polynomial in e over the field $\mathcal{R}_q/(X^{d/l} + \zeta)$. This polynomial has at most two roots in this field, say x_1, x_2. Assuming independence, it follows from Lemma 2.3 that the probability that $e \mod (X^{d/l} + \zeta)$ is either of these roots is at most $\frac{2}{q^{d/l}} + O(\epsilon)$, where ϵ is the error term from Lemma 2.3. This probability is negligible, hence we conclude Equation (32).

[11] That is to say $\bar{\mathbf{y}}_4 \neq \bar{\mathbf{y}}_4'$.

We finally prove that $\mathsf{NTT}\left(\overline{m_{\text{bin}}}\right)$ is binary. We just shown that the extracted $\overline{m_{\text{bin}}}$ is such that $\overline{m_{\text{bin}}}(\overline{m_{\text{bin}}}-1)=0$. Let $X^{d/l}-\zeta$ be any of the irreducible factors of $X^d+1 \mod q$. We have the following :

$$\begin{aligned}\overline{m_{\text{bin}}}(\overline{m_{\text{bin}}}-1)&=0\\ \mathsf{NTT}\left(\overline{m_{\text{bin}}}(\overline{m_{\text{bin}}}-1)\right)&=0\\ \mathsf{NTT}\left(\overline{m_{\text{bin}}}\right)\circ\left(\mathsf{NTT}\left(\overline{m_{\text{bin}}}\right)-1\right)&=0.\end{aligned}$$

Since $\mathbb{Z}_q[X]/(X^{d/l}-\zeta)$ is a field, we either have $\overline{m_{\text{bin}}} \mod (X^{d/l}-\zeta)=0$ or $\overline{m_{\text{bin}}} \mod (X^{d/l}-\zeta)=1$. This holds for all the NTT coefficients, from which we conclude Equation (21).

We just proved that $\overline{m_{\text{bin}}}$ is the inverse NTT vector of a binary element of $\mathcal{R}_q$. We then prove that this element is the binary representation of $\bar{m}$ via Equation (31). To do so, we notice that by taking the difference of Equation (18) for both transcripts and plugging the expressions of $\overline{m_{\text{bin}}},\bar{m},\bar{g}$, the extractor finds that the latter variables are such that

$$j=\bar{g}+\overline{m_{\text{bin}}}\mathsf{NTT}^{-1}\left(\mathbf{Q}^T\phi\right)-\bar{m}\mathsf{NTT}^{-1}\left(\phi\right). \tag{34}$$

Equation (19) says that the first d/l coefficients of j are 0. On the other hand, using Lemmas 2.1 and 2.2, we have

$$l\sum_{i=0}^{d/l-1}j_i=\sum_{i=0}^{d/l-1}g_i+\langle\mathbf{Q}\ \mathsf{NTT}\left(\overline{m_{\text{bin}}}\right)-\mathsf{NTT}\left(\bar{m}\right),\phi\rangle,$$

where the latter equality is over $\mathbb{Z}_q^l$. If $\mathbf{Q}\ \mathsf{NTT}\left(\overline{m_{\text{bin}}}\right)-\mathsf{NTT}\left(\bar{m}\right)\neq 0$, then since the challenge ϕ is uniformly random, so is $\langle\mathbf{Q}\ \mathsf{NTT}\left(\overline{m_{\text{bin}}}\right)-\mathsf{NTT}\left(\bar{m}\right),\phi\rangle$. Notice that g was committed to by the prover prior to its knowledge of ϕ, and therefore, the probability that Equation (34) holds without Equation (31) being true is $\frac{1}{q^{d/l}}$. Since this probability is negligible, we conclude Equation (31).

Finally, to prove that $\bar{m}$ is a valid identity, we notice that Equation (31) yields that all NTT coefficients of $\bar{m}$ are equal. Together with the fact that $\overline{m_{\text{bin}}}$ is binary, this yields that $\mathsf{NTT}\left(\bar{m}\right)=(\bar{m},\ldots,\bar{m})$, and it follows that $\bar{m}\in\mathcal{I}$.

We now prove that $\mathcal{E}$ can extract $\bar{\mathbf{s}}_1,\bar{\mathbf{s}}_2,\bar{\mathbf{s}}_3$ that together with the previously extracted $\bar{m}$ (that, we showed, verifies Equations (21) and (23)) verifying Equation (20). The extractor acquires 4 transcripts

$$\begin{aligned}&(\mathbf{f},\mathbf{t},\mathbf{v},c,\phi,j,\mathbf{z}_1,\mathbf{z}_2,\mathbf{z}_3,e,\mathbf{z}_4)\\ &(\mathbf{f},\mathbf{t},\mathbf{v},c,\phi,j,\mathbf{z}_1,\mathbf{z}_2,\mathbf{z}_3,e',\mathbf{z}_4')\\ &(\mathbf{f},\mathbf{t},\mathbf{v}_1',v_2,v_3,v_4',c',\phi,j,\mathbf{z}_1',\mathbf{z}_2',\mathbf{z}_3',e'',\mathbf{z}_4'')\\ &(\mathbf{f},\mathbf{t},\mathbf{v}_1',v_2,v_3,v_4',c',\phi,j,\mathbf{z}_1',\mathbf{z}_2',\mathbf{z}_3',e^{(3)},\mathbf{z}_4^{(3)}).\end{aligned}$$

We proceed to describe how the extractor gets those transcripts and what his success probability is. Let ϵ be the probability of a deterministic prover to produce a proof that passes verification. The extractor first runs $\frac{\log 10}{\epsilon}$ times the prover, or until the prover returns a valid transcript $(\mathbf{f},\mathbf{t},\mathbf{v},c,\phi,j,\mathbf{z}_1,\mathbf{z}_2,\mathbf{z}_3,e,\mathbf{z}_4)$. If the prover fails to do so, $\mathcal{E}$ aborts. Next, $\mathcal{E}$ runs the prover $\frac{\log 10}{\epsilon/2-C(c)}$

times, answering the same challenge c (and ϕ) in the first verifier interaction, and challenges $e' \leftarrow C$ in the second verifier interaction. Again, if the prover fails to produce a valid transcript, $\mathcal{E}$ aborts. Otherwise, $\mathcal{E}$ then receives a second transcript $(\mathbf{f}, \mathbf{t}, \mathbf{v}, c, \phi, j, \mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, e', \mathbf{z}_4')$. Thirdly, $\mathcal{E}$ runs the prover on fresh challenges (c', ϕ', e'') with the only condition that $c' \neq c$ for a total of $\frac{\log 10}{\epsilon - C(c)}$ times. Unless the prover provided a valid transcript $(\mathbf{f}, \mathbf{t}, \mathbf{v}_1', v_2, v_3, v_4', c', \phi', j, \mathbf{z}_1', \mathbf{z}_2', \mathbf{z}_3', e'', \mathbf{z}_4'')$, $\mathcal{E}$ aborts. Finally, $\mathcal{E}$ repeats the second step : $\mathcal{E}$ runs the prover on c', ϕ' with fresh $e^{(3)} \leftarrow C \backslash \{e''\}$ for a total of $\frac{\log 10}{\epsilon/2 - C(e'')}$ times.

We now calculate the probability that $\mathcal{E}$ never aborts and indeed acquires the 4 transcripts. The extractor has 4 opportunities to abort and thus not receive 4 transcripts from the prover. We write $\mathbf{abort}_i$ the event where $\mathcal{E}$ acquires the first $i-1$ transcripts but aborts when done trying to get the i-th. Since the failure probability of the prover for one iteration is $1-\epsilon$, $\Pr(\mathbf{abort}_1) = (1-\epsilon)^{\log 10/\epsilon} \leqslant \exp(-\epsilon)^{\log 10/\epsilon}$, and finally $\Pr(\mathbf{abort}_1) \leqslant 1/10$. For $\mathcal{E}$ to abort in step 2, $\mathcal{E}$ must have received a first transcript for challenges c, ϕ, e. From the heavy-rows lemma [OO98], the probability that the (c, ϕ) row is heavy[12] is at least 1/2. Moreover, from the definition of a heavy row, the success probability of the prover when (c, ϕ) yields a heavy row is at least $\epsilon' = \epsilon/2 - \Pr_{c' \leftarrow C}(c' = c)$. Therefore, we have $\Pr(\mathbf{abort}_2) \leqslant (1-\epsilon')^{\log 10/\epsilon'} + 1/2 \leqslant 3/5$. Similarly, we have $\Pr(\mathbf{abort}_3) \leqslant 1/10$, and $\Pr(\mathbf{abort}_4) \leqslant 3/5$. The probability that $\mathcal{E}$ never aborts is the product of $1 - \Pr(\mathbf{abort}_i)$ for $i = 1, 2, 3, 4$, which is given by $81/625 \geqslant 1/8$.

Using the previous result on both the first pair and the second pair of transcripts, the extractor finds $\bar{\mathbf{r}}, \bar{m}, \bar{\mathbf{w}}$ and $\bar{\mathbf{r}}', \bar{m}', \bar{\mathbf{w}}'$ that verify Equations (21) to (24). Since $\bar{\mathbf{r}}\bar{e}' - \bar{\mathbf{r}}'\bar{e}$ is small[13], then $(\bar{\mathbf{r}}, \bar{m}, \bar{\mathbf{w}}) = (\bar{\mathbf{r}}', \bar{m}', \bar{\mathbf{w}}')$. The extractor defines $\bar{\mathbf{s}}_1 = \mathbf{z}_1 - \mathbf{z}_1', \bar{\mathbf{s}}_2 = \mathbf{z}_2 - \mathbf{z}_2', \bar{\mathbf{s}}_3 = \mathbf{z}_3 - \mathbf{z}_3', \bar{c} = c - c'$. Then, using Equation (24), $\bar{\mathbf{s}}_1, \bar{\mathbf{s}}_2, \bar{\mathbf{s}}_3, \bar{c}$ verify Equation (20), and therefore the protocol is sound.

Zero-knowledge. We define the simulator $\mathcal{S}$ as follows :

1. Generate $c \leftarrow C, \phi \leftarrow \mathcal{M}_q^l, e \leftarrow C$
2. Generate $(\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \mathbf{z}_4) \leftarrow D_\sigma^{2d\alpha} \times D_\sigma^{3d\alpha} \times D_\sigma^{3d\alpha} \times D_{\sigma'}^{(\kappa+\lambda+\alpha+5)d}$
3. Generate $\mathbf{f} \leftarrow \mathcal{R}_q^{\kappa+\alpha+5}, \mathbf{r} \leftarrow \chi^{\kappa+\lambda+\alpha+5}$
4. Generate $j \leftarrow (\mathbf{0}_{d/l} | \mathbb{Z}_q^{d/l \times (l-1)})$
5. Set $\mathbf{v}, \mathbf{t}$ so Equations (14) to (17) hold
6. Output $(\mathbf{f}, \mathbf{t}, j, \mathbf{v}, \mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \mathbf{z}_4, c, \phi, e)$

To conclude zero-knowledge for π, we show that the distribution output by $\mathcal{S}$ is indistinguishable from the distribution of a non-aborting accepting transcript from Figure 5. The variables c, ϕ, e are distributed exactly as in the procedure. The vectors $(\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \mathbf{z}_4)$ in non-aborting proofs follow a distribution independent to c, ϕ, e that is indistinguishable ([BLS19], Lemma 3.2) of $D_\sigma^{2d\alpha} \times D_\sigma^{3d\alpha} \times D_\sigma^{3d\alpha} \times D_{\sigma'}^{(\kappa+\lambda+\alpha+5)d}$, which is their distribution in the output of $\mathcal{S}$.

Under the hardness of $\mathsf{MLWE}_{\kappa+\lambda+\alpha+5,\kappa,S_\mu}$, $\mathbf{t}_1, \mathbf{t}_2$ are indistinguishable from uniform. Under the hiding property of the commitment scheme, which in turn relies on the hardness of $\mathsf{MLWE}_{\kappa+\alpha+5,\lambda,\chi}$, the distribution of $\mathbf{f}$ in honestly generated transcripts is indistinguishable from uniform, which is its distribution in the output of the simulator $\mathcal{S}$.

[12] Challenges (c, ϕ) are in a *heavy row* when the success probability of the prover conditionned on the first challenges to be these c, ϕ is at least $\epsilon/2$. We refer to [OO98] for further detail.

[13] Otherwise, $\bar{\mathbf{r}}\bar{e}' - \bar{\mathbf{r}}'\bar{e}$ is a solution for MSIS for $\mathbf{A}_0$ of norm at most $8\omega^2\sigma'\sqrt{2(\kappa+\lambda+\alpha+5)d}$

Equation (14) uniquely determines $\mathbf{t} = \mathbf{A}_0\mathbf{z}_4 - e\mathbf{f}_0$ from $\mathbf{z}_4, e, \mathbf{f}_0$. Similarly, all the coordinates of $\mathbf{v}$ are uniquely determined by the sampled variables of $\mathcal{S}$ from Equations (15) to (18). To summarize, the variables that $\mathcal{S}$ samples follow a distribution indistinguishable from the one from the actual interaction and the other variables are binded by the verification equations in the accepting transcripts, from which we conclude zero-knowledge.

3.2 Decryption

The verifiable encryption scheme of our group signature scheme is hidden in the commitment scheme. The group manager sets the commitment public vectors for m and $\sqrt{q}m$ to be $\mathbf{t}_1 = \mathbf{A}_0^T\mathbf{h}_1 + \mathbf{e}_1$,$\mathbf{t}_2 = \mathbf{A}_0^T\mathbf{h}_2 + \mathbf{e}_2$, where $\mathbf{h}_1, \mathbf{h}_2, \mathbf{e}_1, \mathbf{e}_2$ are the group manager decryption secret key. The idea of the encryption is the following : the ciphertext is of the form

$$\mathbf{A}_0\mathbf{r} = \mathbf{u}_0 \tag{35}$$

$$\mathbf{t}_1^T\mathbf{r} + m = u_1 \tag{36}$$

$$\mathbf{t}_1^T\mathbf{r} + \sqrt{q}m = u_2. \tag{37}$$

The steps to decrypt using the secrets $\mathbf{h}_1, \mathbf{h}_2, \mathbf{e}_1, \mathbf{e}_2$ are as follows. Compute $x_1 = u_1 - \mathbf{h}_1^T\mathbf{u}_0 = \mathbf{h}_1^T\mathbf{r} + m, x_2 = u_2 - \mathbf{h}_2^T\mathbf{u}_0 = \mathbf{h}_2^T\mathbf{r} + \sqrt{q}m$. Then, compute $\sqrt{q}x_1 - x_2 = (\sqrt{q}\mathbf{h}_1^T - \mathbf{h}_2^T)\mathbf{r}$. If this latter polynomial has all its coefficients less than $q/2$, then this equality holds over the integers. Moreover, if we take the parameters such that $\mathbf{h}_1^T\mathbf{r} \leqslant \sqrt{q}$, then we have $k = \sqrt{q}x_1 - x_2 \mod \sqrt{q} = -\mathbf{h}_1^T\mathbf{r}$. To finish, we have $m = (x_2 + k)/\sqrt{q}$, provided that $(\sqrt{q}\mathbf{h}_1^T - \mathbf{h}_2^T)\mathbf{r} \leqslant q/2$ and $\mathbf{h}_1^T\mathbf{r} \leqslant \sqrt{q}$.

The problem is that the proof that the ciphertext is valid given through π does not ensure Equations (35) to (37) but rather a relaxed proof that there exists a $\bar{e} \in \bar{\mathcal{C}}$ and $\bar{r}$ such that

$$\mathbf{A}_0\bar{\mathbf{r}} = \bar{e}\mathbf{u}_0$$

$$\mathbf{t}_1^T\bar{\mathbf{r}} + \bar{e}m = \bar{e}u_1$$

$$\mathbf{t}_1^T\bar{\mathbf{r}} + \bar{e}\sqrt{q}m = \bar{e}u_2.$$

We use a similar technique as [LN17, dPLS18]. The idea is that the group manager is given a proof π for a challenge e. The soundness of the proof ensures that there exists another challenge e' such that with $\bar{e} = e - e'$ (and notations from π) : $(\bar{e}\mathbf{f}_0, \bar{e}f_5, \bar{e}f_6)$ is a valid ciphertext. The known technique that we use to tackle this is to try and decrypt $(\bar{e}\mathbf{f}_0, \bar{e}f_5, \bar{e}f_6)$ for a random second challenge $e' \leftarrow C$. Possibly not any challenge e' yields a decryption to the right message m. What [LN17] shown for their verifiable encryption scheme is that a simple test condition can 1) reject the challenges e' that do not yield a decryption to m 2) Not reject too many challenges so the decryption runtime is reasonable. We have a similar result for our decryption, where the condition on Line 6 plays this role, where both correctness of the decryption and 'reasonable' runtime are stated in Lemma 3.2.

Lemma 3.2. *If the verification of a proof π from Figure 5 passes, then Algorithm 1 on input π and the group manager secret key returns a unique decryption in expected running-time at most $O(h_2)$, where h_2 is the number of queries to the second random oracle made by the prover to generate a signature. For an honest prover, the expected number of iterations is $\sqrt{3}$.*

Proof. Let π be a transcript from the zero-knowledge proof described in Figure 5 for which the verifier accepts. From the extraction, we know that the verifier is convinced that there exists $e', \bar{\mathbf{r}}$

Algorithm 1 Decryption algorithm $\mathsf{GSdec}(\pi, \mathbf{h}_1, \mathbf{h}_2, \mathbf{e}_1, \mathbf{e}_2)$:

```
1: $e' \leftarrow C$
2: $\bar{e} = e - e'$
3: $x_1 = f_5 - \mathbf{h}_1^T \mathbf{f}_0$
4: $x_2 = f_6 - \mathbf{h}_2^T \mathbf{f}_0$
5: $k = \bar{e}(\sqrt{q}x_1 - x_2) \mod \sqrt{q}$
6: **if** $\|\bar{e}(\sqrt{q}x_1 - x_2)\|_\infty \leqslant \frac{q}{4\omega}$ **then**
7:     **return** $(\bar{e}x_2 + k)/(\sqrt{q}\bar{e})$
8: **else** go to Line 1
9: **end if**
```

and m^* such that with $\bar{e} = e - e'$:

$$\mathbf{A}_0 \bar{\mathbf{r}} = \bar{e}\mathbf{f}_0$$
$$\mathbf{t}_1^T \bar{\mathbf{r}} + \bar{e}m^* = \bar{e}f_5$$
$$\mathbf{t}_2^T \bar{\mathbf{r}} + \sqrt{q}\bar{e}m^* = \bar{e}f_6.$$

We chose parmeters such that 1) $\|\mathbf{e}_2^T \bar{\mathbf{r}}\|_\infty \leqslant \frac{\sqrt{q}}{2\omega}$ and 2) $\|\bar{e}(\sqrt{q}\mathbf{e}_1^T - \mathbf{e}_2^T)\bar{\mathbf{r}}\|_\infty \leqslant \frac{q}{4\omega}$. This yields that if this e' is the one sampled in Line 1, then the equation $\bar{e}(\sqrt{q}x_1 - x_2) = \bar{e}(\sqrt{q}\mathbf{e}_1^T - \mathbf{e}_2^T)\bar{r}$ holds over the integers. This further means that $k = -\bar{e}\mathbf{e}_2^T \bar{\mathbf{r}}$, and hence if this e' is sampled on Line 1 then the decryption output by GSdec is m^*.

Expected run time. Let $e' \in \mathcal{C}$ be the challenge sampled in Line 1. From the discussion above, we see that if the prover's response for e' passes verification, then the condition on Line 6 is verified. Therefore, by sampling e' from C, we have that the probability that an iteration of GSdec passes Line 6 is at least the probability that the prover returns a response to a random challenge from the verifier. When the prover follows the protocol Figure 5, this probability is 1− the rejection rate, which yields an expected number of iterations lower than $\sqrt{3}$.

Uniqueness.

We run Algorithm 1 on π with the group manager key twice and get two decryptions m, m' for different sampled challenges inducing respectively $\bar{e}, \bar{e}'$. We have

$$\begin{aligned} m = (\bar{e}x_2 + k)/(\sqrt{q}\bar{e}) &= \frac{e'}{e'}(\bar{e}x_2 + k)/(\sqrt{q}\bar{e}) \\ &= (\bar{e}\bar{e}'\sqrt{q})^{-1}(\bar{e}\bar{e}'x_2 + \bar{e}'(\bar{e}(\sqrt{q}x_1 - x_2) \mod \sqrt{q})) \qquad (38) \\ &= (\bar{e}\bar{e}'\sqrt{q})^{-1}(\bar{e}\bar{e}'x_2 + (\bar{e}\bar{e}'(\sqrt{q}x_1 - x_2) \mod \sqrt{q})) \qquad (39) \end{aligned}$$

and

$$\begin{aligned} m' &= (\bar{e}'x_2 + k)/(\sqrt{q}\bar{e}') \\ &= (\bar{e}\bar{e}'\sqrt{q})^{-1}(\bar{e}\bar{e}'x_2 + (\bar{e}\bar{e}'(\sqrt{q}x_1 - x_2) \mod \sqrt{q})). \qquad (40) \end{aligned}$$

Equation (38) to Equation (39) and Equation (40) are correct since $x_1, x_2, \bar{e}$ and $x_1, x_2, \bar{e}'$ passed the condition on Line 6 and therefore the multiplication with $\bar{e}$ and $\bar{e}'$ respectively does not induce a reduction $\mod q$. This way, the reduction $\mod \sqrt{q}$ can be taken after the multiplication with $\bar{e}$ or $\bar{e}'$ respectively. The right hand side of Equations (39) and (40) are identical, henceforth $m = m'$. This means that no matter what e' is used in the decryption, we have $\mathsf{GSdec}(\pi, \mathbf{h}_1, \mathbf{h}_2, \mathbf{e}_1, \mathbf{e}_2) = m^*$, which completes the proof.

4 Security and parameters

In this section, prove the two security notions required for a group signature, that is, anonymity and traceability. Afterwards, we propose a set of parameters for the group signature that achieve a signature size of rougly 203 KB.

4.1 Security

Throughout this subsection, we will write $\epsilon_{\mathcal{A}}^{G}$ the success probability of an adversary $\mathcal{A}$ against a game G. The proof for traceability is done in two steps : we reduce the traceability of our scheme to a hybrid trace^* game, and then reduce the latter to lattice problems. The trace^* game (more formally defined as G_{11} of Appendix B) is informally defined as follows. The challenger $\mathcal{B}$ runs KeyGen honestly, except for the following steps :

1. The public matrix $[\mathbf{A}|\mathbf{B}|\mathbf{B}']$ is crafted such that $\mathbf{A}$ is uniformly random, $\mathbf{B} = \mathbf{AR} - m^*\mathbf{G}$ where $\mathbf{R}$ is generated as in KeyGen and $m^* \leftarrow \mathcal{I}$ is a uniformly random identity, and $\mathbf{B}' = \mathbf{AR}'$ where $\mathbf{R}'$ is distributed as $\mathbf{R}$.
2. The vector $\mathbf{u}$ is defined as $\mathbf{u} = \mathbf{A}\mathbf{s}_1 + \mathbf{AR}\mathbf{s}_2 + \mathbf{AR}'\mathbf{s}_3$, and thus corresponds to the identity m^*.

Similarly as in the traceability game for our scheme, the adversary can query secret keys, signatures and the random oracles.

Lemma 4.1. *Let $\mathcal{A}$ be an adversary to the traceability game for our scheme. Let ZK be the zero-knowledge game for π. We have*

$$\epsilon_{\mathcal{A}}^{\mathsf{traceability}} \leqslant 5\epsilon_{\mathcal{A}}^{\mathsf{MLWE}_{\alpha,\alpha,S_1}} + \epsilon_{\mathcal{A}}^{\mathsf{ZK}} + \epsilon_{\mathcal{A}}^{\mathsf{trace}*}.$$

The proof is deferred to Appendix B.

Lemma 4.2. *let $\mathcal{A}$ be an adversary that runs in time T and has success probability ϵ against the $\mathsf{trace}*$ game. Let h_1 (respectively h_2) be the number of queries that $\mathcal{A}$ can make to the first random oracle (respectively to the second random oracle), $B \geqslant 4\sigma\sqrt{d\alpha}(1 + 2\omega)(3d\alpha + 1)$ and $B_2 \geqslant 8\omega^2\sigma'\sqrt{2d(\kappa + \lambda + \alpha + 5)}$. Then, there exists an adversary $\mathcal{B}$ that runs in time $O(T/\epsilon)$ and that has probability at least $1/(8|\mathcal{I}|)$ to find either a solution to $\mathsf{MSIS}_{\alpha,2\alpha,B}$ or to $\mathsf{MSIS}_{\kappa,\kappa+\lambda+\alpha+5,B_2}$.*

Proof. Let $\mathcal{A}$ be an adversary. We assume that with at most h_1 queries to the first random oracle, h_2 queries to the second random oracle and Q queries to the signing algorithm, $\mathcal{A}$ has a probability ϵ of successfully outputing a forgery. Let $\mathcal{B}$ be an algorithm that can query $\mathcal{A}$. The goal of $\mathcal{B}$ is to either solve $\mathsf{MSIS}_{\alpha,2\alpha,B}$ for some matrix $\mathbf{X} \in \mathcal{R}_q^{\alpha\times 2\alpha}$, or solve $\mathsf{MSIS}_{\kappa,\kappa+\lambda+\alpha+5,B_2}$ for some matrix $\mathbf{Y}$.

Description of $\mathcal{B}$:

Given the instances of MSIS, $\mathcal{B}$ sets the public parameters of the scheme honestly, except for the following steps. The public matrix $\mathbf{A}$ is set as $\mathbf{A} = \mathbf{X}$, and $\mathbf{B} = \mathbf{XR} - m\mathbf{G}^T$ for some uniformly random guess m of the identity that the adversary is going to impersonate. The public commitment matrix $\mathbf{A}_0$ is set as $\mathbf{Y}$. The parameters of this game are identical to the $\mathsf{trace}*$ game, therefore provided that $\mathcal{B}$ can answer secret key signature queries, $\mathcal{A}$ shall have a probability ϵ to output a forgery. Note that on top of his knowledge of the trapdoors $\mathbf{R}, \mathbf{R}'$, $\mathcal{B}$ knows the honestly generated secret vector $\mathbf{s}_1^{\mathsf{gm}}, \mathbf{s}_2^{\mathsf{gm}}, \mathbf{s}_3^{\mathsf{gm}}$ and defines u as

$$u = \begin{bmatrix}\mathbf{X} | \ \mathbf{XR} | \ \mathbf{AR}'\end{bmatrix} \begin{bmatrix}\mathbf{s}_1^{\mathsf{gm}} \\ \mathbf{s}_2^{\mathsf{gm}} \\ \mathbf{s}_3^{\mathsf{gm}}\end{bmatrix}.$$

To answer a secret key query for an identity $m' \neq m^*$ from $\mathcal{A}$, $\mathcal{B}$ uses his knowledge of the trapdoor $\mathbf{R}$ to sample a valid secret key $(\mathbf{s}_1^{m'}, \mathbf{s}_2^{m'}, \mathbf{s}_3^{m'})$ such that

$$\begin{bmatrix}\mathbf{X} | \ \mathbf{XR} + (m' - m^*)\mathbf{G} | \ \mathbf{AR}'\end{bmatrix} \begin{bmatrix}\mathbf{s}_1^{m'} \\ \mathbf{s}_2^{m'} \\ \mathbf{s}_3^{m'}\end{bmatrix} = u.$$

If $\mathcal{A}$ queries the secret key for m^*, $\mathcal{B}$ is unable to use its trapdoor $\mathbf{R}$ so he returns $(\mathbf{s}_1^{\mathsf{gm}}, \mathbf{s}_2^{\mathsf{gm}}, \mathbf{s}_3^{\mathsf{gm}})$.

To answer $\mathcal{A}$'s signing queries, $\mathcal{B}$ will run the simulator $\mathcal{S}$ from the zero-knowledge property of π. From Theorem 3.1, the distribution of the transcripts that $\mathcal{B}$ sends to $\mathcal{A}$ is computationnally indistinguishable from the actual distribution of the transcripts from the signature, and therefore $\mathcal{A}$ can indeed be provided with as many as Q signing queries.

When $\mathcal{A}$ is ready to produce forgeries, $\mathcal{B}$ will follow the transcript-acquisition from the extractor $\mathcal{E}$ of the soundness of π. More precisely, $\mathcal{B}$ has a probability at least 1/8 to get 4 forged signatures from $\mathcal{A}$ in time $O(T/\epsilon)$: 2 different second challenges e per first different challenges c, ϕ. We reuse the notations from the extraction.

Next, $\mathcal{B}$ will proceed as $\mathcal{E}$ and recover either a solution to $\mathsf{MSIS}_{\kappa,\kappa+\lambda+\alpha+5,B_2}$ - or recover vectors $\bar{\mathbf{s}}_1, \bar{\mathbf{s}}_2, \bar{\mathbf{s}}_3, \bar{\mathbf{r}}$ and polynomials $m, \bar{c}$ such that $m \in \mathcal{I}$ and $\mathbf{X}\bar{\mathbf{s}}_1 + (\mathbf{XR} + (m - m^*)\mathbf{G})\bar{\mathbf{s}}_2 + \mathbf{XR}'\bar{\mathbf{s}}_3 = \mathbf{u}$, which completes the proof either way.

From Lemma 3.2, the decryption of all of $\mathcal{A}$'s forgeries are the same $m \in \mathcal{I}$. Since $\mathcal{A}$ is playing the trace^* game, we assume that he never queried nor received the secret key for identity m. With probability $|\mathcal{I}|^{-1}$, $\mathcal{B}$'s uniformly random guess m^* is indeed the identity that $\mathcal{A}$ impersonates. If this did not happen, then $\mathcal{B}$ fails and aborts. From now on, we assume that $m = m^*$. On one hand, $\mathcal{B}$ received from $\mathcal{A}$ short vectors that satisfy $\mathbf{X}(\bar{\mathbf{s}}_1 + \mathbf{R}\bar{\mathbf{s}}_2 + \mathbf{R}'\bar{\mathbf{s}}_3) = \bar{c}u$. On the other hand, $\mathcal{B}$ knows $(\mathbf{s}_1^{\mathsf{gm}}, \mathbf{s}_2^{\mathsf{gm}}, \mathbf{s}_3^{\mathsf{gm}})$ verifying $\mathbf{X}(\mathbf{s}_1^{\mathsf{gm}} + \mathbf{R}\mathbf{s}_2^{\mathsf{gm}} + \mathbf{R}\mathbf{s}_3^{\mathsf{gm}}) = u$. In other words, $\bar{\mathbf{s}}_1 - \bar{c}\mathbf{s}_1^{\mathsf{gm}} + \mathbf{R}(\bar{\mathbf{s}}_2 - \bar{c}\mathbf{s}_2^{\mathsf{gm}}) + \mathbf{R}'(\bar{\mathbf{s}}_3 - \bar{c}\mathbf{s}_3^{\mathsf{gm}})$ is a solution to MSIS for the given random matrix $\mathbf{X}$ of size with overwhelming probability at most $4\sigma\sqrt{d\alpha}(1 + 2\omega)(3d\alpha + 1)$.

Theorem 4.3. *The group signature scheme, where the signature is the Fiat-Shamir transform of π defined in Figure 5 is untraceable and anonymous.*

More precisely for anonymity, the advantage of an adversary $\mathcal{A}$ against the anonymity game is upper bounded by $2^{99}/M + 2\epsilon_{\mathcal{A}}^{\mathsf{MLWE}}$.

For traceability, with $B \geqslant 4\sigma\sqrt{d\alpha}(1 + 2\omega)(3d\alpha + 1)$ and

$$B_2 \geqslant 8\omega^2\sigma'\sqrt{2d(\kappa + \lambda + \alpha + 5)},$$

the success probability $\epsilon_{\mathcal{A}}^{\mathsf{traceability}}$ of an adversary $\mathcal{A}$ against the traceability game is upper bounded by

$$3\epsilon_{\mathcal{A}}^{\mathsf{MLWE}_{\alpha,\alpha,S_1}} + 8|\mathcal{I}| \max(\epsilon_{\mathcal{A}}^{\mathsf{MSIS}_{\alpha,2\alpha,B}}, \epsilon_{\mathcal{A}}^{\mathsf{MSIS}_{\kappa,\kappa+\lambda+\alpha+5,B_2}}).$$

The proof is deferred to Appendix B.

4.2 Parameters

Similarly as in [dPLS18], we apply the Fiat-Shamir transformation [FS86] on the interactive protocol in Fig. 5 to obtain a group signature. We first compute sizes for our signature and then propose several common optimisations.

To begin with, we set $(q, d, l) = (\approx 2^{64}, 128, 64)$ so that $q^{-d/l} \approx \mathsf{p}^{d/l} \approx 2^{-128}$. Next, we aim for the repetition rate of our protocol to be 27 as in [dPLS18]. Hence, we set M such that $M^2 = 27$ [14], i.e. $M = 3^{3/2}$. We compute an upper-bound T_s on$\|c(\boldsymbol{s}_1||\boldsymbol{s}_2||\boldsymbol{s}_3)\|$, where $c \leftarrow C$, as follows. Recall that using a trapdoor sampling similar to [dPLS18], coefficients of vectors $\boldsymbol{s}_i$ follow a discrete Gaussian distribution with standard deviation $\mathfrak{s}_{tr} \leqslant 2(3\sqrt{\alpha d}+1)\sqrt{[q^{1/3}]^2+1}$ (see [dPLS18, Section 2.6] for more details). Hence, by Lemma 2.7 with an overwhelming probability we have

$$\|c(\boldsymbol{s}_1||\boldsymbol{s}_2||\boldsymbol{s}_3)\|^2 = \sum_{i=1}^{3} \|c\boldsymbol{s}_i\|^2 \leqslant \sum_{i=1}^{3} (d\|\boldsymbol{s}_i\|)^2 \leqslant 16d^2\mathfrak{s}_{tr}^2\alpha d.$$

Thus, we set $T_s = 8(3\sqrt{\alpha d}+1)d\sqrt{[q^{1/3}]^2+1}\sqrt{\alpha d}$ and $\sigma = 8T_s$ in order to have $M = 3^{3/2}$ (as in Equation 8).

Let T_r be an upper-bound on $\|e\boldsymbol{r}\|$ where e is the challenge in Fig. 5. We apply the exact method as in [LNS21a, Appendix C]. Namely, we use the observation that

$$\|e\boldsymbol{r}\|^2 \leqslant d \left\| \sum_{i=1}^{\ell} \sigma_{-1}(r_i) r_i \right\|_1$$

where $\boldsymbol{r} = (r_1, \ldots, r_{\kappa+\lambda+\alpha+5})$ and σ_{-1} is the Galois automorphism $\sigma_{-1} : X \longmapsto X^{-1}$. Then, we heuristically choose T_r so that the expression on the right-hand side is less than T_r^2 with probability at least 99%. Similarly as before, we set $\mathfrak{s}' = 8T_r$.

In Fig. 2 we choose parameters $\kappa, \lambda, \alpha, \mu$ so that the MSIS and MLWE problems described in the previous subsections are hard. For a fair comparison with [dPLS18], we measure the hardness with the root Hermite factor δ and aim for $\delta \approx 1.0036$.

We now turn to computing the signature size. As "full-sized" elements of $\mathcal{R}_q$ we have $\boldsymbol{f}$ and j (it is missing d/l coefficients but this has negligible impact on the sizes). Therefore, we have in total $\kappa + \alpha + 5 + 1$ full elements of $\mathcal{R}_q$ which give us

$$(\kappa + \alpha + 6)d \log q \text{ bits.}$$

What we have left are vectors of short polynomials $\boldsymbol{z}_1, \boldsymbol{z}_2, \boldsymbol{z}_3$ and $\boldsymbol{z}_4$. Since they come from a discrete Gaussian distribution with standard deviation $\mathfrak{s}$ and $\mathfrak{s}'$ respectively, with high probability we can upper-bound their coefficients by $6\mathfrak{s}$ and $6\mathfrak{s}'$ [Lyu12]. Thus, they require at most:

$$8\alpha d \log(12\mathfrak{s}) + (\kappa + \lambda + \alpha + 5)d \log(12\mathfrak{s}') \text{ bits.}$$

Finally, the challenges c, e cost at most $4 \cdot d = 512$ bits.

[14] Recall that in Fig. 5 we run four rejection algorithms. However, for efficiency purposes we can merge the ones for $\boldsymbol{z}_1, \boldsymbol{z}_2, \boldsymbol{z}_3$ since they follow the same standard deviation σ.

Various optimisations. First, we apply the rejection strategy introduced [LNS21a] for $\boldsymbol{z}_4$. Namely. we use the algorithm Rej_1 defined in Fig. 3 instead of Rej_0. Consequently, we manage to significantly reduce the standard deviation $\mathfrak{s}'$ at a cost of leaking one bit of the randomness $\boldsymbol{r}$. This is fine in our case since each new signature requires a fresh randomness vector.

We cannot use the rejection approach from [LNS21a] for $\boldsymbol{z}_1, \boldsymbol{z}_2, \boldsymbol{z}_3$ since each signature would reveal some more information about secret vectors $\boldsymbol{s}_i$. In order to reduce the standard deviation σ, we will use Bimodal Gaussians [DDLL13] instead. We remark that this technique is not new and it was recently used in e.g. [LNS21b, Section 1.5] and [LNS21a, Appendix B].

Concretely, we additionally commit to a randomly chosen sign $b \in \{-1, 1\}$:

$$f_8 = \boldsymbol{a}_5^T \boldsymbol{r} + b.$$

Then, we send $\boldsymbol{z}_i = \boldsymbol{y} + bc\boldsymbol{s}_i$ for $i = 1, 2, 3$ and later prove that

$$\begin{bmatrix} \boldsymbol{A} | \ \boldsymbol{B} + m\boldsymbol{G} | \ \boldsymbol{B}' \end{bmatrix} \begin{bmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \\ \mathbf{z}_3 \end{bmatrix} = \boldsymbol{w} + bc\boldsymbol{u}$$

where BDLOP commitments to m, b and $\boldsymbol{w}$ are given. Furthermore, we need to prove that f_8 is a commitment to -1 or 1. First, we prove that $(b+1)(b-1) = 0$ over $\mathcal{R}_q$. This implies that the NTT coefficients of b are either -1 or 1. Next, we show that all coefficients of b are the same, i.e. the NTT vector $\vec{b} = \mathsf{NTT}(b)$ of b satisfies $V\vec{b} = \vec{0}$ where

$$V = \begin{pmatrix} 1 & -1 & 0 & \dots & 0 \\ 0 & 1 & -1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -1 & 0 & 0 & \dots & 1 \end{pmatrix} \in \mathbb{Z}_q^{l \times l}.$$

Since we already prove linear and multiplicative relations in Fig. 5, the additional proofs for b do not affect the total signature size (as mentioned in Appendix A.1). Hence, we manage to decrease the standard deviation σ at a cost of committing to one more polynomial b. Eventually, with the aforementioned optimisations, we manage to decrease the standard deviations to $\sigma = 0.7T_s$ and $\sigma' = 0.7T_r$. Thanks to these modifications, the extracted MSIS solution from the traceability game has Euclidean norm at most 2^{61} which is less than q.

With the given parameters, we obtain a group signature of size around 203KB which is around a factor of three improvement over [dPLS18].

q	d	l	κ	λ	α	μ
$\approx 2^{64}$	128	64	20	24	24	127

Table 2. Group signature parameters.

Acknowledgements

We would like to thank anonymous reviewers for the useful feedback. This work was supported by the SNSF ERC Transfer Grant CRETP2-166734 FELICITY and the EU H2020 ERC Project 101002845 PLAZA.

References

ABB10. Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In *EUROCRYPT*, pages 553–572, 2010.

ALS20. Thomas Attema, Vadim Lyubashevsky, and Gregor Seiler. Practical product proofs for lattice commitments. In *CRYPTO (2)*, volume 12171 of *Lecture Notes in Computer Science*, pages 470–499. Springer, 2020.

Ban93. Wojciech Banaszczyk. New bounds in some transference theorems in the geometry of numbers. *Mathematische Annalen*, 296(1):625–635, Dec 1993.

BCN18. Cecilia Boschini, Jan Camenisch, and Gregory Neven. Floppy-sized group signatures from lattices. In *ACNS*, volume 10892 of *Lecture Notes in Computer Science*, pages 163–182. Springer, 2018.

BDK$^+$21. Ward Beullens, Samuel Dobson, Shuichi Katsumata, Yi-Fu Lai, and Federico Pintore. Group signatures and more from isogenies and lattices: Generic, simple, and efficient. Cryptology ePrint Archive, Report 2021/1366, 2021. `https://ia.cr/2021/1366`.

BDL$^+$18. Carsten Baum, Ivan Damgård, Vadim Lyubashevsky, Sabine Oechsner, and Chris Peikert. More efficient commitments from structured lattice assumptions. In *SCN*, pages 368–385, 2018.

BLS19. Jonathan Bootle, Vadim Lyubashevsky, and Gregor Seiler. Algebraic techniques for short (er) exact lattice-based zero-knowledge proofs. In *CRYPTO*, pages 176–202. Springer, 2019.

CS97. Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups (extended abstract). In Burton S. Kaliski Jr., editor, *Advances in Cryptology - CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 410–424. Springer, 1997.

DDLL13. Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal gaussians. In *CRYPTO (1)*, pages 40–56, 2013.

DKL$^+$18. Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-dilithium: A lattice-based digital signature scheme. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(1):238–268, 2018.

dPLS18. Rafaël del Pino, Vadim Lyubashevsky, and Gregor Seiler. Lattice-based group signatures and zero-knowledge proofs of automorphism stability. In *ACM Conference on Computer and Communications Security*, pages 574–591. ACM, 2018.

ENS20. Muhammed F. Esgin, Ngoc Khanh Nguyen, and Gregor Seiler. Practical exact proofs from lattices: New techniques to exploit fully-splitting rings. In *ASIACRYPT (2)*, pages 259–288, 2020.

ESLL19. Muhammed F. Esgin, Ron Steinfeld, Joseph K. Liu, and Dongxi Liu. Lattice-based zero-knowledge proofs: New techniques for shorter and faster constructions and applications. In *CRYPTO (1)*, volume 11692 of *Lecture Notes in Computer Science*, pages 115–146. Springer, 2019.

ESS$^+$19. Muhammed F. Esgin, Ron Steinfeld, Amin Sakzad, Joseph K. Liu, and Dongxi Liu. Short lattice-based one-out-of-many proofs and applications to ring signatures. In *ACNS*, volume 11464 of *Lecture Notes in Computer Science*, pages 67–88. Springer, 2019.

ESZ21. Muhammed F. Esgin, Ron Steinfeld, and Raymond K. Zhao. Matrict+: More efficient post-quantum private blockchain payments. Cryptology ePrint Archive, Report 2021/545, 2021. `https://ia.cr/2021/545`.

EZS$^+$19. Muhammed F. Esgin, Raymond K. Zhao, Ron Steinfeld, Joseph K. Liu, and Dongxi Liu. Matrict: Efficient, scalable and post-quantum blockchain confidential transactions protocol. In *CCS*, pages 567–584. ACM, 2019.

FS86. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, pages 186–194, 1986.

GKV10. S. Dov Gordon, Jonathan Katz, and Vinod Vaikuntanathan. A group signature scheme from lattice assumptions. In *Advances in Cryptology - ASIACRYPT 2010*, pages 395–412, 2010.

GSW13. Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO (1)*, volume 8042 of *Lecture Notes in Computer Science*, pages 75–92. Springer, 2013.

LLNW16. Benoît Libert, San Ling, Khoa Nguyen, and Huaxiong Wang. Zero-knowledge arguments for lattice-based accumulators: Logarithmic-size ring signatures and group signatures without trapdoors. In *EUROCRYPT (2)*, volume 9666 of *Lecture Notes in Computer Science*, pages 1–31. Springer, 2016.

LN17. Vadim Lyubashevsky and Gregory Neven. One-shot verifiable encryption from lattices. In *EUROCRYPT*, 2017.

LNS20. Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. Practical lattice-based zero-knowledge proofs for integer relations. In *CCS*, pages 1051–1070. ACM, 2020.

LNS21a. Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. Shorter lattice-based zero-knowledge proofs via one-time commitments. In *Public Key Cryptography (1)*, volume 12710 of *Lecture Notes in Computer Science*, pages 215–241. Springer, 2021.

LNS21b. Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. Smile: Set membership from ideal lattices with applications to ring signatures and confidential transactions. Cryptology ePrint Archive, Report 2021/564, 2021. `https://eprint.iacr.org/2021/564`.

LS15. Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Designs, Codes and Cryptography*, 75(3):565–599, 2015.

Lyu09. Vadim Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In *ASIACRYPT*, pages 598–616, 2009.

Lyu12. Vadim Lyubashevsky. Lattice signatures without trapdoors. In *EUROCRYPT*, pages 738–755, 2012.

MP12. Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT*, pages 700–718, 2012.

OO98. Kazuo Ohta and Tatsuaki Okamoto. On concrete security treatment of signatures derived from identification. In *Annual International Cryptology Conference*, pages 354–369. Springer, 1998.

Reg09. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6), 2009.

YAZ+19. Rupeng Yang, Man Ho Au, Zhenfei Zhang, Qiuliang Xu, Zuoxia Yu, and William Whyte. Efficient lattice-based zero-knowledge arguments with standard soundness: Construction and applications. In *CRYPTO (1)*, volume 11692 of *Lecture Notes in Computer Science*, pages 147–175. Springer, 2019.

A Additional Background

A.1 Proving Linear and Multiplicative Relations

In this paper we will use techniques developed in [ALS20, ENS20, LNS21a] to prove certain linear and multiplicative relations. For completeness, we will briefly recall the protocols to prove such relations and refer to [ALS20, ENS20, LNS20, LNS21a] for more details.

Opening proof [ALS20]. The key component of proving linear and multiplicative relations is the proof of knowledge of a committed message. Concretely, the prover $\mathcal{P}$ wants to convince the verifier $\mathcal{V}$, which has a BDLOP commitment $(\boldsymbol{t}_0, t_1)$, that $\mathcal{P}$ knows a short randomness vector $\boldsymbol{r}$ over $\mathcal{R}_q$ and a message $m \in \mathcal{R}_q$ such that $\boldsymbol{A}_0\boldsymbol{r} = \boldsymbol{t}_0$ and $\boldsymbol{a}_1^T\boldsymbol{r} + m = t_1$.

The proof goes as follows. Prover $\mathcal{P}$ starts by sampling a vector $\boldsymbol{y}$ from discrete Gaussian and computing $\boldsymbol{w} = \boldsymbol{A}_0\boldsymbol{y}$. Then, it sends $\boldsymbol{w}$ to the verifier. After receiving the challenge $c \leftarrow \mathcal{C}$ from $\mathcal{V}$, the prover computes $\boldsymbol{z} = \boldsymbol{y} + c\boldsymbol{r}$ and applies rejection sampling. If it does not abort, $\mathcal{P}$ sends $\vec{z}$. Finally, the verifier checks that coefficients of $\boldsymbol{z}$ are small and $\boldsymbol{A}_0\boldsymbol{z} \overset{?}{=} \boldsymbol{w} + c\boldsymbol{t}_0$.

Product proof [ALS20]. In our group signature, we will want to prove that a committed polynomial m has binary NTT coefficients, i.e. $\mathsf{NTT}(m) \in \{0,1\}^l \subset \mathcal{M}_q^l$. This is equivalent to proving $m(m-1) = 0$ over $\mathcal{R}_q$.

Let $(\boldsymbol{t}_0, t_1)$ be a commitment to m, i.e. $\boldsymbol{A}_0\boldsymbol{r} = \boldsymbol{t}_0$ and $\boldsymbol{a}_1^T\boldsymbol{r} + m = t_1$ for a short randomness vector $\boldsymbol{r}$. Let us adapt the notation from the opening proof above and set $f := \boldsymbol{a}_1^T\boldsymbol{z} - ct_1$. Note that f can be computed by the verifier. Since $\boldsymbol{z} = \boldsymbol{y} + c\boldsymbol{r}$ we have that $f = \boldsymbol{a}_1^T\boldsymbol{y} - cm$. Thus,

$$f(f+c) = v_0 + v_1 c + m(m-1)c^2$$

for $v_0 := (\boldsymbol{a}_1^T\boldsymbol{y})^2$ and $v_1 = \boldsymbol{a}_1^T\boldsymbol{y}(1-2m)$. Hence, the goal is to prove that the quadratic coefficient of $f(f-1)$ vanishes. We proceed as follows.

The prover starts by generating vectors $\boldsymbol{r}, \boldsymbol{y}$ and the BDLOP commitment $\boldsymbol{t} = (\boldsymbol{t}_0, t_1, t_2)$ where

$$\begin{cases} \boldsymbol{A}_0 \boldsymbol{r} = \boldsymbol{t}_0 \\ \boldsymbol{a}_1^T \boldsymbol{r} + m = t_1 \\ \boldsymbol{a}_2^T \boldsymbol{r} + v_1 = t_2. \end{cases}$$

As in the opening proof, it computes $\boldsymbol{w} = \boldsymbol{A}_0 \boldsymbol{y}$. Then, it outputs $(t, \boldsymbol{w}, v_0')$ where $v_0' = v_0 + +\boldsymbol{a}_2^T \boldsymbol{y}$. After getting a challenge $c \leftarrow \mathcal{C}$ from the verifier, $\mathcal{P}$ computes $\boldsymbol{z} = \boldsymbol{y} + c\boldsymbol{r}$ and applies rejection sampling. If it does not fail, the prover outputs $\boldsymbol{z}$. Then, $\mathcal{V}$ checks that $\boldsymbol{z}$ has small coefficients and $\boldsymbol{A}_0 \boldsymbol{z} \overset{?}{=} \boldsymbol{w} + c\boldsymbol{t}_0$. Next, it computes

$$f = \boldsymbol{a}_1^T \boldsymbol{z} - ct_1 \text{ and } f_2 = \boldsymbol{a}_2^T \boldsymbol{z} - ct_2.$$

Eventually, $\mathcal{P}$ checks whether:

$$f(f + c) + f_2 \overset{?}{=} v_0'.$$

As described in [ALS20], proving additional quadratic relations does not affect the proof size.

Linear proof [ENS20]. Another building block, which will be used in our protocols, is the proof of a linear relation. Namely, let $(\boldsymbol{t}_0, t_1, t_2)$ be a BDLOP commitment defined below:

$$\begin{cases} \boldsymbol{A}_0 \boldsymbol{r} = \boldsymbol{t}_0 \\ \boldsymbol{a}_1^T \boldsymbol{r} + s = t_1 \\ \boldsymbol{a}_2^T \boldsymbol{r} + u = t_2. \end{cases} .$$

Suppose the prover commits to polynomials s and u such that $\vec{s} = \mathsf{NTT}(s) \in \mathbb{Z}_q^l$ and $\vec{u} = \mathsf{NTT}(u)\, \mathbb{Z}_q^l$ have integers coefficients and wants to prove that

$$A\vec{s} = \vec{u}$$

for a public matrix A over $\mathbb{Z}_q$. The idea is to ask the verifier for a random challenge $\vec{\gamma} \leftarrow \mathcal{M}_q^l$ and then prove that

$$\langle A\vec{s} - \vec{u}, \vec{\gamma} \rangle = 0.$$

Recall that the inner product here is the operation defined in Section 2.2. Now, by Lemma 2.1 and the fact that $\vec{s} \in \mathbb{Z}_q$ we have

$$\begin{aligned} \langle A\vec{s} - \vec{u}, \vec{\gamma} \rangle &= \langle A\vec{s}, \vec{\gamma} \rangle - \langle \vec{u}, \vec{\gamma} \rangle \\ &= \langle \vec{s}, A^T \vec{\gamma} \rangle - \langle \vec{u}, \vec{\gamma} \rangle \\ &= \sum_{i=0}^{l-1} f_i \end{aligned}$$

where $\vec{f} := \vec{s} \circ (A^T \vec{\gamma}) - \vec{u} \circ \vec{\gamma}$. In other words, the sum of coefficients of $\vec{f}$ is equal to $\langle A\vec{s} - \vec{u}, \vec{\gamma} \rangle$. Next, we apply Lemma 2.2 to conclude that the first d/l coefficients of $\check{f} := \mathsf{NTT}^{-1}\left(\vec{f}\right) \in \mathcal{R}_q$ satisfy:

$$\sum_{i=0}^{d/l-1} \check{f}_i = \frac{\langle A\vec{s} - \vec{u}, \vec{\gamma} \rangle}{l}.$$

Hence, the goal is to prove that the first d/l coefficients of $\check{f}$ are all zeroes. Note that given commitments to polynomials s and u, the verifier can manually construct a commitment to $\check{f}$ as follows:

$$t_f := t_1 \mathsf{NTT}^{-1}\left(A^T\vec{\gamma}\right) - t_2\mathsf{NTT}^{-1}\left(\vec{\gamma}\right) = \left(\mathsf{NTT}^{-1}\left(A^T\vec{\gamma}\right)\boldsymbol{a}_1 - \mathsf{NTT}^{-1}\left(\vec{\gamma}\right)\boldsymbol{a}_2\right)^T \boldsymbol{r} + \check{f}.$$

In order to prove $\check{f}_0 = \ldots = \check{f}_{d/l-1} = 0$, at the beginning $\mathcal{P}$ commits to a uniformly random polynomial $g \in \mathcal{R}_q$ that also has first d/l coefficients equal to zeroes, i.e. $t_3 = \boldsymbol{a}_3^T\boldsymbol{r} + g$. After getting a challenge $\vec{\gamma}$ from the verifier, $\mathcal{P}$ computes $h = g + \check{f}$ and sends h. Then, $\mathcal{V}$ checks whether $h_0, h_1, \ldots, h_{d/l-1}$ are all zeroes. Finally, we need to prove that the equation $h = g + \check{f}$ holds. We do it by proving that $t_3 + t_f - h$ is a commitment to zero.

We are now ready to sketch out the protocol. First, the prover generates $\boldsymbol{r}, \boldsymbol{y}$ as before and computes $\boldsymbol{t} = (\boldsymbol{t}_0, t_1, t_2, t_3)$ defined above. As usual, it computes $\boldsymbol{w} = \boldsymbol{A}_0\boldsymbol{y}$. Then, it sends $(\boldsymbol{t}, \boldsymbol{w})$ to the verifier. Next, given a challenge $\vec{\gamma} \leftarrow \mathcal{M}_q^l$ from $\mathcal{V}$, the prover computes $h := g + \check{f}$ and w' defined as

$$w' := \left(\boldsymbol{a}_4 + \mathsf{NTT}^{-1}\left(A^T\vec{\gamma}\right)\boldsymbol{a}_1 - \mathsf{NTT}^{-1}\left(\vec{\gamma}\right)\boldsymbol{a}_2\right)^T \boldsymbol{y}$$

Then, $\mathcal{P}$ outputs (h, w'). Further, $\mathcal{V}$ outputs the challenge $c \leftarrow \mathcal{C}$. Eventually, $\mathcal{P}$ calculates $\boldsymbol{z} = \boldsymbol{y}+c\boldsymbol{r}$ and runs rejection sampling as before. After obtaining $\boldsymbol{z}$, the verifier checks that coefficients of $\boldsymbol{z}$ are small, $\boldsymbol{A}_0\boldsymbol{z} \overset{?}{=} \boldsymbol{w} + c\boldsymbol{t}_0$ as in the opening proof. Then, it also verifier whether the first d/l coefficents of h and indeed zeroes as well as

$$\left(\boldsymbol{a}_4 + \mathsf{NTT}^{-1}\left(A^T\vec{\gamma}\right)\boldsymbol{a}_1 - \mathsf{NTT}^{-1}\left(\vec{\gamma}\right)\boldsymbol{a}_2\right)^T \boldsymbol{z} \overset{?}{=} w' + c\left(t_3 + t_f - h\right).$$

Note that if $\mathcal{R}_q$ does not split fully, then in order to use Lemma 2.1 in the soundness argument, we need to additionally prove that $\vec{s} = \mathsf{NTT}(s)$ is a vector over $\mathbb{Z}_q$ (and not just $\mathcal{M}_q$). In our construction, this will be combined with the binary proof described above. Consequently, we have $\vec{s} \in \{0,1\}^l \subset \mathbb{Z}_q^l$.

Finally, we highlight that having additional linear relations does not affect the total proof size.

B Security proofs

Proof (Theorem 4.3).
Anonymity.

Let $\mathcal{A}$ be an adversary to the anonymity game G_0 defined as follows.

G_0 : The adversary has access to the secrets keys $(\mathsf{sk}_{m_1}, \mathsf{sk}_{m_2}, \mathsf{sk}_{m_3}, \ldots)$ for all identities in $\mathcal{I}$. Then, $\mathcal{A}$ choses a message M and two identities which we write m_1, m_2. A challenger samples a random bit $b \in \{0,1\}$, computes $\mathsf{GSign}(M, \mathsf{sk}_{m_b})$ and returns the signature to $\mathcal{A}$. The adversary $\mathcal{A}$ wins if he outputs b.

We define the following game.

G_1 : This game is the same as G_0 except, instead of honestly computing the signature of M with m, the challenger runs $\mathcal{S}$, the simulator from the zero-knowledge property of π and returns it to $\mathcal{A}$.

G_2 : This game is the same as G_1 except, instead of honestly computing the signature of M with m', the challenger runs $\mathcal{S}$, the simulator from the zero-knowledge property of π and returns it to $\mathcal{A}$.

From the zero-knowledge property of π and Lemma 2.8, we have $|\epsilon_{\mathcal{A}}^{G_1} - \epsilon_{\mathcal{A}}^{G_0}| \leqslant 2^{-100}/M + \epsilon_{\mathcal{A}}^{\mathsf{MLWE}}|\epsilon_{\mathcal{A}}^{G_2} - \epsilon_{\mathcal{A}}^{G_1}| \leqslant 2^{-100}/M + \epsilon_{\mathcal{A}}^{\mathsf{MLWE}}$. The advantage of $\mathcal{A}$ in G_2 is 0 since the distributions are independent of the secrets, from which we conclude $\epsilon_{\mathcal{A}}^{G_0} \leqslant 2^{-99}/M + 2\epsilon_{\mathcal{A}}^{\mathsf{MLWE}}$.

Traceability. This is a direct consequence of plugging together Lemmas 4.1 and 4.2.

Proof (Lemma 4.1). We define a sequence of games.

G_0 : This game is the traceability game. The challenger $\mathcal{B}$ runs KeyGen honestly, thus generates a public matrix $[\mathbf{A}|\mathbf{B}|\mathbf{B}']$ where $\mathbf{B} = \mathbf{AR}$. The adversary can query up to $Q_s \leqslant |\mathcal{I}|$ secret keys for any identity m of his choice, to which the challenger returns honestly generated secret vectors (first sampling $\mathbf{s}_3 \leftarrow D_\sigma^{3d\alpha}$ and then using the $m\mathbf{G}$-trapdoor $\mathbf{R}$). If $m = 0$, $\mathbf{R}$ is not a valid trapdoor anymore and $\mathcal{B}$ returns $(\mathbf{s}_1^{\mathsf{gm}}, \mathbf{s}_2^{\mathsf{gm}}, \mathbf{s}_3^{\mathsf{gm}})$. The adversary can make up to h_1 (respectively h_2) queries to the first (respectively the second) random oracle. Finally, the adversary can query up to Q signatures for the identity and message of his choice, to which $\mathcal{B}$ answers honestly.

The goal of $\mathcal{A}$ is to produce a forgery that 1) passes verification, and 2) such that the decryption of this signature is an identity that was not queried by $\mathcal{A}$.

G_1 : This game is the same as the previous one, except $\mathcal{B}$ uses the simulator $\mathcal{S}$ from π to answer signature queries from $\mathcal{A}$. Distinguishing this game to G_0 is the zero-knowledge property of π.

G_2 : This game is the same as the previous one, except $\mathbf{B}'$ is replaced by $\mathbf{B}'_2 = \mathbf{B}' - \mathbf{G}$. This game is statistically indistinguishable from G_1.

G_3 : This game is the same as the previous one, except $\mathbf{B}'$ is replaced by $\mathbf{AR}'$ (hence we have $\mathbf{B}'_3 = \mathbf{AR} - \mathbf{G}$), where $\mathbf{R}'$ is distributed as $\mathbf{R}$. This game is computationally indistinguishable from G_2 under $\mathsf{MLWE}_{\alpha,\alpha,S_\mu}$.

G_4 : This game is the same as the previous one, except when $\mathcal{B}$ is queried a secret key by $\mathcal{A}$, he first samples $\mathbf{s}_2 \leftarrow D_\sigma^{3d\alpha}$ and then uses the $\mathbf{G}$-trapdoor $\mathbf{R}$ to sample $\mathbf{s}_1$ and $\mathbf{s}_2$. This game is statistically indistinguishable from G_3.

G_5 : This game is the same as the previous one, except $\mathbf{B}$ is generated uniformly random. This game is computationally indistinguishable from G_4 under $\mathsf{MLWE}_{\alpha,\alpha,S_\mu}$.

G_6 : This game is the same as the previous one, except $\mathbf{B}$ is replaced with $\mathbf{B}_2 = \mathbf{B} - m^*\mathbf{G}$ for a uniformly random identity $m^* \leftarrow \mathcal{I}$. This game is statistically indistinguishable from G_5.

G_7 : This game is the same as the previous one, except $\mathbf{B}$ is generated as $\mathbf{AR}$, thus $\mathbf{B}_2$ is replaced with $\mathbf{B}_3 = \mathbf{AR} - m^*\mathbf{G}$. This game is computationally indistinguishable from G_6 under $\mathsf{MLWE}_{\alpha,\alpha,S_\mu}$.

G_8 : This game is the same as the previous one, except $(\mathbf{s}_1^{\mathsf{gm}}, \mathbf{s}_2^{\mathsf{gm}}, \mathbf{s}_3^{\mathsf{gm}})$ is sampled as the secret key to the identity m^*. When $\mathcal{B}$ is queried a secret key for some identity $m \neq m^*$, $\mathcal{B}$ samples first $\mathbf{s}_3 \leftarrow \mathcal{D}_\sigma^{3d\alpha}$, and then uses the $(m - m^*)\mathbf{G}$-trapdoor $\mathbf{R}$ to sample $\mathbf{s}_1, \mathbf{s}_2$. Notice that if $m = m^*$, $\mathbf{R}$ is not a valid trapdoor thus $\mathcal{B}$ returns $(\mathbf{s}_1^{\mathsf{gm}}, \mathbf{s}_2^{\mathsf{gm}}, \mathbf{s}_3^{\mathsf{gm}})$. This game is statistically indistinguishable from G_7.

G_9 : This game is the same as the previous one, except $\mathbf{AR}'$ is replaced with a uniformly random random matrix $\mathbf{U}$, thus $\mathbf{B}'_3$ is replaced with $\mathbf{B}'_4 = \mathbf{U} - \mathbf{G}$. This game is computationally indistinguishable from G_8 under $\mathsf{MLWE}_{\alpha,\alpha,S_\mu}$.

G_{10} : This game is the same as the previous one, except $\mathbf{B}'_4$ is replaced with a $\mathbf{B}'_5 = \mathbf{B}'_4 + \mathbf{G}$, thus $\mathbf{B}'_4$ is replaced with $\mathbf{B}'_5 = \mathbf{U}$. This game is statistically indistinguishable from G_9.

G_{11} : This game is the same as the previous one, except $\mathbf{B}_5'$ is replaced with $\mathbf{AR}'$, where $\mathbf{R}'$ is distributed as $\mathbf{R}$. This game is computationally indistinguishable from G_{10} under $\mathsf{MLWE}_{\alpha,\alpha,S_\mu}$ and we refer to this game as trace^*.

The game G_{11} is the trace^* game, and the result follows from summing the advantages.