Improved Zero-Knowledge Proofs for Lattice Encryption Schemes, Linking Classical and Lattice Primitives, and Applications

Abstract. It now seems probable that a quantum computer breaking non-trivial instances of the RSA and discrete logarithm problem will be built in the not-too-distant future. Despite this, for many applications such as electronic voting or anonymous authentication where simple encryption and signature schemes are not sufficient, there are currently no practical protocols that one could use to defend against such quantum adversaries. One might think that such protection is not yet needed, but this creates vulnerabilities against an adversary who collects data today and is then able to break all the privacy and anonymity properties of current protocols using a quantum computer in the future.

In the absence of efficient lattice schemes for advanced protocols, in this work we begin the investigation of "hybrid" schemes that try to maintain the efficiency of classical protocols, but with the forward-security of lattice-based ones. Our main result is a proposal of one such protocol – a group signature scheme that combines the efficiency of today's number theoretic schemes, while using lattice encryption to guarantee privacy to its members even after quantum computers are developed. As intermediate steps, we provide a new and more efficient protocol for proving the validity of a lattice-based ciphertexts, and then extend this to a protocol that proves that a classical commitment and a lattice-based encryption contain the same secret. We believe that these two protocols are also interesting in their own right. From these two primitives, our group signature construction then follows a generic folklore approach, for which, to the best of our knowledge, we provide the first formal security proof.

Keywords: Verifiable Encryption, Group Signatures, Zero-Knowledge Proofs for Lattices.

1 Introduction

With quantum computing looming in the (possibly) near future, there has been a push over the last few years towards constructing efficient primitives based on the hardness of lattice problems, which are believed to be resilient against quantum algorithms. Unfortunately, except for encryption and digital signatures, lattice schemes seem to be a lot less practical (especially space-wise) than their classical counterparts. And so if one wishes to use quantum-secure protocols, his choices are either to keep using the current protocols until they are no longer secure, or to migrate to the less-efficient lattice protocols.

We believe that the decision of whether to switch or not should actually depend on the primitive itself. For example, if one were using a statistically-hiding commitment scheme to commit to sensitive data, then quantum computing does not really pose any threat to the data. Once quantum computers are deployed, the commitment will of course be invalid (since it will no longer be binding), but the message will still be hidden. On the other hand, a number theoretic commitment scheme that is only computationally-hiding will not protect the message contents against quantum computing. In short, we believe that in schemes where quantum computers do not pose a threat to privacy, migrating to quantum-secure constructions is not as urgent.

Some more involved schemes are comprised of both types of primitives – ones that would leak private data and others that information-theoretically hide it. In the absence of efficient lattice constructions for such schemes, our proposal is to construct them using a "hybrid" approach between classical and lattice cryptography. The idea is that in a "pre-quantum" world, they would be more efficient than pure lattice-based schemes. And once quantum computers are built, all the private information contained in the ciphertexts or proofs that are generated while using the schemes would still remain secret.

A good example of a scheme that would benefit from such a construction is a group signature. The "raison d'être" of a group signature scheme is to hide the identity of a user who is signing a message in the name of the group - and so it is imperative that the user's identity remains hidden even after quantum computers come about and the signature scheme can no longer be used. In a group signature scheme, while every group member is able to anonymously sign, the master group authority still has a way to unmask a user (in case of disputes or misbehavior, for example). A high level overview for how such a scheme is realized is for a group member to produce a signature using a key he was given by the master authority, then encrypt his identity, and then produce a zero-knowledge proof linking the ciphertext to the key used in the signature (i.e., his identity). A verifier is able to ascertain that the signature and the zero-knowledge proof are valid, but does not learn anything about the identity of the user. The master authority, however, can simply decrypt the ciphertext containing the user's identity.

It would be preferable, therefore, if the encryptions of identities were lattice-based, whereas everything else in the scheme would information-theoretically hide the identity of the user. In this way, assuming that lattice encryption schemes are secure against quantum attacks, a signer's identity would remain hidden.

The main result of this work is a realization of such "hybrid" schemes. The schemes themselves are comprised of several components, which we believe are interesting in their own rights. The first is a zero-knowledge proof of plaintext knowledge of lattice encryption schemes, and the second is a technique to "link" classical commitments to lattice encryptions by proving in zero-knowledge that they contain the same message.

1.1 Improved Proofs of Plaintext Knowledge for Lattice Schemes

In a zero knowledge proof of plaintext knowledge, the encryptor wants to prove in zero knowledge that the ciphertext is of the correct form and that he knows the message. Efficient constructions of these primitives are known based on number-theoretic hardness assumptions such as discrete log, strong RSA, etc..

Ciphertexts in lattice-based encryption schemes generally have the form $t = Ae \bmod q$, where A is some public matrix and e is the unique vector with small coefficients that satisfies the equation (in this general example, we are lumping the message with e). A proof that t is a valid ciphertext (and also a proof of plaintext knowledge), therefore involves proving that one knows short a short e such that $Ae = t$. It is currently known how to accomplish this task in two ways. The first uses a "Stern-type" protocol [35] in which every run has soundness error $2/3$ [24]. It does not seem possible to improve this protocol since some steps in it are inherently combinatorial and non-algebraic.

A second possible approach is to use the Fiat-Shamir approach for lattices using rejection sampling introduced in [25,27]. But while the latter leads to fairly efficient Fiat-Shamir signatures, there are some barriers to obtaining a proof of knowledge. What one is able to extract from a prover are short vectors r', z' such that $Ae' = tc$ for some integer c, which implies that $Ae'c^{-1} = t$. Unfortunately, this does not imply that $e'c^{-1}$ is short unless $c = \pm 1$. This is the main way in which lattice-based Fiat-Shamir proofs differ from traditional schemes like the discrete-log based Schnorr protocol. In the latter, it is enough to extract any discrete log, whereas in lattice protocols, one must extract a *short* vector. Thus, the obvious approach at Fiat-Shamir proofs of knowledge for lattices (i.e. using challenge vectors from $\{0, 1\}$) also leads to protocols with soundness error $1/2$.

Things do not improve for lattice-based proofs of knowledge even if one considers ideal lattices. Even if A and e are a matrix and a vector of polynomials in the ring $\mathbb{Z}_q[X]/(X^n + 1)$, and c is now a polynomial (as in the Ring-LWE based Fiat-Shamir schemes in [26,27]), then one can again extract only a short e' such that $Ae'c^{-1} = t$. In this case, not only is c^{-1} not necessarily short, but it does not even necessarily exist (since the polynomial $X^n + 1$ can factor into up to n terms.) At this point, we are not aware of any techniques that reduce the soundness error of protocols that prove plaintext knowledge of lattice encryptions, except by (parallel) repetition.

In a recent work, Damgard et al. [18] gave an improved *amortized* proof of plaintext knowledge for LWE encryptions. Their very nice protocol allows to prove knowledge of $O(k)$ plaintexts (where k is the security parameter) in essentially the same time as just one plaintext. But it seems that the ideas behind the protocol do not reduce the time requirement for proving just one plaintext, nor do they apply to Ring-LWE based encryption schemes. In particular, Ring-LWE based schemes are able to encrypt $O(n)$ plaintext bits into one (or two) polynomial, which is often all that is needed. Yet, the techniques in [18] do not seem to be helpful here. The reason is that the challenge matrix required in [18] needs to be of a particular form and cannot simply be a ring element in $\mathbb{Z}_q[X]/(X^n + 1)$.

In this paper, we first show that one can reduce the soundness error of lattice-based zero-knowledge proofs of knowledge for ciphertext validity from $1/2$ to $1/(2n)$, which in practice decreases the number of required iterations of the protocol by a factor of approximately 10. Interestingly, our techniques only work for ideal lattices, and we do not know how to adapt them to general ones. The key observation is that when working over the ring $\mathbb{Z}[X]/(X^n + 1)$, the quantity $2/(X^i - X^j)$ for all $0 \leq i \neq j < n$ is a polynomial with coefficients in $\{-1, 0, 1\}$ (this is proved in Section 3.1).

This immediately allows us to prove that, given A and t, we know a vector of short polynomials e such that $Ae = 2t$. While this is not quite the same as proving that $Ae = t$, it is good enough for most applications, since it still allows us to prove knowledge of the plaintext. This result immediately gives improvements in all schemes

that require such a proof of knowledge for Ring-LWE based encryption schemes such as the ring-version of the "dual" encryption scheme [20], the "two element" scheme of Lyubashevsky et al. [28], and NTRU [22,34].

1.2 Linking Lattice and Classical Commitments

A main step in constructing our "hybrid" schemes is to prove that two primitives, one based on classical cryptography and the other on lattices, are committing to the same message (and that the prover knows the message). In our application, we will use the perfectly hiding Pedersen commitment as the classical, and a Ring-LWE encryption scheme as the lattice-based primitive.

While Pedersen commitments and lattice-based encryption schemes work over rather different rings, we show that we can still perform operations "in parallel" on the two commitments. For example, if the message is $\mu_0, \mu_1, \ldots, \mu_{n-1}$, then it is encrypted in Ring-LWE schemes as the polynomial $\mu = \mu_0 + \mu_1 X + \ldots + \mu_{n-1} X^{n-1}$, and each μ_i is encrypted individually using a Pedersen commitment. We will then want to prove that a Ring-LWE encryption of μ encrypts the same thing as n Pedersen commitments of μ_i. Even though the two computations are performed over different rings, we show that by mimicking polynomial multiplications over a polynomial ring by appropriate additions and multiplications of coefficients in exponents, we can use our previously mentioned proof of plaintext knowledge to both prove knowledge of μ and show that the Pedersen commitments are committing to the coefficients μ_i. One of the reasons that we are able to provide such a proof is that the terms dealing with μ (and μ_i) in the proof of knowledge are worked on "over the integers" – that is, modular reduction never needs to be performed on these terms. We describe this protocol in detail in Section 4.

1.3 Applications to Group Signatures and Credentials

Group signatures [12] are schemes that allow members of a group to sign messages on behalf of the group without revealing their identity. In case of a dispute, the group manager can lift the signer's anonymity and reveal his identity. Currently known group signatures based on lattice assumptions should mainly be seen as proofs of concepts, rather than practically useful schemes. The schemes by Gordon et al. [21] and Camenisch et al. [10] have signature size linear in the number of group members. The scheme due to Laguillaumie et al. [23] performs much better asymptotically with signature sizes logarithmic in the number of group members, but, as the authors admit, instantiating it with practical parameters would lead to very large keys and signature sizes. This is in contrast to classical number-theory based solutions, where both the key and the signature size are constant for arbitrarily many group members.

As previously mentioned, quantum computers may have very different impacts for security and privacy: For controversial topics, privacy might be a long term concern, and a user might only be willing to use a scheme if he is convinced that his identity will remain secret, say, fifty years from now. The implications of signature forgeries being broken may be less significant, because one can simply stop accepting classical signatures as soon as quantum computers become practical.

Following this observation, we propose a "hybrid" group signature scheme, where unforgeability holds under classical assumptions, whereas privacy is proved under lattice-based ones. This allows us to combine the flexible tools that are available in the classical framework with the strong privacy guarantees of lattice problems. As a result, we obtain a group signature scheme that is private under quantum-resistant assumptions, where the key and the signature size are only logarithmic in the number of group members. For practical choices of parameters and realistic numbers of group members, the sizes will even be independent of the number of users. Furthermore, by basing our scheme on ring-LWE and not standard LWE, we partially solve an open problem stated in [23].

Our construction follows a generic approach that we believe is folklore, as it underlies several direct constructions in the literature [2,8] and a variant was described explicitly by Chase and Lysyanskaya [11]. When joining the group, a user obtains a certificate from the group manager that is a signature on his identity under the group manager's public key. To sign a message, the user now encrypts his identity under the manager's public encryption key, and then issues a signature proof of knowledge that he possesses a valid signature on the plaintext contained in this encryption. Our construction follows this general paradigm, but with some modifications to better fit the specifics of our proof of plaintext knowledge for lattice encryption. To the best of our knowledge, however, the construction was never proved secure, so our proof can be seen as a contribution of independent interest.

2 Preliminaries

2.1 Notation

We denote algorithms by sans-serif letters such as A, B.

If $\mathcal{S}$ is a set, we write $s \xleftarrow{\$} \mathcal{S}$ to denote that s was drawn uniformly at random from $\mathcal{S}$. Similarly, we write $a \xleftarrow{\$} \mathsf{A}$ if a was computed by a randomized algorithm A, and $d \xleftarrow{\$} D$ for a probability distribution D, if d was drawn according to D.

We write $\Pr[\mathcal{E} : \Omega]$ to denote the probability of event $\mathcal{E}$ over the probability space Ω. For instance, $\Pr[x = y : x, y \xleftarrow{\$} D]$ denotes the probability that $x = y$ if x, y were drawn according to a distribution D.

If v is a vector, we denote by $v_{\lll l}$ an anti-cyclic shift of a vector v by l positions, corresponding to a multiplication by X^l in $R_q = \mathbb{Z}_q[X]/(X^n + 1)$. That is, $v_{\lll l} = (v_0, \ldots, v_{n-1})_{\lll l} = (-v_{n-l}, \ldots, -v_{n-1}, v_0, \ldots, v_{n-l-1})$. Furthermore, we identify the vectors $(a_0, \ldots, a_{n-1})$ with the polynomial $a_0 + a_1 X + \cdots + a_{n-1} X^{n-1}$.

Throughout the paper, λ denotes the main security parameter and ε denotes the empty string.

2.2 Commitment Schemes and Pedersen Commitments

Informally, a commitment scheme is a tuple $(\mathsf{CSetup}, \mathsf{Commit}, \mathsf{COpen})$, where CSetup generates commitment parameters, which are then used to commit to a message m using Commit. A commitment cmt can then be verified using COpen. Informally, a commitment scheme needs to be binding and hiding. The former means that no cmt can be opened to two different messages, while the latter guarantees that cmt does not leak any information about the contained m. A formal definition can be found in Appendix A.1.

We next recap an efficient commitment scheme that was introduced by Pedersen [30]. Let therefore $\{\mathbb{G}(\lambda)\}_{\lambda \in \mathbb{N}}$ be a family of prime order groups such that the discrete logarithm problem is hard in $\mathbb{G}(\lambda)$ for security parameter λ. Let $\tilde{q} = \tilde{q}(\lambda)$ be the order of $\mathbb{G}(\lambda)$.

To avoid confusion in the following, all elements with order $\tilde{q}$ are denoted with a tilde in the following.

CSetup**.** This algorithm chooses $\tilde{g}, \tilde{h} \xleftarrow{\$} \mathbb{G}(\lambda)$ and outputs $cpars = (\tilde{g}, \tilde{h})$.

Commit**.** To commit to a message $m \in \mathcal{M} = \mathbb{Z}_{\tilde{q}}$, this algorithm first chooses $r \xleftarrow{\$} \mathbb{Z}_{\tilde{q}}$. It then outputs the pair $(\widetilde{cmt}, o) = (m\tilde{g} + r\tilde{h}, r)$.

COpen**.** Given a commitment $\widetilde{cmt}$, an opening o, a public key $cpars$ and a message m, this algorithm outputs `accept` if and only if $\widetilde{cmt} \stackrel{?}{=} m\tilde{g} + o\tilde{h}$.

Theorem 2.1. *Under the discrete logarithm assumption for $\mathbb{G}$, the given commitment scheme is perfectly hiding and computationally binding.*

2.3 Semantically Secure Encryption and NTRU

A semantically secure (or IND-CPA secure) encryption scheme is a tuple $(\mathsf{EncKG}, \mathsf{Enc}, \mathsf{Dec})$ of algorithms, where EncKG generates public/private key pair, Enc can be used to encrypt a message m under the public key, and the message can be recovered from the ciphertext by Dec using the secret key. Informally, while $\mathsf{Dec}(\mathsf{Enc}(m)) = m$ should always hold, only knowing the ciphertext and the public key should not leak any information about the contained message. A formal definition is given in Appendix A.2.

In our protocols we will be showing improved zero knowledge proofs of plaintext knowledge for lattice-based encryption schemes, as well as showing how to link messages being encrypted by these schemes to Pedersen commitments. Our improved proof of knowledge protocol will work for any Ring-LWE based scheme where the basic encryption operation consists of taking public key polynomial(s) a_i and computing the ciphertext(s) $b_i = a_i s + e_i$ where s and e_i are polynomials with small norms. Examples of such schemes include the ring-version of the "dual" encryption scheme [20], the "two element" scheme of Lyubashevsky et al. [28], and the NTRU encryption scheme [22,34].

For simplicity, in this paper we will only be working over the rings $R = \mathbb{Z}[X]/\langle X^n + 1\rangle$ and $R_q = R/qR$, for some prime q. Also for simplicity, we will use NTRU as our encryption scheme because its ciphertext has only one element and is therefore simpler to describe in protocols. The NTRU scheme was first proposed by Hoffstein et al. [22], and we will be using a modification of it due to Stehlé and Steinfeld [34].

Definition 2.2. *The* discrete Normal distribution *on $\mathbb{Z}^m$ centered at v with standard deviation σ is defined by the density function $D^m_{v,\sigma}(x) = \rho^m_{v,\sigma}(x)/\rho_\sigma(\mathbb{Z}^m)$, where $\rho^m_{v,\sigma}(x) = \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^m e^{-\frac{\|x-v\|^2}{2\sigma^2}}$ is the continuous normal distribution on $\mathbb{R}^m$ and $\rho_\sigma(\mathbb{Z}^m) = \sum_{z\in\mathbb{Z}^m} \rho^m_{0,\sigma}(z)$ is the scaling factor required to obtain a probability distribution.*

In abuse of notation we will sometimes write $u \xleftarrow{\$} D_{v,\sigma}$ instead of $u \xleftarrow{\$} D^n_{v,\sigma}$ for a polynomial $u \in R_q$ if there is no risk of confusion.

In the following, let p be a prime less than q and $\sigma, \alpha \in \mathbb{R}$.

Message space. The message space is given by $\mathcal{M} = \{y \in R : \|y\|_\infty < p\}$

KeyGen. Sample f', g from $D_{\mathbb{Z}^n,\sigma}$, set $f = pf' + 1$, and resample, if $f \bmod q$ or $g \bmod q$ are not invertible.

Output the public key $h = pg/f$ and the secret key f. Note here that h is invertible.

Encrypt. To encrypt a message $m \in \mathcal{M}$, set $s, e \xleftarrow{\$} D_\alpha$ and return the ciphertext $\mathsf{c} = hs + pe + m \in R_q$.

Decrypt. To decrypt c with secret key f, compute $\mathsf{c}' = f\mathsf{c} \in R_q$ and output $m' = \mathsf{c}' \mod p$.

If the value of σ is large enough (approximately $\tilde{\mathcal{O}}(n\sqrt{q})$), then g/f is uniformly random in R_q [34], and the security of the above scheme is based on the Ring-LWE problem. For smaller values of σ, however, the scheme is more efficient and can be based on the assumption that $h = g/f$ is indistinguishable from uniform. This type of assumption, while not based on any worst-case lattice problem, has been around since the introduction of the original NTRU scheme over fifteen years ago. Our protocol works for either instantiation.

To obtain group signatures, we will need our encryption scheme to additionally be a commitment scheme. In other words, there should not be more than one way to obtain the same ciphertext. For the NTRU encryption scheme, this will require that we work over a modulus q such that the polynomial X^n+1 splits into two irreducible polynomials of degree $n/2$.

Lemma 2.3. *Suppose that $q = 3 \bmod 8$ and let $\#S$,$\#E$, and $\#\mathcal{M}$ be the domain sizes of the parameters s, e, and m in the ciphertext $\mathsf{c} = hs + pe + m$. Additionally suppose that for all $m \in \mathcal{M}$, $\|m\|_\infty < p/2$. Then the probability that for a random h, there exists a ciphertext that can be obtained in two ways is at most $\frac{(2\#\mathcal{M}+1)\cdot(2\#S+1)\cdot(2\#E+1)}{q^{n/2}}$.*

The proof can be found in Appendix C.1. Note that the above lemma applies to NTRU public keys h that are uniformly random. If $h = pg/f$ is not random, then the ability to come up with two plaintexts for the same ciphertext would constitute a distinguisher for the assumed pseudorandomness of h.

2.4 Rejection Sampling

For a protocol to be zero-knowledge, the prover's responses must not depend on its secret inputs, as otherwise they could not be simulated. However, in our protocols, the prover's response will be from a discrete normal distribution which is shifted depending on the secret key. To correct for this, we employ rejection sampling [26,27], where a potential response is only output with a certain probability, and otherwise the protocol is aborted.

Informally, the following theorem states that for sufficiently large σ the rejection sampling procedure outputs results that are independent of the secret. The technique only requires a constant number of iterations before a value is output, and furthermore the output is also statistically close for every secret v with norm at most T. For concrete parameters we refer to the original work of Lyubashevsky [27, Theorem 4.6].

Theorem 2.4. *Let V be a subset of $\mathbb{Z}^\ell$ in which all elements have norms less than T, and let h be a probability distribution over V. Then, for any constant M, there exists a $\sigma = \tilde{\Theta}(T)$ such that the output distributions of the following algorithms A, F are statistically close:*

$$\mathsf{A} : v \xleftarrow{\$} h; \quad z \xleftarrow{\$} D^\ell_{v,\sigma}; \text{ output } (z, v) \text{ with probability } \min\left(D^\ell_{\sigma(z)}/(MD^\ell_{v,\sigma}(z)), 1\right)$$

$$\mathsf{F} : v \xleftarrow{\$} h; \quad z \xleftarrow{\$} D^\ell_{0,\sigma}; \text{ output } (z, v) \text{ with probability } 1/M$$

Moreover, the probability that A outputs something is exponentially close to that of F, i.e., $1/M$.

2.5 Zero-Knowledge Proofs and Σ-Protocols

On a high level, a zero-knowledge proof of knowledge (ZKPoK) is a two party protocol between a *prover* and a *verifier*, which allows the former to convince the latter that it knows some secret piece of information, without revealing anything about the secret apart from what the claim itself already reveals. For a formal definition we refer to Bellare and Goldreich [3].

The ZKPoKs constructed in this paper will be instantiations of the following definition, which is a straightforward generalization of the standard notion of Σ-protocols [13,15]:

Definition 2.5. *Let* (P, V) *be a two-party protocol, where* V *is PPT, and let* $\mathcal{R}, \mathcal{R}'$ *be a binary relation such that* $\mathcal{R} \subseteq \mathcal{R}'$. *Then* (P, V) *is called a* Σ'-protocol *for* $\mathcal{R}, \mathcal{R}'$ *with challenge set* $\mathcal{C}$, *public input* y *and private input* w, *if and only if it satisfies the following conditions:*

- Three-move form*: The protocol is of the following form: The prover* P *computes a commitment* $\mathtt{t}$ *and sends it to* V. *The verifier* V *then draws a challenge* $\mathtt{c} \xleftarrow{\$} \mathcal{C}$ *and sends it to* P. *The prover sends a response* $\mathtt{s}$ *to the verifier. Depending on the protocol transcript* $(\mathtt{t}, \mathtt{c}, \mathtt{s})$*, the verifier finally accepts or rejects the proof.*
  The protocol transcript $(\mathtt{t}, \mathtt{c}, \mathtt{s})$ *is called* accepting*, if the verifier accepts the protocol run.*
- Completeness*: Whenever* $(y, w) \in \mathcal{R}$, *the verifier* V *accepts with probability at least* $1 - \alpha$.
- Special soundness*: There exists a PPT algorithm* E *(the* knowledge extractor*) which takes two accepting transcripts* $(\mathtt{t}, \mathtt{c}', \mathtt{s}'), (t, \mathtt{c}'', \mathtt{s}'')$ *satisfying* $\mathtt{c}' \neq \mathtt{c}''$ *as inputs, and outputs* w' *such that* $(y, w') \in \mathcal{R}'$.
- Special honest-verifier zero-knowledge (HVZK)*: There exists a PPT algorithm* S *(the* simulator*) taking* $y \in L(\mathcal{R})$ *and* $\mathtt{c} \in \mathcal{C}$ *as inputs, that outputs* $(\mathtt{t}, \mathtt{s})$ *so that the triple* $(\mathtt{t}, \mathtt{c}, \mathtt{s})$ *is indistinguishable from an accepting protocol transcript generated by a real protocol run.*

This definition differs from the standard definition of Σ-protocols in two ways. First, we allow the honest prover to fail in at most an α-fraction of all protocol runs, whereas the standard definition requires perfect completeness, i.e., $\alpha = 0$. However, this relaxation is crucial in our construction that is based on rejection sampling [26,27], where the honest prover sometimes has to abort the protocol to achieve zero-knowledge. Second, we introduce a second relation $\mathcal{R}' \supseteq \mathcal{R}$, such that provers knowing a witness in $\mathcal{R}$ are guaranteed privacy, but the verifier is only ensured that the prover knows a witness for $\mathcal{R}'$. This has already been used in [1] and informally also in, e.g., [19,16]. If the *soundness gap* between $\mathcal{R}$ and $\mathcal{R}'$ is sufficiently small, the implied security guarantees are often enough for higher-level applications.

We want to stress that previous results showing that a Σ-protocol is always also an honest-verifier ZKPoK with knowledge error $1/|\mathcal{C}|$ directly carry over to the modified definition whenever $1 - \alpha > 1/|\mathcal{C}|$. Zero-knowledge against arbitrary verifiers can be achieved by applying standard techniques such as Damgård et al. [17,14].

Finally, it is a well known result that negligible knowledge and completeness errors in λ can be achieved, e.g., by running the protocol λ times in parallel and accepting if and only if at least $\lambda(1 - \alpha)/2$ transcripts were valid, if there exists a constant c such that $(1 - \alpha)/2 > 1/|\mathcal{C}| + c$.

Some of the Σ-protocol presented in this paper will further satisfy the following useful properties:

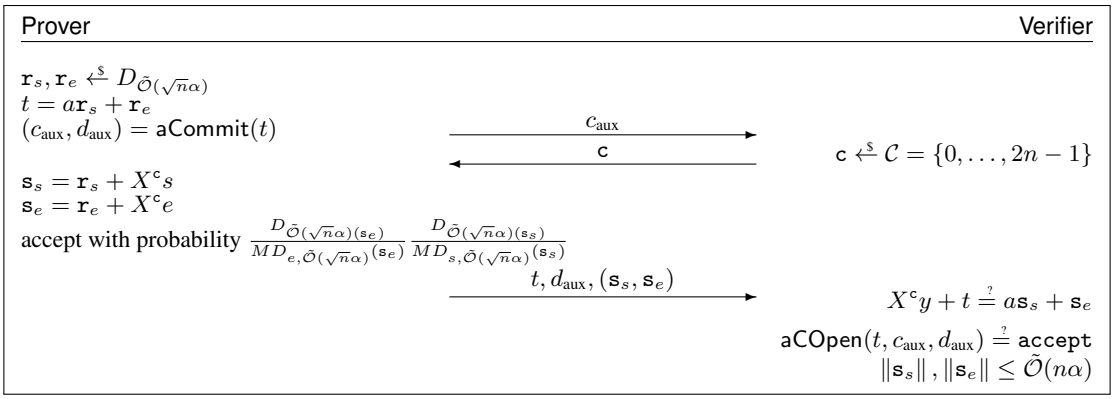
- *Quasi-unique responses*: No PPT adversary A can output $(y, \mathtt{t}, \mathtt{c}, \mathtt{s}, \mathtt{s}')$ with $\mathtt{s} \neq \mathtt{s}'$ such that $\mathsf{V}(y, \mathtt{t}, \mathtt{c}, \mathtt{s}) = \mathsf{V}(y, \mathtt{t}, \mathtt{c}, \mathtt{s}') = \mathtt{accept}$.
- *High-entropy commitments*: For all $(y, w) \in \mathcal{R}$ and for all $\mathtt{t}$, the probability that an honestly generated commitment by P takes on the value $\mathtt{t}$ is negligible.

3 Proving Knowledge of Ring-LWE Secrets

Before stating our basic protocol for efficiently proving knowledge of short s, e such that $2y = as + e$, we prove a technical lemma that is at the heart of the knowledge extractor of our protocols.

3.1 A Technical Lemma

The following lemma guarantees that certain binomials in $\mathbb{Z}[X]/(X^n + 1)$ can be inverted, and their inverses have only small coefficients.



Protocol 3.2: Proof of knowledge of LWE-secrets s, e such that $y = as + e$.

Lemma 3.1. *Let n be a power of 2 and let $0 < i, j < 2n-1$. Then $2(X^i - X^j)^{-1} \bmod (X^n+1)$ only has coefficients in $\{-1, 0, 1\}$.*

Proof. Without loss of generality, assume that $j > i$. Using that $X^n = -1 \bmod (X^n+1)$ we have that $2(X^i - X^j)^{-1} = -2X^{n-i}(1-X^{j-i})^{-1}$. It is therefore sufficient to prove the claim for $i = 0$ only.

Now remark that, for every $k \geq 1$ it holds that: $(1-X^j)(1+X^j+X^{2j}+\ldots+X^{(k-1)j}) = 1 - X^{kj}$.

Let us write $j = 2^{j'} j''$, with j'' a positive odd integer and $0 \leq j' \leq \log_2(n)$, and let us choose $k = 2^{\log_2(n)-j'}$ (recall that n is a power of 2). We then have $jk = nj''$, and $X^{kj} = (-1)^{j''} = -1 \bmod (X^n+1)$, hence $1 - X^{kj} = 2 \bmod (X^n+1)$. Therefore, we have

$$\begin{aligned} 2(1-X^j)^{-1} &= 1 + X^j + X^{2j} + \ldots + X^{(k-1)j} \bmod (X^n+1) \\ &= 1 \pm X^{j \bmod n} \pm X^{2j \bmod n} \pm \ldots \pm X^{(k-1)j \bmod n} \bmod (X^n+1)\,. \end{aligned}$$

Finally, in this equation, no two exponents are equal, since otherwise this means that n divides jk' with $1 \leq k' < k$, which is impossible by definition of k. □

3.2 The Protocol

We next present our basic protocol. Let therefore be $y = as + e$, where the LWE-secrets $s, e \xleftarrow{\$} D_\alpha$ are chosen from a discrete Gaussian distribution with standard deviation α. Protocol 3.2 now allows a prover to convince a verifier that it knows short s' and e' such that $2y = as' + e'$, i.e., the verifier is ensured that the prover knows short secrets for twice the public input. Here, by short we mean the following: An honest prover will always be able to convince the verifier whenever $\|s\|, \|e\| \leq \tilde{\mathcal{O}}(\sqrt{n}\alpha)$, which is the case with overwhelming probability if they were generated honestly. On the other hand, the verifier is guaranteed that the prover knows LWE-secrets with norm at most $\tilde{\mathcal{O}}(n^2\alpha)$. This soundness gap on the size of the witnesses is akin to those in, e.g., [16,1].

To be able to simulate aborts when proving the zero-knowledge property of the protocol, we must not send the prover's first message in the plain, but commit to it and later open it in the last round of the Σ-protocol. We therefore make use of an auxiliary commitment scheme $(\mathsf{aCSetup}, \mathsf{aCommit}, \mathsf{aCOpen})$, and assume that honestly generated commitment parameters are given as common input to both parties. We do not make any assumptions on the auxiliary commitment scheme. However, if it is computationally binding, the resulting protocol is only sound under the respective assumption, and similarly if it is computationally hiding. For simplicity, the reader may just think of the scheme as a random oracle.

Theorem 3.3. *Protocol 3.2 is an HVZK Σ'-protocol for the following relations:*

$$\begin{aligned} \mathcal{R} &= \{((s,e),(a,y)) : y = as + e \quad \wedge \quad \|s\|, \|e\| \leq \tilde{\mathcal{O}}(\sqrt{n}\alpha)\} \\ \mathcal{R}' &= \{((s,e),(a,y)) : 2y = as + e \quad \wedge \quad \|s\|, \|e\| \leq \tilde{\mathcal{O}}(n^2\alpha)\} \end{aligned}$$

The protocol has a knowledge error of $1/(2n)$, a completeness error of $1 - 1/M^2$, and high-entropy commitments.

Proof. We need to prove the properties of Definition 2.5.

Completeness. First note that by Theorem 2.4, the prover will respond with probability $1/M$ for each secret, i.e., with an overall probability of $1/M^2$. If the prover does not abort, we have that:

$$a\mathtt{s}_s + \mathtt{s}_e = a(\mathtt{r}_s + X^{\mathtt{c}}s) + (\mathtt{r}_e + X^{\mathtt{c}}e) = X^{\mathtt{c}}(as+e) + (a\mathtt{r}_s + \mathtt{r}_e) = X^{\mathtt{c}}y + \mathtt{t}\,.$$

For the norms we have that $\|\mathtt{s}_s\| \leq \|\mathtt{r}_s\| + \|s\| \leq \tilde{\mathcal{O}}(n\alpha)$ with overwhelming probability, as the standard deviations of $\mathtt{r}_s$ is $\tilde{\mathcal{O}}(\sqrt{n}\alpha)$, and similarly for $\mathtt{s}_e$.

Honest-verifier zero-knowledge. Given a challenge $\mathtt{c}$, the simulator outputs $(\mathsf{aCommit}(0), \mathtt{c}, \perp)$ with probability $1/M^2$. With probability $1 - 1/M^2$, the simulator S proceeds as follows: It chooses $\mathtt{s}_s, \mathtt{s}_e \xleftarrow{\$} D_{\tilde{\mathcal{O}}(\sqrt{n}\alpha)}$, and computes $\mathtt{t} = a\mathtt{s}_s + \mathtt{s}_e - X^{\mathtt{c}}y$, and $(c_{\text{aux}}, d_{\text{aux}}) \xleftarrow{\$} \mathsf{aCommit}(\mathtt{t})$. Finally, S outputs $(c_{\text{aux}}, \mathtt{c}, (\mathtt{t}, d_{\text{aux}}, (\mathtt{s}_s, \mathtt{s}_e)))$.

It follows from Theorem 2.4 that if no abort occurs the distribution of $\mathtt{s}_e, \mathtt{s}_s$ does not depend on s, e, and thus simulated and real protocol transcripts are indistinguishable. In case that an abort occurs, the indistinguishability follows from the hiding property of $\mathsf{aCommit}$ and the fact that aborts are equally likely for every $\mathtt{c}$.

Special soundness. Assume that we are given $(c_{\text{aux}}, \mathtt{c}', (\mathtt{t}', d'_{\text{aux}}, (\mathtt{s}'_s, \mathtt{s}'_e)))$ and $(c_{\text{aux}}, \mathtt{c}'', (\mathtt{t}'', d''_{\text{aux}}, (\mathtt{s}''_s, \mathtt{s}''_e)))$ passing the checks performed by the verifier. From the binding property of the auxiliary commitment scheme we get that $\mathtt{t}' = \mathtt{t}'' =: \mathtt{t}$. Now, by subtracting the verification equations we get: $(X^{\mathtt{c}'} - X^{\mathtt{c}''})y = a(\mathtt{s}'_s - \mathtt{s}''_s) + (\mathtt{s}'_e - \mathtt{s}''_e)$. Multiplying by $2(X^{\mathtt{c}'} - X^{\mathtt{c}''})^{-1}$ yields:

$$2y = a\frac{2(\mathtt{s}'_s - \mathtt{s}''_s)}{X^{\mathtt{c}'} - X^{\mathtt{c}''}} + \frac{2(\mathtt{s}'_e - \mathtt{s}''_e)}{X^{\mathtt{c}'} - X^{\mathtt{c}''}} =: a\hat{s} + \hat{e}\,.$$

Furthermore, we get that $\|\hat{s}\| \leq \|\mathtt{s}'_e - \mathtt{s}''_e\|\sqrt{n}\left\|\frac{2}{X^{\mathtt{c}'} - X^{\mathtt{c}''}}\right\| \leq \tilde{\mathcal{O}}(n^2\alpha)$, where in the second inequality we used Lemma 3.1, and similarly for $\hat{e}$.

High-entropy commitments. This directly follows from the security of the auxiliary commitment scheme. □

By Section 2.5, both the completeness and the knowledge error can be made negligible if $n > M^2$.

4 Proving Equality among Classical and Lattice-Based Primitives

In the following we show how our basic protocol from Section 3 can be used to link number-theory and lattice-based primitives via zero-knowledge proofs of knowledge. We exemplify this by showing how to prove that the messages contained in Pedersen commitments correspond to the plaintext in an encryption under the secure version of NTRU. We want to stress that in particular the choice of the encryption scheme is arbitrary, and it is easy to exchange it against other schemes, including standard NTRU [22] or Ring-LWE encryption [28].

Let $y = hs + pe + m \in R_q$ be the NTRU encryption of a message $m \in \{0,1\}^n$, and let $p > 2n^2$ be coprime with q. Let further $\tilde{g}, \tilde{h}$ be a Pedersen commitment parameters, cf. Section 2.2, and let $\widetilde{cmt}_i = m_i\tilde{g} + r_i\tilde{h}$ for $i = 0, \ldots, n-1$ be commitments to coefficients of m, where the order of $\tilde{g}$ and $\tilde{h}$ is $\tilde{q} > 2n^2$.

Then Protocol 4.1 can be used to prove, in zero-knowledge, that the commitments and the ciphertext are broadly well-formed and consistent, i.e., contain the same message. More precisely, the protocol guarantees the verifier that the prover knows the plaintext encrypted in $2y$, and that the coefficients of the respective message are all smaller than p. Furthermore, it shows that the messages are the same that are contained in $2\widetilde{cmt}_i$, i.e., $2y$ and the $2\widetilde{cmt}_i$ are consistent.

Prover		Verifier
$\mathtt{r}_s, \mathtt{r}_e \xleftarrow{\$} D_{\tilde{\mathcal{O}}(\sqrt{n}\alpha)}$ $\mathtt{r}_m \xleftarrow{\$} D_{\tilde{\mathcal{O}}(\sqrt{n})}$ $\mathtt{r}_{r,i} \xleftarrow{\$} \mathbb{Z}_{\tilde{q}}$ for $i = 0, \dots, n-1$ $\mathtt{t} = h\mathtt{r}_s + p\mathtt{r}_e + \mathtt{r}_m$ $\tilde{\mathtt{t}}_i = \mathtt{r}_{m,i}\tilde{g} + \mathtt{r}_{r,i}\tilde{h}$ for $i = 0, \dots, 2n-1$ $(c_{\text{aux}}, d_{\text{aux}}) = \mathsf{aCommit}(\mathtt{t}, (\tilde{\mathtt{t}}_i)_{i=0}^{n-1})$	$\xrightarrow{c_{\text{aux}}}$ $\xleftarrow{\mathtt{c}}$	$\mathtt{c} \xleftarrow{\$} \mathcal{C} = \{0, \dots, 2n-1\}$
$\mathtt{s}_s = \mathtt{r}_s + X^{\mathtt{c}} s$ $\mathtt{s}_e = \mathtt{r}_e + X^{\mathtt{c}} e$ $\mathtt{s}_m = \mathtt{r}_m + X^{\mathtt{c}} m$ $\mathtt{s}_r = \mathtt{r}_r + (r_0, \dots, r_{n-1})_{\ll \mathtt{c}}$ accept with probability $\frac{D_{\tilde{\mathcal{O}}(\sqrt{n}\alpha)}(\mathtt{s}_e)}{MD_{e,\tilde{\mathcal{O}}(\sqrt{n}\alpha)}(\mathtt{s}_e)} \frac{D_{\tilde{\mathcal{O}}(\sqrt{n}\alpha)}(\mathtt{s}_s)}{MD_{s,\tilde{\mathcal{O}}(\sqrt{n}\alpha)}(\mathtt{s}_s)} \frac{D_{\tilde{\mathcal{O}}(\sqrt{n})}(\mathtt{s}_m)}{MD_{m,\tilde{\mathcal{O}}(\sqrt{n})}(\mathtt{s}_m)}$		
	$\xrightarrow{(\mathtt{t}, (\tilde{\mathtt{t}}_i)_{i=0}^{n-1}), d_{\text{aux}}, (\mathtt{s}_s, \mathtt{s}_e, \mathtt{s}_m, \mathtt{s}_r)}$	
		$X^{\mathtt{c}} y + \mathtt{t} \stackrel{?}{=} h\mathtt{s}_s + p\mathtt{s}_e + \mathtt{s}_m$
		$(\widetilde{cmt}_0, \dots, \widetilde{cmt}_{n-1})_{\ll \mathtt{c}} + (\tilde{\mathtt{t}}_0, \dots, \tilde{\mathtt{t}}_{n-1}) \stackrel{?}{=} \mathtt{s}_m\tilde{g} + \mathtt{s}_r\tilde{h}$
		$\mathsf{aCOpen}((\mathtt{t}, \tilde{\mathtt{t}}_0, \dots, \tilde{\mathtt{t}}_{n-1}), c_{\text{aux}}, d_{\text{aux}}) \stackrel{?}{=} \mathtt{accept}$
		$\|\mathtt{s}_s\|, \|\mathtt{s}_e\| \leq \tilde{\mathcal{O}}(n\alpha)$
		$\|\mathtt{s}_m\| \leq \tilde{\mathcal{O}}(n)$

Protocol 4.1: Proof that Pedersen commitments and NTRU encryption contain the same plaintext.

Theorem 4.2. *Protocol 4.1 is an HVZK Σ'-protocol for the following relations:*

$$\mathcal{R} = \left\{ ((m, s, e, (r_i)_{i=0}^{n-1}), (\tilde{g}, \tilde{h}, (\widetilde{cmt}_i)_{i=0}^{n-1}, h, p, y)) : y = hs + pe + m \wedge \bigwedge_{i=0}^{n-1} \widetilde{cmt}_i = m_i\tilde{g} + r_i\tilde{h} \right.$$
$$\left. \wedge \|m\|_\infty \leq 1 \wedge \|s\|, \|e\| \leq \tilde{\mathcal{O}}(\sqrt{n}\alpha) \right\},$$

$$\mathcal{R}' = \left\{ ((m, s, e, (r_i)_{i=0}^{n-1}), (\tilde{g}, \tilde{h}, (\widetilde{cmt}_i)_{i=0}^{n-1}, h, p, y)) : 2y = hs + pe + m \wedge \bigwedge_{i=0}^{n-1} 2\widetilde{cmt}_i = m_i\tilde{g} + r_i\tilde{h} \right.$$
$$\left. \wedge \|m\|_\infty \leq 2n^2 \wedge \|s\|, \|e\| \leq \tilde{\mathcal{O}}(n^2\alpha) \right\}$$

The protocol has a knowledge error of $1/(2n)$, and a completeness error of $1 - 1/M^3$.

Furthermore, if for the auxiliary commitment scheme a commitment does not only bind the user to the message, but also to the opening information, the protocol has quasi-unique responses and high-entropy commitments.

A detailed proof is given in Appendix C.2. By the remark in Section 2.5, both the completeness and the knowledge error can be made negligible if $n > M^3$.

5 Application to Group Signatures

We next show how Protocol 4.1 can be used to construct a group signature scheme whose signature size is only logarithmic in the number of group members. As discussed in Section 1.3, our scheme will be private under lattice assumptions, but only unforgeable under the discrete logarithm assumption, which is sufficient for many applications such as e-voting, where long-term privacy is vital.

However, before presenting the actual signature scheme, we will prove secure a generic construction that is folklore, and underlies several direct constructions in the literature [2,8] and a variant was described explicitly by Chase and Lysyanskaya [11]. However, we provide the first formal proof that this construction actually yields secure group signature schemes.

The resulting construction will satisfy the following definition of group signatures providing full (CCA) anonymity, and is a variant of the definition put forth by Bellare et al. [4].

Experiment $\mathbf{Exp}^{\text{anon}-b}_{\mathsf{A}}(\lambda)$:	Experiment $\mathbf{Exp}^{\text{trace}}_{\mathsf{A}}(\lambda)$:
$(gpk, gok, \mathbf{gsk}) \xleftarrow{\$} \mathsf{GKG}(1^\lambda, 1^n)$	$(gpk, gok, \mathbf{gsk}) \xleftarrow{\$} \mathsf{GKG}(1^\lambda, 1^n)$
$(st, i_0^*, i_1^*, m^*) \xleftarrow{\$} \mathsf{A}^{\mathsf{GOpen}(gok,\cdot,\cdot)}((gpk, \mathbf{gsk}), \varepsilon)$	$(m, \sigma) \xleftarrow{\$} \mathsf{A}^{\mathsf{GSign}(\mathbf{gsk}[\cdot],\cdot), \mathbf{gsk}[\cdot]}(gpk, gok)$
$\sigma^* \xleftarrow{\$} \mathsf{GSign}(\mathbf{gsk}[i_b], m^*)$	$i \xleftarrow{\$} \mathsf{GOpen}(gok, m, \sigma)$
$b' \xleftarrow{\$} \mathsf{A}^{\mathsf{GOpen}(gok,\cdot,\cdot)}(\sigma^*, st)$	If $\mathsf{GVerify}(gpk, m, \sigma) = 1 \ \wedge \ i \notin \mathcal{Q}_{\mathbf{gsk}}$
If $(m^*, \sigma^*) \notin \mathcal{Q}_{\mathsf{GOpen}}$ return b' else return 0	$\wedge \ (i, m) \notin \mathcal{Q}_{\mathsf{GSign}}$ then return 1 else return 0

Fig. 1. The anonymity (left) and traceability (right) experiments for group signatures. The sets $\mathcal{Q}_{\mathsf{GOpen}}, \mathcal{Q}_{\mathsf{GSign}}, \mathcal{Q}_{\mathbf{gsk}}$ contain all queries (m, σ), (i, m), and i that A submitted to its GOpen, GSign, and $\mathbf{gsk}$ oracles, respectively.

Definition 5.1. *A* group signature scheme *is a tuple* $(\mathsf{GKG}, \mathsf{GSign}, \mathsf{GVerify}, \mathsf{GOpen})$ *where:*

- *On input* $1^\lambda, 1^n$*, the key generation algorithm* GKG *outputs a group public key* gpk*, an opening key* gok*, and a vector of* n *signing keys* $\mathbf{gsk}$ *where* $\mathbf{gsk}[i]$ *is given to user* $i \in \{1, \ldots, n\}$*.*
- *On input* $gsk = \mathbf{gsk}[i]$ *and message* $m \in \mathcal{M}$*, the signing algorithm* GSign *outputs a group signature* σ*.*
- *On input* gpk, m, σ*, the verification algorithm* $\mathsf{GVerify}$ *outputs* `accept` *or* `reject`*.*
- *On input* gok, m, σ*, the opening algorithm* GOpen *outputs the identity of the purported signer* $i \in \{1, \ldots, n\}$ *or* $\perp$ *to indicate failure.*

The algorithms satisfy the following properties:

- Correctness*: Verification accepts whenever keys and signatures are honestly generated, i.e., for all* $\lambda, n \in \mathbb{N}$*, all* $i \in \{1, \ldots, n\}$*, and all* $m \in \mathcal{M}$

$$\Pr[\mathsf{GVerify}(gpk, m, \sigma) = \texttt{accept} : (gpk, gok, \mathbf{gsk}) \xleftarrow{\$} \mathsf{GKG}(1^\lambda, 1^n), \sigma \xleftarrow{\$} \mathsf{GSign}(\mathbf{gsk}[i], m)] = 1 \ .$$

- Anonymity*: One cannot tell which signer generated a particular signature, even when given access to an opening oracle. Referring to Figure 1, for all PPT* A *there exists a negligible function* negl *such that*

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

- Traceability*: One cannot generate a signature that cannot be opened or that opens to an honest user. Referring to Figure 1, for all PPT* A *there exists a negligible function* negl *such that*

$$\Pr[\mathbf{Exp}^{\text{trace}}_{\mathsf{A}}(\lambda)] \leq \mathsf{negl}(\lambda) \ .$$

5.1 Building Blocks

The construction is based on weakly unforgeable standard signatures, and signature proofs of knowledge. In the following, we recap the respective definitions.

Informally, a signature scheme is a triple $(\mathsf{SKG}, \mathsf{SSign}, \mathsf{SVerify})$, where SKG generates a signing/verification key pair (ssk, spk), SSign can be used to sign a message m using the signing key, and $\mathsf{SVerify}$ can be used to check the validity of a signature only using the public verification key. It should hold that honestly computed signatures are always valid, and that no adversary can come up with a valid signature on a new message after having received signatures on message that he chose before obtaining spk. A formal definition can be found in Appendix A.3.

We adapt definitions of Chase-Lysyanskaya to allow for signatures in the random-oracle model that are simulated by programming the random oracle and extracted through rewinding. We also add definition of simulation soundness, meaning that adversary cannot produce new signatures for false statements even after seeing simulated signatures on arbitrary statements.

Definition 5.2. *A* signature of knowledge scheme *for a language* $\mathcal{L} \subseteq \{0,1\}^*$ *with witness relation* $R_{\mathcal{L}}$ *is a tuple* $(\mathsf{SoKSetup}, \mathsf{SoKSign}, \mathsf{SoKVerify}, \mathsf{SoKSim})$ *where:*

– *On input* 1^λ*, the setup algorithm* SoKSetup *outputs common parameters* $sokp$*.*
– *On input* $sokp, x, w$ *such that* $(x, w) \in R_{\mathcal{L}}$ *and message* $m \in \mathcal{M}$*, the signing algorithm* SoKSign *outputs a signature of knowledge* sok*.*
– *On input* $sokp, x, m, sok$*, the verification algorithm* SoKVerify *outputs* `accept` *or* `reject`*.*
– *The stateful simulation algorithm* SoKSim *can be called in three modes. When called as* $(sokp, st) \xleftarrow{\$} \mathsf{SoKSim}(\mathtt{setup}, 1^\lambda, \varepsilon)$*, it produces simulated parameters* $sokp$*, possibly keeping a trapdoor in its internal state* st*. When run as* $(h, st') \xleftarrow{\$} \mathsf{SoKSim}(\mathtt{ro}, Q, st)$*, it produces a response* h *for a random oracle query* Q*. When run as* $(sok, st') \xleftarrow{\$} \mathsf{SoKSim}(\mathtt{sign}, x, m, st)$*, it produces a simulated signature of knowledge* sok *without using a witness.*
For ease of notation, let $\mathsf{StpSim}(1^\lambda)$ *be the algorithm that returns the first part of* $\mathsf{SoKSim}(\mathtt{setup}, 1^\lambda, st)$*, let* $\mathsf{ROSim}(Q)$ *be the algorithm that returns the first part of* $\mathsf{SoKSim}(\mathtt{ro}, Q, st)$*, let* $\mathsf{SSim}(x, w, m)$ *be the algorithm that returns the first part of* $\mathsf{SoKSim}(\mathtt{sign}, x, m, st)$ *if* $(x, w) \in R_{\mathcal{L}}$ *and returns* $\perp$ *otherwise, and let* $\mathsf{SSim}'(x, m)$ *be the algorithm that returns the first part of* $\mathsf{SoKSim}(\mathtt{sign}, x, m, st)$ *without checking language membership. The experiment keeps a single synchronized state for* SoKSim *across all invocations of these derived algorithms.*

The algorithms satisfy the following properties:

– Correctness*: Verification accepts whenever parameters and signatures are correctly generated, i.e., for all* $\lambda \in \mathbb{N}$*, all* $(x, w) \in R_{\mathcal{L}}$*, and all* $m \in \mathcal{M}$*, there exists a negligible function* negl *such that*

$$\Pr\left[\begin{array}{c}\mathsf{SoKVerify}(sokp, x, m, sok) = \mathtt{reject} : \\ sokp \xleftarrow{\$} \mathsf{SoKSetup}(1^\lambda),\ sok \xleftarrow{\$} \mathsf{SoKSign}(sokp, x, w, m)\end{array}\right] \leq \mathsf{negl}(1^\lambda) .$$

– Simulatability*: No adversary can distinguish whether it is interacting with a real random oracle and signing oracle, or with their simulated versions. Formally, for all PPT* A *there exists a negligible function* negl *such that*

$$\begin{aligned}&\Big|\Pr[b = 1 : sokp \xleftarrow{\$} \mathsf{StpSim}(1^\lambda), b \xleftarrow{\$} \mathsf{A}^{\mathsf{ROSim}(\cdot),\mathsf{SSim}(\cdot,\cdot,\cdot)}(sokp)] \\ &- \Pr[b = 1 : sokp \xleftarrow{\$} \mathsf{SoKSetup}(1^\lambda), b \xleftarrow{\$} \mathsf{A}^{H(\cdot),\mathsf{SoKSign}(sokp,\cdot,\cdot,\cdot)}(sokp)]\Big| \leq \mathsf{negl}(\lambda) .\end{aligned}$$

– Extractability*: The only way to produce a valid signature of knowledge is by knowing a witness. Formally, for all PPT* A *there exists an extractor* $\mathsf{SoKExt}_{\mathsf{A}}$ *and a negligible function* negl *such that*

$$\Pr\left[\begin{array}{c}\mathsf{SoKVerify}(sokp, x, m, sok) = \mathtt{accept} \wedge (x, w, m) \notin \mathcal{Q} \wedge (x, w) \notin R_{\mathcal{L}} : \\ sokp \xleftarrow{\$} \mathsf{StpSim}(1^\lambda; \rho_{\mathsf{S}}), (x, m, sok) \xleftarrow{\$} \mathsf{A}^{\mathsf{ROSim}(\cdot),\mathsf{SSim}(\cdot,\cdot,\cdot)}(sokp; \rho_{\mathsf{A}}), \\ w \xleftarrow{\$} \mathsf{SoKExt}_{\mathsf{A}}(sokp, x, m, sok, \rho_{\mathsf{S}}, \rho_{\mathsf{A}})\end{array}\right] \leq \mathsf{negl}(\lambda) ,$$

where $\mathcal{Q}$ *is the set of queries* (x, w, m) *that* A *submitted to its* SSim *oracle.*
– Simulation-soundness*: No adversary can produce a new signature on a false statement, even after seeing a signature on an arbitrary statement. Formally, for all PPT* A *there exists a negligible function* negl *such that*

$$\Pr\left[\begin{array}{c}\mathsf{SoKVerify}(sokp, x, m, sok) = \mathtt{accept} \wedge (x', m', sok') \neq (x, m, sok) \wedge x \notin \mathcal{L} : \\ sokp \xleftarrow{\$} \mathsf{StpSim}(1^\lambda), (x, m, st) \xleftarrow{\$} \mathsf{A}^{\mathsf{ROSim}(\cdot)}(sokp), sok \xleftarrow{\$} \mathsf{SSim}'(x, m), \\ (x', m', sok') \xleftarrow{\$} \mathsf{A}^{\mathsf{ROSim}(\cdot)}(sok, st)\end{array}\right] \leq \mathsf{negl}(\lambda) .$$

5.2 Generic Construction

A folklore construction of group signatures is to have a user's signing key be a standard signature on his identity i, and to have a group signature on message m be an encryption of his identity together with a signature of knowledge on m that the encrypted identity is equal to the identity in his signing key. The construction appeared implicitly [2,8] or explicitly [11] in the literature, but was never proved secure.

To obtain full anonymity, this generic construction would probably require CCA security from encryption scheme, but our NTRU variant is only semantically secure. We could apply a generic CCA-yielding transformation using random oracles or non-interactive zero-knowledge proofs of knowledge (NIZK), but this would make the signature of knowledge hopelessly inefficient. Instead, we take inspiration from the Naor-Yung construction [29,32] by using a semantically secure scheme to encrypt the user's identity twice under two different public keys and letting the signature of knowledge prove that both ciphertexts encrypt the same plaintext. Moreover, our proof system for LWE encryption in Section 3 does not quite prove knowledge of (m, ρ) such that $C = \mathsf{Enc}(epk, m, \rho)$, but rather of (m', ρ') such that $2C = \mathsf{Enc}(epk, m', \rho')$. We therefore give a generic construction that deviates slightly from the general idea, but that is sufficient and that we can efficiently instantiate with our protocol from Section 3.

Let $(\mathsf{EncKG}, \mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme with message space $\mathcal{M}$ and a homomorphism (Φ, φ, ψ) such that φ is an injective function and such that for all epk and for all $m \in \mathcal{ID} \subseteq \mathcal{M}$

$$C = \mathsf{Enc}(epk, m; \rho) \;\Rightarrow\; \Phi(C) = \mathsf{Enc}(epk, \varphi(m); \psi(\rho)) \; .$$

Let $(\mathsf{SKG}, \mathsf{SSign}, \mathsf{SVerify})$ be a standard signature scheme, and let $(\mathsf{SoKSetup}, \mathsf{SoKSign}, \mathsf{SoKVerify}, \mathsf{SoKSim})$ be a signature of knowledge scheme for the language $\mathcal{L}$ with witness relationship

$$R_{\mathcal{L}} = \{((spk, epk_1, epk_2, C_1, C_2), (i, sig, \rho_1, \rho_2)) \; : \\ \mathsf{SVerify}(spk, i, sig) = \mathtt{accept} \wedge \Phi(C_1) = \mathsf{Enc}(epk_1, \varphi(i); \rho_1) \wedge \Phi(C_2) = \mathsf{Enc}(epk_2, \varphi(i); \rho_2)\} \; ,$$

consider the following group signature scheme with user identities $i \in \mathcal{ID}$:

- $\mathsf{GKG}(1^\lambda, 1^n)$: The group manager generates signature keys $(spk, ssk) \xleftarrow{\$} \mathsf{SKG}(1^\lambda)$, encryption keys $(epk_1, esk_1) \xleftarrow{\$} \mathsf{EncKG}(1^\lambda)$, $(epk_2, esk_2) \xleftarrow{\$} \mathsf{EncKG}(1^\lambda)$, and parameters $sokp \xleftarrow{\$} \mathsf{SoKSetup}(1^\lambda)$. He computes $\mathbf{gsk}[i] \xleftarrow{\$} \mathsf{SSign}(ssk, i)$ for $i \in \mathcal{ID}$ and outputs $gpk = (spk, epk_1, epk_2, sokp)$, $gok = esk_1$, and $\mathbf{gsk}$.
- $\mathsf{GSign}(gsk, m)$: Signer i computes two ciphertexts $C_1 \leftarrow \mathsf{Enc}(epk_1, i; \rho_1)$ and $C_2 \leftarrow \mathsf{Enc}(epk_2, i; \rho_2)$, computes a signature of knowledge $sok \xleftarrow{\$} \mathsf{SoKSign}(sokp, (spk, epk_1, epk_2, C_1, C_2), (i, sig, \psi(\rho_1), \psi(\rho_2)), m)$ and outputs $\sigma = (C_1, C_2, sok)$.
- $\mathsf{GVerify}(gpk, m, \sigma)$: To verify a group signature, one checks that $\mathsf{SoKVerify}(sokp, (spk, epk_1, epk_2, C_1, C_2), m, sok) = \mathtt{accept}$.
- $\mathsf{GOpen}(gok, m, \sigma)$: The opener decrypts $d \leftarrow \mathsf{Dec}(esk_1, C_1)$ and returns $i \in \mathcal{ID}$ such that $\varphi(i) = d$ (by inverting φ if possible, otherwise by exhaustive search).

Theorem 5.3. *The group signature scheme sketched above is anonymous if the encryption scheme is semantically secure and perfectly complete, and the signature of knowledge scheme is simulatable and simulation-sound.*

Theorem 5.4. *The group signature scheme is traceable if the underlying signature scheme is weakly unforgeable, the encryption scheme is perfectly complete, and the signature of knowledge scheme is simulatable and extractable.*

The proofs of the last two theorems are omitted here and are given in Appendices C.3 and C.4.

5.3 Signatures of Knowledge from Σ-Protocols

We now show a construction of the required signatures of knowledge in the random-oracle model from a signature scheme and an encryption scheme with Σ-protocol proofs. More particularly, we require that the signature scheme can prove knowledge of a signature on a committed message, while the encryption scheme can prove that an encrypted plaintext is equal to a committed message.

Let $(\mathsf{EncKG}, \mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme with message space $\mathcal{M}$, let $(\mathsf{CSetup}, \mathsf{Commit}, \mathsf{COpen})$ be a commitment scheme, and let (Φ, φ, ψ) and (Ψ, φ, χ) be homomorphisms as before such that for all $cpars$:

$$\mathsf{COpen}(cpars, cmt, o) = \mathtt{accept} \;\Rightarrow\; \mathsf{COpen}(cpars, \Psi(cmt), \chi(o)) = \mathtt{accept} \; .$$

Let $(\mathsf{P}_s, \mathsf{V}_s, \mathsf{S}_s)$ be a Σ-protocol for the language $\mathcal{L}_s$ with

$$R_{\mathcal{L}_s} = \{((spk, cpars, cmt), (sig, m, o)) : \mathsf{SVerify}(spk, m, sig) = \mathtt{accept} \wedge \mathsf{COpen}(cpars, m, cmt, o) = \mathtt{accept}\}$$

and let $(\mathsf{P}_e, \mathsf{V}_e, \mathsf{S}_e)$ be a Σ-protocol for the language $\mathcal{L}_e$ with

$$R_{\mathcal{L}_e} = \{((epk, C, cpars, cmt), (m, \rho, o)) : \Phi(C) = \mathsf{Enc}(epk, m; \rho) \wedge \mathsf{COpen}(cpars, m, \Psi(cmt), o) = \mathtt{accept}\} .$$

Let $\mathcal{C}_s$ and $\mathcal{C}_e$ be the challenge spaces for these respective Σ-protocols, and let $H : \{0,1\}^* \to \mathcal{C}_s \times \mathcal{C}_e$. Consider the following construction of a signature of knowledge scheme for the language $\mathcal{L}$:

- $\mathsf{SoKSetup}(1^\lambda)$: Return $sokp = cpars \xleftarrow{\$} \mathsf{CSetup}(1^\lambda)$.
- $\mathsf{SoKSign}(sokp, x, w, m)$: Create a commitment $(cmt, o) \xleftarrow{\$} \mathsf{Commit}(i, cpars)$. Compute the first round of one signature Σ-protocol and two encryption Σ-protocols: $(\mathtt{t}_s, st_s) \xleftarrow{\$} \mathsf{P}_s((spk, cpars, cmt), (sig, m, o))$ and $(\mathtt{t}_j, st_j) \xleftarrow{\$} \mathsf{P}_e((epk_j, C_j, cpars, cmt), (\varphi(i), \psi(\rho_j), \chi(o)))$ for $j = 1, 2$. Generate the challenges $(\mathtt{c}_s, \mathtt{c}_e) \leftarrow H(spk, cpars, cmt, epk_1, C_1, epk_2, C_2, \mathtt{t}_s, \mathtt{t}_1, \mathtt{t}_2, m)$. Generate the responses $\mathtt{s}_s \leftarrow \mathsf{P}_s(\mathtt{c}_s, st_s)$ and $\mathtt{s}_j \leftarrow \mathsf{P}_e(\mathtt{c}_e, st_e)$ for $j = 1, 2$ and output the signature of knowledge $sok = (\mathtt{t}_s, \mathtt{t}_1, \mathtt{t}_2, \mathtt{s}_s, \mathtt{s}_1, \mathtt{s}_2)$.
- $\mathsf{SoKVerify}(sokp, x, m, sok)$: Recompute the challenges $(\mathtt{c}_s, \mathtt{c}_e) \leftarrow H(spk, cpars, cmt, epk_1, C_1, epk_2, C_2, \mathtt{t}_s, \mathtt{t}_1, \mathtt{t}_2, m)$. Return $\mathtt{accept}$ if $\mathsf{V}_s((spk, cpars, cmt), \mathtt{t}_s, \mathtt{c}_s, \mathtt{s}_s) = \mathtt{accept}$ and $\mathsf{V}_e((epk_j, C_j, cpars, cmt), \mathtt{t}_j, \mathtt{c}_e, \mathtt{s}_j) = \mathtt{accept}$ for $j = 1, 2$. Otherwise, return $\mathtt{reject}$.
- SoKSim: The simulation algorithm keeps in its state its random tape, an initially empty table HT to keep track of previous random-oracle queries, and a counter ctr initialized to zero. The simulator's random tape ρ includes random-oracle responses $h_1, \ldots, h_{q_H+q_S} \xleftarrow{\$} \mathcal{C}_s \times \mathcal{C}_e$, where q_H and q_S are upper bounds on the number of random-oracle and signing queries that an adversary can make. When called as $\mathsf{SoKSim}(\mathtt{setup}, 1^\lambda, \varepsilon)$, it generates commitment parameters $cpars \xleftarrow{\$} \mathsf{CSetup}(1^\lambda)$ and returns $(cpars, st = (\rho, HT, ctr, cpars))$. When run as $\mathsf{SoKSim}(\mathtt{ro}, Q, st)$, it checks whether the query Q was made before. If so, it returns $h_{HT[q]}$. Otherwise, it increases the counter ctr, sets $HT[Q] \leftarrow ctr$, and returns h_{ctr}. When run as $\mathsf{SoKSim}(\mathtt{sign}, (spk, epk_1, epk_2, C_1, C_2), m, st)$, the simulator first creates a commitment $(cmt, o) \xleftarrow{\$} \mathsf{Commit}(1, cpars)$. It then increases the counter ctr and parses h_{ctr} as $(\mathtt{c}_s, \mathtt{c}_e)$. It runs the simulators $\mathsf{S}_s, \mathsf{S}_e$ to obtain simulated protocol transcripts $(\mathtt{t}_s, \mathtt{s}_s) \xleftarrow{\$} \mathsf{S}_s((spk, cpars, cmt), \mathtt{c}_s)$ and $(\mathtt{t}_j, \mathtt{s}_j) \xleftarrow{\$} \mathsf{S}_s((epk_j, C_j, cpars, cmt), \mathtt{c}_e)$ for $j = 1, 2$. If $HT[spk, cpars, cmt, epk_1, C_1, epk_2, C_2, \mathtt{t}_s, \mathtt{t}_1, \mathtt{t}_2, m]$ is not defined, then set it to h_{ctr}, else abort.

Theorem 5.5. *The above scheme is correct if the proof systems $(\mathsf{P}_s, \mathsf{V}_s)$ and $(\mathsf{P}_e, \mathsf{V}_e)$ are complete.*

Theorem 5.6. *The above scheme is simulatable in the random-oracle model if the commitment scheme is hiding and the proof systems $(\mathsf{P}_s, \mathsf{V}_s)$ and $(\mathsf{P}_e, \mathsf{V}_e)$ are special HVZK and have high-entropy commitments.*

Theorem 5.7. *The above scheme is extractable in the random-oracle model if the commitment scheme is binding and the proof systems $(\mathsf{P}_s, \mathsf{V}_s)$ and $(\mathsf{P}_e, \mathsf{V}_e)$ are special-sound and have super-polynomial challenge spaces.*

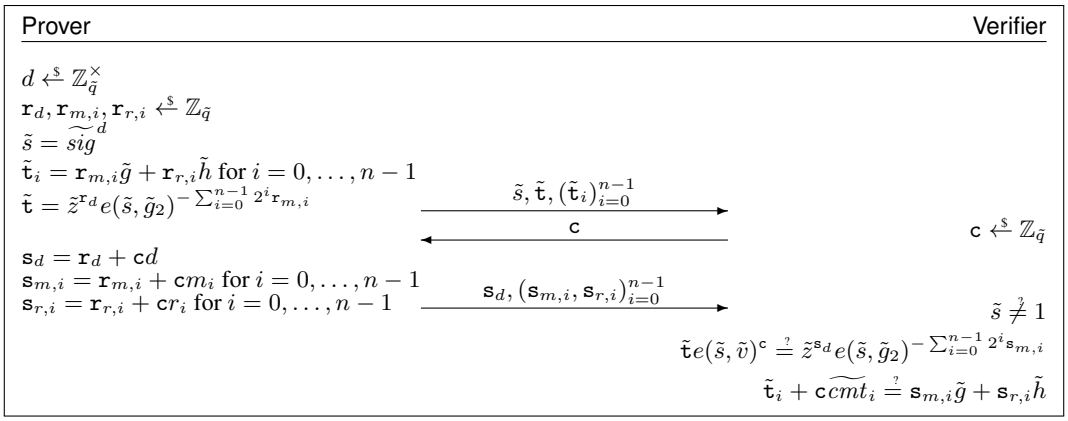
Theorem 5.8. *The above scheme is simulation-sound if the underlying commitment scheme is binding and the underlying Σ-protocols $(\mathsf{P}_s, \mathsf{V}_s, \mathsf{S}_s)$ and $(\mathsf{P}_e, \mathsf{V}_e, \mathsf{S}_e)$ are special-sound, have quasi-unique responses, and have super-polynomial challenge spaces.*

Due to length limitations, the proofs of the previous theorems can be found in Appendices C.5 to C.8.

5.4 Σ-Protocols for Boneh-Boyen Signatures and the Group Signature Scheme

In the following we briefly recap the weakly unforgeable version of the Boneh-Boyen signature scheme [6,7]. We assume that the reader is familiar with bilinear pairings and the strong Diffie-Hellman (SDH) assumption, and only recap them in Appendix A.4.

The Boneh-Boyen signature scheme is defined as follows for a bilinear group generator BGGen:



Protocol 5.10: Proof of possession of a signature on m, which is also contained in a set of Pedersen commitments.

SKG. This algorithm first computes $(\tilde{q}, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \xleftarrow{\$} \mathsf{BGGen}(1^\lambda)$. It chooses $\tilde{g}_1 \xleftarrow{\$} \mathbb{G}_1$, $\tilde{g}_2 \xleftarrow{\$} \mathbb{G}_2$, $x \xleftarrow{\$} \mathbb{Z}_{\tilde{q}}^{\times}$, and defines $\tilde{v} = \tilde{g}_2^x$ and $\tilde{z} = e(\tilde{g}_1, \tilde{g}_2)$. It outputs $spk = ((\tilde{q}, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e), \tilde{g}_1, \tilde{g}_2, \tilde{v}, \tilde{z})$ and $ssk = x$.

SSign. To sign a message $m \in \mathbb{Z}_{\tilde{q}}$ with secret key $ssk = x$, this algorithm outputs the signature $\widetilde{sig} = \tilde{g}_1^{\frac{1}{x+m}}$.

SVerify. Given a signature public key spk, a message $m \in \mathbb{Z}_{\tilde{q}}$ and a signature $\widetilde{sig}$, this algorithm outputs `accept` if $\tilde{v}\tilde{g}_2^m = 1$ in case $\widetilde{sig} = 1$, and if $e(\widetilde{sig}, \tilde{v}\tilde{g}_2^m) = \tilde{z}$ in case $\widetilde{sig} \neq 1$. In all other cases, it outputs `reject`.

Lemma 5.9. *If the SDH assumption holds for* BGGen*, then the above scheme is a weakly unforgeable signature scheme according to Definition A.3.*

We next show how a user can prove possession of a Boneh-Boyen signature on a message m, while keeping both, the message and the signature, private. In addition, the proof will additionally show that the m is also contained in a set of Pedersen commitments $\widetilde{cmt}_i = m_i\tilde{g} + r_i\tilde{h}$ such that $m = \sum_{i=0}^{n-1} 2^i m_i$, cf. Section 2.2.

The idea underlying Protocol 5.10 is similar to that in Camenisch et al. [9]: The prover first rerandomizes the signature to obtain a value s, which it sends to the verifier. Subsequently, the prover and the verifier run a standard Schnorr proof for the resulting statement.

Theorem 5.11. *Protocol 5.10 is a perfectly HVZK proof of knowledge for the following relations:*

$$\mathcal{R} = \Bigg\{ ((\widetilde{sig}, m, r, (m_i, r_i)_{i=0}^{n-1}), (spk, (\widetilde{cmt}_i)_{i=0}^{n-1})) : \mathsf{SVerify}(spk, m, \widetilde{sig}) = \mathtt{accept} \ \wedge \\ \widetilde{cmt}_i = m_i\tilde{g} + r_i\tilde{h} \ \wedge \ \tilde{v}\tilde{g}_2^m \neq 1 \ \wedge \ m = \sum_{i=0}^{n-1} 2^i m_i \Bigg\}$$

and $\mathcal{R}'$ is defined as $\mathcal{R}$ but without the condition that $\tilde{v}\tilde{g}_2^m \neq 1$.

The protocol is perfectly complete, and has a knowledge error of $1/\tilde{q}$. Furthermore, the protocol has quasi unique responses and high-entropy commitments.

The proof of this theorem is straightforward and can be found in Appendix C.9.

The Group Signature Scheme. Combining Protocols 4.1 and 5.10 now directly gives a group signature by the construction from Section 5.3. The $\mathcal{ID}$ is given by $\{0,1\}^\ell$ (which can be identified with $\{0, \ldots, 2^\ell - 1\}$), where $\ell = \min(n, \lfloor \log_2 \tilde{q} \rfloor - 1)$, n is the dimension of the ring being used and $\tilde{q}$ is the order of the groups of the commitment- and the signature schemes. All homomorphisms there can just be instantiated with a multiplication by 2. Note here that for the given identity space and the way our constructions work, multiplying by 2 can be seen as multiplication over the integers, as no modular reductions take place for all elements from $\mathcal{ID}$. Note further that NTRU can be made perfectly complete by truncating the discrete Gaussian distributions for key generation and encryption appropriately without touching security.

References

1. Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 483–501. Springer, April 2012.
2. Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. A practical and provably secure coalition-resistant group signature scheme. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 255–270. Springer, August 2000.
3. Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In *CRYPTO*, pages 390–420, 1992.
4. Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 614–629. Springer, May 2003.
5. Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 06*, pages 390–399. ACM Press, October / November 2006.
6. Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 56–73. Springer, May 2004.
7. Dan Boneh and Xavier Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *Journal of Cryptology*, 21(2):149–177, April 2008.
8. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, August 2004.
9. Jan Camenisch, Maria Dubovitskaya, and Gregory Neven. Oblivious transfer with access control. In *ACM Conference on Computer and Communications Security*, pages 131–140, 2009.
10. Jan Camenisch, Gregory Neven, and Markus Rückert. Fully anonymous attribute tokens from lattices. In Ivan Visconti and Roberto De Prisco, editors, *SCN 12*, volume 7485 of *LNCS*, pages 57–75. Springer, September 2012.
11. Melissa Chase and Anna Lysyanskaya. On signatures of knowledge. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 78–96. Springer, August 2006.
12. David Chaum and Eugène van Heyst. Group signatures. In Donald W. Davies, editor, *EUROCRYPT'91*, volume 547 of *LNCS*, pages 257–265. Springer, April 1991.
13. Ronald Cramer. *Modular Design of Secure yet Practical Cryptographic Protocols*. PhD thesis, CWI and University of Amsterdam, 1997.
14. Ivan Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In *EUROCRYPT*, pages 418–430, 2000.
15. Ivan Damgård. On Σ-Protocols. Lecture on Cryptologic Protocol Theory; Faculty of Science, University of Aarhus, 2010.
16. Ivan Damgård and Eiichiro Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In *ASIACRYPT*, pages 125–142, 2002.
17. Ivan Damgård, Oded Goldreich, Tatsuaki Okamoto, and Avi Wigderson. Honest verifier vs dishonest verifier in public coin zero-knowledge proofs. In *CRYPTO*, pages 325–338, 1995.
18. Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *CRYPTO*, pages 643–662, 2012.
19. Eiichiro Fujisaki and Tatsuaki Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In *CRYPTO*, pages 16–30, 1997.
20. Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206, 2008.
21. S. Dov Gordon, Jonathan Katz, and Vinod Vaikuntanathan. A group signature scheme from lattice assumptions. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 395–412. Springer, December 2010.
22. Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In *ANTS*, pages 267–288, 1998.
23. Fabien Laguillaumie, Adeline Langlois, Benoît Libert, and Damien Stehlé. Lattice-based group signatures with logarithmic signature size. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 41–61. Springer, December 2013.
24. San Ling, Khoa Nguyen, Damien Stehlé, and Huaxiong Wang. Improved zero-knowledge proofs of knowledge for the isis problem, and applications. In *Public Key Cryptography*, pages 107–124, 2013.
25. Vadim Lyubashevsky. Lattice-based identification schemes secure under active attacks. In *Public Key Cryptography*, pages 162–179, 2008.
26. Vadim Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In *ASIACRYPT*, pages 598–616, 2009.
27. Vadim Lyubashevsky. Lattice signatures without trapdoors. In *EUROCRYPT*, pages 738–755, 2012.
28. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. *J. ACM*, 60(6):43, 2013. Preliminary version appeared in EUROCRYPT 2010.
29. Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd ACM STOC*, pages 427–437. ACM Press, May 1990.
30. Torben P. Pedersen. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In *CRYPTO*, pages 129–140, 1991.

31. David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000.
32. Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th FOCS*, pages 543–553. IEEE Computer Society Press, October 1999.
33. Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In *CRYPTO*, pages 239–252, 1989.
34. Damien Stehlé and Ron Steinfeld. Making NTRU as secure as worst-case problems over ideal lattices. In *EUROCRYPT*, pages 27–47, 2011.
35. Jacques Stern. A new identification scheme based on syndrome decoding. In *CRYPTO*, pages 13–21, 1993.

A Definitions

A.1 Commitment Schemes

In the following we formally define commitment schemes.

Definition A.1. *A* commitment scheme *is a triple* $(\mathsf{CSetup}, \mathsf{Commit}, \mathsf{COpen})$ *such that:*

- *On input* 1^λ*, the key generation algorithm* CSetup *outputs commitment parameters* $cpars$*.*
- *The commitment algorithm* Commit *takes as inputs a message* m *from a message space* $\mathcal{M}$ *and commitment parameters* $cpars$*, and outputs a commitment/opening pair* (cmt, o)*.*
- *The opening algorithm* COpen *takes parameters* $cpars$*, a message* m*, a commitment* cmt*, and opening* o*, and outputs* `accept` *or* `reject`*.*

Furthermore, the algorithms satisfy the following properties:

- Correctness*:* COpen *outputs* `accept` *whenever the inputs were computed by an honest party, i.e.,*

$$\Pr\left[\mathsf{COpen}(cpars, m, cmt, o) = \texttt{accept} : \begin{array}{c} cpars \xleftarrow{\$} \mathsf{CSetup}(1^\lambda), m \in \mathcal{M}, \\ (cmt, o) \xleftarrow{\$} \mathsf{Commit}(cpars, m) \end{array}\right] = 1$$

- Binding*: A commitment cannot be opened to different messages. A scheme is said to be* computationally binding *if no PPT adversary can come up with a commitment and two different openings, i.e., for every PPT adversary* A *there exists a negligible function* negl *such that:*

$$\Pr\left[b = b' = \texttt{accept} \wedge m \neq m' : \begin{array}{c} cpars \xleftarrow{\$} \mathsf{CSetup}(1^\lambda), (cmt, m, o, m', o') \xleftarrow{\$} \mathsf{A}(cpars), \\ b = \mathsf{COpen}(cpars, m, cmt, o), \\ b' = \mathsf{COpen}(cpars, m', cmt, o') \end{array}\right] \leq \mathsf{negl}(\lambda)$$

  A scheme is said to be perfectly binding *if this holds unconditionally, i.e.,* $\mathsf{negl}(\lambda) = 0$*.*
- Hiding*: A commitment* computationally hides *the committed message: for every probabilistic polynomial time (PPT) adversary* A *there is a negligible function* negl *such that:*

$$\Pr\left[b = b' : \begin{array}{c} cpars \xleftarrow{\$} \mathsf{CSetup}(1^\lambda), (m_0, m_1, \mathsf{aux}) \xleftarrow{\$} \mathsf{A}_1(cpars), b \xleftarrow{\$} \{0,1\}, \\ (cmt, o) = \mathsf{Commit}(m_b, cpars), b' \xleftarrow{\$} \mathsf{A}_2(cmt, \mathsf{aux}) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)\,.$$

  A scheme is said to be perfectly hiding *if a commitment information-theoretically does not contain any information about the contained message, i.e.,* $\mathsf{negl}(\lambda) = 0$*.*

A.2 Semantically Secure Encryption

Definition A.2. *A semantically secure public key encryption scheme is a triple* $(\mathsf{EncKG}, \mathsf{Enc}, \mathsf{Dec})$ *satisfying the following:*

- *On input* 1^λ*,* EncKG *outputs a secret key/public key pair* (esk, epk)*.*
- *The encryption algorithm* Enc *takes the public key* epk *and a message* m *from some message space* $\mathcal{M}$*, and outputs a ciphertext* c*.*

– *The decryption algorithm* Dec *takes the secret key* esk *and a ciphertext* c *and returns a message* $m' \in \mathcal{M} \cup \{\perp\}$.

Furthermore, the algorithms have to satisfy the following properties:

– Correctness*:* Dec *returns* m *if* c *is an honest encryption of* m *with overwhelming probability. That is, there exists a negligible function* negl *such that:*

$$\Pr\left[m = m' : (esk, epk) \xleftarrow{\$} \mathsf{EncKG}(1^\lambda), m \in \mathcal{M}, c \xleftarrow{\$} \mathsf{Enc}(epk, m), m' = \mathsf{Dec}(esk, c)\right] \geq 1 - \mathsf{negl}(\lambda)$$

For every $m \in \mathcal{M}$ *we have that* $\mathsf{Dec}(esk, \mathsf{Enc}(epk, m)) = m$ *whenever* $(esk, epk) \xleftarrow{\$} \mathsf{EncKG}(1^\lambda)$

– Semantic security*: A scheme is called* semantically *(or* IND-CPA*) secure, if for every PPT adversary* $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ *there exists a negligible function* negl *such that:*

$$\Pr\left[b = b' : \begin{array}{c}(esk, epk) \xleftarrow{\$} \mathsf{EncKG}(1^\lambda), b \xleftarrow{\$} \{0,1\}, (m_0, m_1, \mathsf{aux}) \xleftarrow{\$} \mathsf{A}_1(epk) \\ c = \mathsf{Enc}(epk, m_b), b' \xleftarrow{\$} \mathsf{A}_2(\mathsf{aux}, c)\end{array}\right] \leq \mathsf{negl}(\lambda)\,.$$

A.3 Weakly Unforgeable Signatures

Definition A.3. *A* standard signature scheme *is a tuple* $(\mathsf{SKG}, \mathsf{SSign}, \mathsf{SVerify})$ *where:*

– *On input* 1^λ*, the key generation algorithm* SKG *outputs a public key* spk *and a signing key* ssk*.*
– *On input* ssk *and message* $m \in \mathcal{M}$*, the signing algorithm* SSign *outputs a signature* sig*.*
– *On input* spk, m, sig*, the verification algorithm* SVerify *outputs* `accept` *or* `reject`*.*

The algorithms satisfy the following properties:

– Correctness*:* SVerify *outputs* `accept` *whenever the keys and signature are honestly generated, i.e., for all* $\lambda \in \mathbb{N}$ *and all* $m \in \mathcal{M}$

$$\Pr[\mathsf{SVerify}(spk, m, sig) = \texttt{accept} : (spk, ssk) \xleftarrow{\$} \mathsf{SKG}(1^\lambda), sig \xleftarrow{\$} \mathsf{SSign}(ssk, m)] = 1\,.$$

– Weak unforgeability*: The signature scheme is* weakly unforgeable *if for every PPT adversary* A *there is a negligible function* negl *such that*

$$\Pr\left[\begin{array}{c}\mathsf{SVerify}(spk, m, sig) = \texttt{accept} \ \wedge \ m \notin \{m_1, \ldots, m_s\} : \\ (m_1, \ldots, m_s) \xleftarrow{\$} \mathsf{A}(1^\lambda), (spk, ssk) \xleftarrow{\$} \mathsf{SKG}(1^\lambda), \\ \textit{For } i = 1, \ldots, s \textit{ do } sig_i \xleftarrow{\$} \mathsf{SSign}(ssk, m_i), \\ (m, sig) \xleftarrow{\$} \mathsf{A}(spk, m_1, \ldots, m_s)\end{array}\right] \leq \mathsf{negl}(\lambda)\,.$$

A.4 Bilinear Pairings and the SDH Assumption

Let therefore q be prime, and $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be groups of order q. A mapping $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is called a bilinear pairing, if it satisfies the following:

1. Bilinearity: $\forall u \in \mathbb{G}_1, \forall v \in \mathbb{G}_2, \forall a, b \in \mathbb{Z}$ it holds that $e(u^a, v^b) = e(u, v)^{ab}$,
2. Non-degeneracy: if g_1 and g_2 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively, then $e(g_1, g_2)$ generates $\mathbb{G}_T$, and
3. Computability: all group operations and e can be efficiently computed.

The scheme presented by Boneh and Boyen is provable secure under the strong Diffie-Hellman assumption recapitulated next.

Definition A.4. *Let* BGGen*, the* bilinear group generator*, be a PPT algorithm that, on input* 1^λ*, outputs a prime* q*, groups* $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ *of order* q*, and a bilinear map* $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$*. We say that the* strong Diffie-Hellman (SDH) assumption *holds for* BGGen*, if for every PPT adversary* A *and every polynomially bounded function* k *there exists a negligible function* negl *such that the following holds:*

$$\Pr\left[(a_1, a_2) = (c, g_1^{\frac{1}{x+c}}) \in \mathbb{Z} \times \mathbb{G}_1 : \begin{array}{c}(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \xleftarrow{\$} \mathsf{BGGen}(1^\lambda), g_1 \xleftarrow{\$} \mathbb{G}_1, g_2 \xleftarrow{\$} \mathbb{G}_2, \\ x \xleftarrow{\$} \mathbb{Z}_q^\times, (a_1, a_2) \xleftarrow{\$} \mathsf{A}(g_1, g_1^x, \ldots, g_1^{(x^{k(\lambda)})}, g_2, g_2^x)\end{array}\right] \leq \mathsf{negl}(\lambda)\,.$$

B Generalized Forking Lemma

We make use of the forking lemma of Bellare and Neven [5], which is a generalized variant of the original forking lemma due to Pointcheval and Stern [31].

Lemma B.1. *Let q be an integer and let $\mathcal{C}$ be a set. Let A be a randomized algorithm that on input $in, h_1, \ldots, h_q$ returns a pair (I, out), where $0 \leq I \leq q$. Let IG be a randomized input generator algorithm. The* accepting probability *of A is*

$$acc \;=\; \Pr\left[I \geq 1 \;:\; in \xleftarrow{\$} \mathsf{IG},\, h_1, \ldots, h_q \xleftarrow{\$} \mathcal{C},\, (I, out) \xleftarrow{\$} \mathsf{A}(in, h_1, \ldots, h_q)\right] .$$

The forking algorithm $\mathsf{Fork}_{\mathsf{A}}(in)$ *associated to A first chooses random coins ρ for A and $h_1, \ldots, h_q \xleftarrow{\$} \mathcal{C}$. It then runs $\mathsf{A}(in, h_1, \ldots, h_q; \rho)$ to an integer I and side output out. If $I = 0$ then it returns $(0, \varepsilon, \varepsilon)$, otherwise it chooses fresh random $h'_I, \ldots, h'_q \xleftarrow{\$} \mathcal{C}$ and runs A again on input $(in, h_1, \ldots, h_{I-1}, h'_I, \ldots, h'_q)$ and on the same random tape ρ) until it outputs (I', out'). If $I = I'$ and $h_I \neq h'_I$, then $\mathsf{Fork}_{\mathsf{A}}$ outputs $(1, out, out')$, otherwise it returns $(0, \varepsilon, \varepsilon)$. Let*

$$frk \;=\; \Pr\left[b = 1 \;:\; in \xleftarrow{\$} \mathsf{IG}, (b, out, out') \xleftarrow{\$} \mathsf{Fork}_{\mathsf{A}}(in)\right] .$$

Then

$$frk \geq acc \cdot \left(\frac{acc}{q} - \frac{1}{\#\mathcal{C}}\right) .$$

Alternatively,

$$acc \leq \frac{q}{\#\mathcal{C}} + \sqrt{q \cdot frk} .$$

C Proofs

C.1 Proof of Lemma 2.3

Proof. Because $q = 3 \bmod 8$, the polynomial $X^n + 1$ splits into two irreducible polynomials modulo q. Now suppose that there are two possible plaintexts $hs + pe + m$ and $hs' + pe + m'$ that correspond to the same NTRU ciphertext c. Then $h(s - s') + p(e - e') + (m - m') = 0$. Since $X^n + 1$ splits into two irreducible polynomials of degree $n/2$ modulo q, we have that for a fixed $s - s', e - e', m - m'$, $Pr_h[h(s - s') + p(e - e') + (m - m') = 0] \leq \frac{1}{q^{n/2}}$. The claim in the lemma is obtained by applying the union bound. □

C.2 Proof of Theorem 4.2

Proof. We need to prove the properties of Definition 2.5.

Completeness. First note that by Theorem 2.4, the prover will respond with probability $1/M$ for each secret, i.e., with an overall probability of $1/M^3$. Assuming that the prover does not abort, we have that:

$$\begin{aligned} h\mathtt{s}_s + p\mathtt{s}_e + \mathtt{s}_m &= h(\mathtt{r}_s + X^{\mathsf{c}} s) + p(\mathtt{r}_e + X^{\mathsf{c}} e) + (\mathtt{r}_m + X^{\mathsf{c}} m) \\ &= X^{\mathsf{c}}(hs + pe + m) + (h\mathtt{r}_s + p\mathtt{r}_e + \mathtt{r}_m) = X^{\mathsf{c}} y + \mathtt{t} . \end{aligned}$$

For the norms it is easy to see that $\|\mathtt{s}_s\| \leq \|\mathtt{r}_s\| + \|s\| \leq \|\mathtt{r}_s\| + \sqrt{n}\alpha \leq n\alpha$ with overwhelming probability because the standard deviation of $\mathtt{r}_s$ is much smaller than $n\alpha$, and similar for $\mathtt{s}_e$ and $\mathtt{s}_m$.

What remains to show are the verification equations for the Pedersen commitments. First, note that as $q, \tilde{q}$ are much larger than the standard deviation of $\mathtt{r}_s, \mathtt{r}_m$, the computation of $\mathtt{s}_s$ and $\mathtt{s}_m$ is over the integers, i.e., with overwhelming probability there is no modular reduction. Furthermore, multiplying by X^{c} corresponds to anti-cyclically shifting the coefficients in the respective polynomial.

$$\begin{aligned} \mathtt{s}_m\tilde{g} + \mathtt{s}_r\tilde{h} &= ((\mathtt{r}_{m,0}, \ldots, \mathtt{r}_{m,n-1}) + (m_0, \ldots, m_{n-1})_{\ll \mathsf{c}})\,\tilde{g} + ((\mathtt{r}_{r,0}, \ldots, \mathtt{r}_{r,n-1}) + (r_0, \ldots, r_{n-1})_{\ll \mathsf{c}})\,\tilde{h} \\ &= (\mathtt{r}_{m,0}, \ldots, \mu_{m,n-1})\tilde{g} + (\mathtt{r}_{r,0}, \ldots, \mathtt{r}_{r,n-1})\tilde{h} + (m_0, \ldots, m_{n-1})_{\ll \mathsf{c}}\tilde{g} + (r_0, \ldots, r_{n-1})_{\ll \mathsf{c}}\tilde{h} \\ &= (\tilde{\mathtt{t}}_0, \ldots, \tilde{\mathtt{t}}_{n-1}) + \widetilde{cmt}_{\ll \mathsf{c}} \end{aligned}$$

Finally, by the completeness of the auxiliary commitment scheme, aCOpen will always output accept for an honest prover.

Honest-verifier zero-knowledge. The proof is a straightforward generalization of that for Theorem 3.3 and the basic Schnorr protocol [33], and therefore omitted.

Soundness. Assume that we have two accepting transcripts for the same first message but different challenges, i.e., that we have $(c_{\text{aux}}, \mathtt{c}', ((\mathtt{t}', (\tilde{\mathtt{t}}'_i)_{i=0}^{n-1}), d'_{\text{aux}}, (\mathtt{s}'_s, \mathtt{s}'_e, \mathtt{s}'_m, \mathtt{s}'_r))$ and $(c_{\text{aux}}, \mathtt{c}'', ((\mathtt{t}'', (\tilde{\mathtt{t}}''_i)_{i=0}^{n-1}), d''_{\text{aux}}, (\mathtt{s}''_s, \mathtt{s}''_e, \mathtt{s}''_m, \mathtt{s}''_r))$ that pass the checks performed by the verifier.

Then, by the binding property of the auxiliary commitment scheme, we have that $\mathtt{t}' = \mathtt{t}'' =: \mathtt{t}$ and $(\tilde{\mathtt{t}}'_i)_{i=0}^{n-1} = (\tilde{\mathtt{t}}''_i)_{i=0}^{n-1} =: (\tilde{\mathtt{t}}_i)_{i=0}^{n-1}$.

Now, by subtracting the verification equations, we get that:

$$(\widetilde{cmt}_0, \dots, \widetilde{cmt}_{n-1})_{\ll \mathtt{c}''} - (\widetilde{cmt}_0, \dots, \widetilde{cmt}_{n-1})_{\ll \mathtt{c}'} = (\mathtt{s}''_m - \mathtt{s}'_m)\tilde{g} + (\mathtt{s}''_r - \mathtt{s}'_r)\tilde{h}\,,$$
$$(X^{\mathtt{c}''} - X^{\mathtt{c}'})y = h(\mathtt{s}''_s - \mathtt{s}'_s) + p(\mathtt{s}''_e - \mathtt{s}'_e) + (\mathtt{s}''_m - \mathtt{s}'_m)\,.$$

By the symmetry of anti-cyclic shifts and multiplication by powers of X, this is easily seen to be equivalent to:

$$(\widetilde{cmt}_0, \dots, \widetilde{cmt}_{n-1})C = (\mathtt{s}''_m - \mathtt{s}'_m)\tilde{g} + (\mathtt{s}''_r - \mathtt{s}'_r)\tilde{h}\,, \tag{1}$$
$$yC = h(\mathtt{s}''_s - \mathtt{s}'_s) + p(\mathtt{s}''_e - \mathtt{s}'_e) + (\mathtt{s}''_m - \mathtt{s}'_m)\,, \tag{2}$$

where the i^{th} column of the matrix C contains the coefficients of $(X^{\mathtt{c}'} - X^{\mathtt{c}''}) \mod X^n + 1$. Note that as $\|\mathtt{s}'_m\|, \|\mathtt{s}''_m\|$ are small, the coefficients of $\mathtt{s}'_m - \mathtt{s}''_m$ in R_q and $\mathbb{Z}_{\tilde{q}}$ are the same over the integers, as no modular reduction takes place here.

By Lemma 3.1 we have that $2(X^{\mathtt{c}'} - X^{\mathtt{c}''})$ is invertible in $\mathbb{Z}/(X^n + 1)$, and the inverse only has coefficients in $\{1, 0, 1\}$. Let d be the inverse, and D denote the corresponding matrix. Since 2 is coprime with q and $\tilde{q}$, D can also be seen as a matrix with entries in R_q and $\mathbb{Z}_{\tilde{q}}$. By multiplying (1) and (2) by D from the right, we get that $\hat{m} = 2(\mathtt{s}'_m - \mathtt{s}''_m)d$, $\hat{r} = 2(\mathtt{s}'_r - \mathtt{s}''_r)D$, $\hat{e} = 2(\mathtt{s}'_e - \mathtt{s}''_e)d$ and $\hat{s} = 2(\mathtt{s}'_s - \mathtt{s}''_s)d$ satisfy the following:

$$h\hat{s} + p\hat{e} + \hat{m} = 2y$$
$$\hat{m}_i g + \hat{r}_i h = 2\mathtt{t}_i \quad \text{for} \quad 0 \le i < n,$$

and the norms of $\hat{e}, \hat{m}, \hat{s}$ satisfy $\|\hat{m}\|_\infty \le n(\|\mathtt{s}'_m\|_\infty + \|\mathtt{s}''_m\|_\infty) \le 2n^2$, and similarly for $\hat{e}$ and $\hat{s}$. Note again that as $q, \tilde{q} > 2n^2$, $\hat{m}$ is the same in R_q and $\mathbb{Z}_{\tilde{q}}$ over the integers.

Quasi-unique responses. Assume that a PPT adversary outputs $(c_{\text{aux}}, \mathtt{c}, (\mathtt{t}', \tilde{\mathtt{t}}'_0, \dots, \tilde{\mathtt{t}}'_{n-1}), d'_{\text{aux}}, (\mathtt{s}'_s, \mathtt{s}'_e, \mathtt{s}'_m, \mathtt{s}'_r))$ and $(c_{\text{aux}}, \mathtt{c}, (\mathtt{t}'', \tilde{\mathtt{t}}''_0, \dots, \tilde{\mathtt{t}}''_{n-1}), d''_{\text{aux}}, (\mathtt{s}''_s, \mathtt{s}''_e, \mathtt{s}''_m, \mathtt{s}''_r))$ that pass the tests performed by the verifier. By the strong binding requirements on the auxiliary commitment scheme, we get that $\mathtt{t}' = \mathtt{t}'' =: \mathtt{t}$, $\tilde{\mathtt{t}}'_i = \tilde{\mathtt{t}}''_i =: \tilde{\mathtt{t}}_i$ for all i, and $d'_{\text{aux}} = d''_{\text{aux}}$. What remains to show is that $\mathtt{s}_s, \mathtt{s}_e, \mathtt{s}_m, \mathtt{s}_r$ are unique as well.

We have that $0 = (\mathtt{s}'_m - \mathtt{s}''_m)\tilde{g} + (\mathtt{s}'_r - \mathtt{s}''_r)\tilde{h}$. Not that for every i, $\mathtt{s}'_{m,i} \neq \mathtt{s}''_{m,i}$ if and only if $\mathtt{s}'_{r_i} \neq \mathtt{s}''_{r_i}$. In this case we get that $\log_{\tilde{g}} \tilde{h} = -\frac{\mathtt{s}'_m - \mathtt{s}''_m}{\mathtt{s}'_r - \mathtt{s}''_r}$, which any PPT adversary can only compute with negligible probability under the assumed DLOG assumption.

We further get that $\mathtt{s}'_s = \mathtt{s}''_s$, $\mathtt{s}'_m = \mathtt{s}''_m$ and $\mathtt{s}'_e = \mathtt{s}''_e$ in R_q, as otherwise we had that $h(\mathtt{s}'_s - \mathtt{s}''_s) + p(\mathtt{s}'_e - \mathtt{s}''_e) + (\mathtt{s}'_m - \mathtt{s}''_m) = 0$. However, if an adversary could come up with such values, it would be able to distinguish uniformly random h from NTRU public keys, as by Lemma 2.3 there does not exist a non-trivial solution to the last equation with overwhelming probability for truly random h.

High-entropy commitments. This directly follows from the security of the auxiliary commitment scheme. □

C.3 Proof of Theorem 5.3

Proof. We prove the theorem by describing a sequence of indistinguishable games, where the adversary's view in the final game is completely independent of the hidden bit b chosen by the experiment.

Game 0: This is the original game as described in Figure 1.

Game 1: We now use the simulator SoKSim to generate the signature of knowledge sok^* in the challenge signature $\sigma^* = (C_1^*, C_2^*, sok^*)$. Meaning, we use StpSim to generate the parameters $sokp$, we use ROSim to respond to A's random-oracle queries, and we compute sok^* as $\mathsf{SSim}((spk, epk_1, epk_2, C_1, C_2), (i, \mathbf{gsk}[i_b^*], \rho_1, \rho_2), m^*)$. It is easy to see that any adversary A distinguishing Game 0 from Game 1 can be used to break the simulatability of the signature of knowledge scheme.

Game 2: Instead of computing sok^* using SSim, we generate it as $\mathsf{SSim}'((spk, epk_1, epk_2, C_1, C_2), m^*)$, i.e., without first checking language membership. Since we only call SSim' on actual language elements anyway, this change is purely conceptual, so this game is identical to the previous game from the point of view of the adversary.

Game 3: In the computation of the target signature $\sigma^* = (C_1^*, C_2^*, sok^*)$, we compute C_2^* as $\mathsf{Enc}(epk_2, 1)$ instead of $\mathsf{Enc}(epk_2, i_b^*)$. Any adversary distinguishing this game from the previous one can be used to break the semantic security of the encryption scheme. Note that we continue to simulate the opening oracle using esk_1, as in the real scheme, so we never need to decrypt any ciphertexts under epk_2. Also note that now sok^* is actually a signature of knowledge on a false statement, because C_1^* and C_2^* encrypt different user identities.

Game 4: We now respond to A's opening queries by decrypting C_2 using esk_2 instead of decrypting C_1 using esk_1. This does not change the view of the adversary as long as all pairs (C_1, C_2) included in A's opening queries decrypt to the same plaintext. The simulation-soundness guarantees that, except with negligible probability, all group signatures include actual language elements, meaning that there exist i, ρ_1, ρ_2 such that $\Phi(C_1) = \mathsf{Enc}(epk_1, \varphi(i); \rho_2$ and $\Phi(C_2) = \mathsf{Enc}(epk_2, \varphi(i); \rho_2)$. Due to the perfect completeness of the encryption scheme, this means that both ciphertexts decrypt to the same plaintext d. Due to the injectivity of φ, they therefore lead to the same i such that $\varphi(i) = d$.

Game 5: We now also replace C_1^* with an encryption of 1 instead of b. Since we no longer decrypt anything under esk_1 during the simulation, this game is indistinguishable from the previous one by the semantic security of the encryption scheme. Note that at this point, the simulation is completely independent of the bit b, so that A has exactly the same probability to output 1 for either choice of b. □

C.4 Proof of Theorem 5.4

Proof. We distinguish between the case that A wins because $\mathsf{GOpen}(gok, m, \sigma) = \perp$, and the case that it does so because $\mathsf{GVerify}(gpk, m, \sigma) = 1 \ \wedge \ i \notin \mathcal{Q}_{\mathbf{gsk}} \ \wedge \ (i, m) \notin \mathcal{Q}_{\mathsf{GSign}} \ \wedge \ i \neq \perp$.

In the former case, simulatability and extractability together mean that, except with negligible probability, there must exist i, sig, ρ_1 such that $\mathsf{SVerify}(spk, i, sig) = \mathtt{accept}$ and $\Phi(C_1) = \mathsf{Enc}(epk_1, \varphi(i); \rho_1)$. By the perfect completeness of the encryption scheme, such a ciphertext cannot fail to decrypt under esk_1 to some plaintext d. The only way for opening to fail is if there does not exist $i' \in \mathcal{ID}$ such that $d = \varphi(i')$, which means that $i \notin \mathcal{ID}$, meaning that sig must be a forgery.

In the latter case, consider the following sequence of games.

Game 0: This is the real traceability game.

Game 1: We now simulate random-oracle responses and the signatures of knowledge included in issued group signatures with SoKSim. This game is indistinguishable from Game 0 by the simulatability of the signature of knowledge scheme.

Game 2: We now use the extractor SoKExt on the signature of knowledge in the forgery to obtain (sig, i) such that $\mathsf{SVerify}(spk, i, sig) = \mathtt{accept}$. By the extractability property, the extractor succeeds in doing so with non-negligible probability.

Game 3: We now simulate all signatures of knowledge in GSign responses using SSim', i.e., without checking the witness. This game is identical to the previous one since we call SSim only on valid language elements. Note that the simulation no longer requires to compute standard signatures on the user identities i.

Given an adversary A with non-negligible probability of winning in Game 3, consider the following adversary B against the weak unforgeability of the standard signature scheme. Algorithm B guesses an identity $i^* \xleftarrow{\$} \{1, \ldots, n\}$ and asks for signatures on all identities $1, \ldots, n$ except i^*. It runs A, using the challenge public key spk as part of the group public key gpk, and simulating random-oracle responses and signatures of knowledge

in group signatures with SoKSim. When A outputs its forgery, B extracts (sig, i). If $i = i^*$, B outputs (sig, i) as its own forgery. □

C.5 Proof of Theorem 5.5

Proof. The SoKVerify algorithm only rejects if V_s or one of the two executions of V_e reject. Given that they each reject at most with negligible probability α, SoKVerify rejects with probability at most 3α. □

C.6 Proof of Theorem 5.6

Proof. We prove the theorem through a sequence of indistinguishable games, where the first game is identical to $\mathsf{A}^{H(\cdot),\mathsf{SoKSign}(\cdot,\cdot,\cdot)}(sokp)$ with real parameters $sokp$ and the last is identical to $\mathsf{A}^{\mathsf{ROSim}(\cdot),\mathsf{SSim}(\cdot,\cdot,\cdot)}(sokp)$ with simulated parameters.

Game 0: This is the original game $\mathsf{A}^{H(\cdot),\mathsf{SoKSign}(\cdot,\cdot,\cdot)}(sokp)$ with $sokp \xleftarrow{\$} \mathsf{SoKSetup}(1^\lambda)$.

Game 1: Generate $sokp \xleftarrow{\$} \mathsf{StpSim}(1^\lambda)$ and respond to A's random-oracle queries, as well as the random-oracle queries made by the SoKSign oracle, using $\mathsf{ROSim}(\cdot)$. This game is identical to the previous game since both generate parameters through $\mathsf{CSetup}(1^\lambda)$ and both return truly random responses to random-oracle queries.

Game 2: If the random-oracle query caused by an invocation of the SoKSign oracle was made previously, either by the adversary, or by another SoKSign query, then the simulation is aborted. This happens with negligible probability due to the high-entropy commitments of the proof systems $(\mathsf{P}_\mathrm{s}, \mathsf{V}_\mathrm{s})$ and $(\mathsf{P}_\mathrm{e}, \mathsf{V}_\mathrm{e})$.

Game 3: Respond to all signature queries $\mathsf{SoKSign}(x, w, m)$ with $\mathsf{SSim}(x, w, m)$. We prove that Game 2 is indistinguishable from Game 1 through a hybrid argument on the signature queries of A. Let Game 2.j be the game where A's first j signature queries are responded to with SSim, while all later queries are responded to with SoKSign. Clearly, we have that Game 2.0 is identical to Game 2 and Game 2.q_S to Game 3. Suppose that A can distinguish with non-negligible probability between Game 2.j and Game 2.$(j+1)$. Then consider the following sequence of sub-games.

- **Game 2.j.0:** Identical to Game 2.j
- **Game 2.j.1:** When responding to the $j+1$st SSim query, rather than computing $\mathtt{t}_\mathrm{s}, \mathtt{s}_\mathrm{s}$ through P_s, use the simulator S_s using the same challenge value c_s. This game is indistinguishable from Game 2.j.0 by the special honest-verifier zero-knowledge property of $(\mathsf{P}_\mathrm{s}, \mathsf{V}_\mathrm{s})$.
- **Game 2.j.2:** Similarly, use S_e instead of P_e to generate $\mathtt{t}_1, \mathtt{s}_1, \mathtt{t}_2, \mathtt{s}_2$. Indistinguishability follows from the honest-verifier zero-knowledge property of $(\mathsf{P}_\mathrm{s}, \mathsf{V}_\mathrm{s})$.
- **Game 2.j.3:** Finally, generate the commitment cmt as a commitment of 1 instead of of i. Indistinguishability follows from the hiding property of the commitment scheme. □

C.7 Proof of Theorem 5.7

Proof. Consider the algorithm B that runs the adversary A against a simulated random oracle and signing oracle. More precisely, on input $1^\lambda, h_1, \ldots, h_q \xleftarrow{\$} \mathcal{C}$, and random tape $(\rho_\mathsf{S}, \rho_\mathsf{A})$, algorithm B runs $sokp \xleftarrow{\$} \mathsf{StpSim}(1^\lambda; \rho_\mathsf{S})$ and $(x^*, m^*, sok^*) \xleftarrow{\$} \mathsf{A}^{\mathsf{ROSim}(\cdot),\mathsf{SSim}(\cdot,\cdot,\cdot)}(sokp; \rho_\mathsf{A})$, using the values $h_1, \ldots, h_q$ as random-oracle responses. Let $Q^* = (spk, cpars, cmt, epk_1, C_1, epk_2, C_2, \mathtt{t}_\mathrm{s}, \mathtt{t}_1, \mathtt{t}_2, m^*)$ be the "crucial" random oracle query that is involved in the computation and verification of sok^*. We can assume without loss of generality that A makes this query before it stops executing, so let $I = HT[Q^*]$ be the index of the returned random-oracle response. If $\mathsf{SoKVerify}(sokp, x^*, m^*, sok^*) = \mathtt{accept}$ and A did not make a signature query (x^*, w, m^*) for any w such that $(x^*, w) \in R_\mathcal{L}$, then B outputs $(I, (x^*, m^*, sok^*))$, otherwise it outputs $(0, \varepsilon)$.

Using the generalized forking lemma, recalled in Lemma B.1, for any algorithm B that outputs (I, out^*) with probability acc, the forking algorithm Fork_B returns two accepting outputs out^*, out'^* corresponding to random-oracle responses $h_I \neq h'_I$ with probability

$$frk \geq acc \cdot \left(\frac{acc}{q} - \frac{1}{\#\mathcal{C}} \right) .$$

Let $\mathsf{SoKExt}_{\mathsf{A}}(sokp, x, m, sok, \rho_{\mathsf{S}}, \rho_{\mathsf{A}})$ be an algorithm that runs the forking algorithm as described above to obtain two valid signatures of knowledge $(x^*, m^*, sok^*), (x'^*, m'^*, sok'^*)$ for two different random-oracle responses $h_I \neq h'_I$. Since both runs of A are identical up to the I-th random-oracle response, the arguments Q^* and Q^* to A's I-th random-oracle query are identical as well. Because these arguments contain the full descriptions of the language elements $x^*, x^*_{\mathrm{s}}, x^*_1, x^*_2$, we know that these are the same for both runs, as well as the message m^* and the first rounds of the Σ-protocols $\mathtt{t}^*_{\mathrm{s}}, \mathtt{t}^*_1, \mathtt{t}^*_2$. Because $\mathsf{SoKVerify}(sokp, x^*, m^*, sok^*) = \mathsf{SoKVerify}(sokp, x^*, m^*, sok'^*) = \mathtt{accept}$, we therefore have two valid transcripts $(\mathtt{t}^*_{\mathrm{s}}, \mathtt{c}^*_{\mathrm{s}}, \mathtt{s}^*_{\mathrm{s}}), (\mathtt{t}^*_{\mathrm{s}}, \mathtt{c}'^*_{\mathrm{s}}, \mathtt{s}'^*_{\mathrm{s}})$ for $\mathsf{V}_{\mathrm{s}}((spk^*, cpars^*, cmt^*), \cdot)$ and two valid transcripts $(\mathtt{t}^*_j, \mathtt{c}^*_{\mathrm{e}}, \mathtt{s}^*_j), (\mathtt{t}^*_j, \mathtt{c}'^*_{\mathrm{e}}, \mathtt{s}'^*_j)$ for $\mathsf{V}_{\mathrm{e}}((epk^*_j, C^*_j, cpars^*, cmt^*), \cdot)$ for each of $j = 1, 2$.

If now $\mathtt{c}^*_{\mathrm{s}} \neq \mathtt{c}'^*_{\mathrm{s}}$ and $\mathtt{c}^*_{\mathrm{e}} \neq \mathtt{c}'^*_{\mathrm{e}}$, then we can use the special soundness of the proof systems and extract witnesses (sig, i, o) and (i', ρ_j, o') for $j = 1, 2$ using the knowledge extractors $\mathsf{E}_{\mathrm{s}}, \mathsf{E}_{\mathrm{e}}$. By the binding property of the commitment scheme, we have that $\varphi(i) = i'$, because otherwise $(\varphi(i), \chi(o))$ and (i', o') would be valid openings to different messages of the same commitment cmt^*. From the extracted witnesses, we can therefore compose a valid witness (i, sig, ρ_1, ρ_2) for $x^* \in \mathcal{L}$.

However, the forking lemma only guarantees that $h_I = (\mathtt{c}^*_{\mathrm{s}}, \mathtt{c}^*_{\mathrm{e}}) \neq h'_I = (\mathtt{c}'^*_{\mathrm{s}}, \mathtt{c}'^*_{\mathrm{e}})$, which does not necessarily imply that $\mathtt{c}^*_{\mathrm{s}} \neq \mathtt{c}'^*_{\mathrm{s}}$ and $\mathtt{c}^*_{\mathrm{e}} \neq \mathtt{c}'^*_{\mathrm{e}}$. Since all random-oracle responses are chosen at random, the probability that any of the components is equal is

$$\Pr\left[\mathtt{c}^*_{\mathrm{s}} = \mathtt{c}'^*_{\mathrm{s}} \vee \mathtt{c}^*_{\mathrm{e}} = \mathtt{c}'^*_{\mathrm{e}}\right] \leq \Pr\left[\mathtt{c}^*_{\mathrm{s}} = \mathtt{c}'^*_{\mathrm{s}}\right] + \Pr\left[\mathtt{c}^*_{\mathrm{e}} = \mathtt{c}'^*_{\mathrm{e}}\right] = \frac{1}{\#\mathcal{C}_{\mathrm{s}}} + \frac{1}{\#\mathcal{C}_{\mathrm{e}}}\,.$$

The probability that $\mathsf{SoKExt}_{\mathsf{A}}$ succeeds is therefore at least

$$\begin{aligned} frk - \frac{1}{\#\mathcal{C}_{\mathrm{s}}} - \frac{1}{\#\mathcal{C}_{\mathrm{e}}} &\geq acc \cdot \left(\frac{acc}{q} - \frac{1}{\#\mathcal{C}}\right) - \frac{1}{\#\mathcal{C}_{\mathrm{s}}} - \frac{1}{\#\mathcal{C}_{\mathrm{e}}} \\ &\geq \frac{acc^2}{q} - \frac{3}{\min(\#\mathcal{C}_{\mathrm{s}}, \#\mathcal{C}_{\mathrm{e}})}\,. \end{aligned}$$

If acc is non-negligible, then the first term is non-negligible. If the challenge spaces $\mathcal{C}_{\mathrm{s}}, \mathcal{C}_{\mathrm{e}}$ are super-polynomial, then the second term is negligible, so that overall $\mathsf{SoKExt}_{\mathsf{A}}$ succeeds with non-negligible probability. □

C.8 Proof of Theorem 5.8

Proof. Suppose there exists an adversary A that breaks the simulation-soundness with non-negligible probability. Without loss of generality, we assume that A makes both random-oracle queries Q, Q' involved in the verification $\mathsf{SoKVerify}(sokp, x, m, sok)$ and $\mathsf{SoKVerify}(sokp, x', m', sok')$. Let $sok = (\mathtt{t}_{\mathrm{s}}, \mathtt{t}_1, \mathtt{t}_2, \mathtt{s}_{\mathrm{s}}, \mathtt{s}_1, \mathtt{s}_2)$ and $sok' = (\mathtt{t}'_{\mathrm{s}}, \mathtt{t}'_1, \mathtt{t}'_2, \mathtt{s}'_{\mathrm{s}}, \mathtt{s}'_1, \mathtt{s}'_2)$.

If $Q = Q'$, then we have that $(x, m, \mathtt{t}_{\mathrm{s}}, \mathtt{t}_1, \mathtt{t}_2) = (x, m, \mathtt{t}_{\mathrm{s}}, \mathtt{t}_1, \mathtt{t}_2)$, since all of these items are part of the random-oracle query. Since $sok = sok'$, it must hold that $(\mathtt{s}_{\mathrm{s}}, \mathtt{s}_1, \mathtt{s}_2) \neq (\mathtt{s}'_{\mathrm{s}}, \mathtt{s}'_1, \mathtt{s}'_2)$. We therefore have for at least one of the three Σ-protocols two accepting transcripts with the same commitment and challenge but different responses, which violates the quasi-unique responses of the proof systems.

If $Q \neq Q'$, then by a similar reasoning as in the proof of Theorem 5.7, the forking lemma (Lemma B.1) allows us to extract with non-negligible probability two valid signatures sok', sok'' for the same statement x' but for different random-oracle responses, if the commitment scheme is binding and the proof systems have super-polynomial challenge spaces. From these, a witness for x' can be extracted, which contradicts that $x' \notin \mathcal{L}$. □

C.9 Proof of Theorem 5.11

Proof. We need to prove completeness, honest-verifier zero-knowledge and soundness. As the proof is standard, we only sketch it here.

Completeness. This is trivial to verify.

Honest-verifier zero-knowledge. Given a challenge $\mathtt{c}$, the simulator S draws $\tilde{s}' \xleftarrow{\$} \mathbb{G}_1$ and $\mathtt{s}'_d, \mathtt{s}'_{m,i}, \mathtt{s}'_{r,i} \xleftarrow{\$} \mathbb{Z}_{\tilde{q}}$. It computes $\tilde{\mathtt{t}}' = e(\tilde{s}', \tilde{v})^{-\mathtt{c}} \tilde{z}^{\mathtt{s}'_d} e(\tilde{s}', \tilde{g}_2)^{-\sum_{i=0}^{n-1} 2^i \mathtt{s}'_{m,i}}$ and $\tilde{\mathtt{t}}'_i = -\mathtt{c}\widetilde{cmt}_i + \mathtt{s}'_{m,i}\tilde{g} + \mathtt{s}'_{r,i}\tilde{h}$. The simulator outputs

$((\tilde{s}', \tilde{\mathtt{t}}', (\tilde{\mathtt{t}}'_i)_{i=0}^{n-1}), \mathtt{c}, (\mathtt{s}'_d, (\mathtt{s}'_{m,i}, \mathtt{s}'_{r,i})_{i=0}^{n-1}))$. By standard arguments one can see that the output distribution is identical to that of real protocol runs for challenge $\mathtt{c}$.

Soundness. Using standard arguments one can show that, given two accepting protocol transcripts with the same first message but different challenges, one can efficiently extract a tuple $(d', (m'_i)_{i=0}^{n-1}, (r'_i)_{i=0}^{n-1})$ such that $e(\tilde{s}, \tilde{v}) = \tilde{z}^{d'} e(\tilde{s}, \tilde{g}_2)^{-\sum_{i=0}^{n-1} 2^i m'_i}$ and $\widetilde{cmt}_i = m'_i\tilde{g} + r'_i\tilde{h}$ for $i = 0, \ldots, n-1$. Let now $m' = \sum_{i=0}^{n-1} 2^i m'_i$. If $d' \neq 0$, setting $\widetilde{sig}' = \tilde{s}^{1/d'}$ yields $e(\widetilde{sig}'^{d'}, \tilde{v}) = \tilde{z}^{d'} e(\widetilde{sig}'^{d'}, \tilde{g}_2)^{-m'}$. This is equivalent to $e(\widetilde{sig}', \tilde{v}) = \tilde{z} e(\widetilde{sig}, \tilde{g}_2)^{-m'}$ or $e(\widetilde{sig}', \tilde{v}\tilde{g}_2^{m'}) = \tilde{z}$, and thus $\widetilde{sig}'$ is a valid signature for m'. If $d' = 0$, we get that $e(\tilde{s}, \tilde{v}) = e(\tilde{s}, \tilde{g}_2)^{-m'}$ or $e(\tilde{s}, \tilde{v}\tilde{g}_2^{m'}) = 1$, and therefore $\tilde{v}\tilde{g}_2^{m'} = 1$ because $\tilde{s} \neq 1$. In this case setting $\widetilde{sig}' = 1$ gives a valid signature for m'.

Quasi-unique responses. Assume that for some $\tilde{\mathtt{t}}, (\tilde{\mathtt{t}}_i)_{i=0}^{n-1}, \mathtt{c}$, a PPT adversary A outputs $(\mathtt{s}_d, (\mathtt{s}_{m,i}, \mathtt{s}_{r,i})_{i=0}^{n-1}), (\mathtt{s}'_d, (\mathtt{s}'_{m,i}, \mathtt{s}'_{r,i})_{i=0}^{n-1})$ that pass the verification checks. Define $\mathtt{s}_m = \sum_{i=0}^{n-1} \mathtt{s}_{m,i}$, and similarly for $\mathtt{s}'_m$. By dividing the verification equations we get that:

$$1 = \tilde{z}^{\mathtt{s}_d - \mathtt{s}'_d} e(\tilde{s}, \tilde{g}_2)^{\mathtt{s}'_m - \mathtt{s}_m} \quad \text{and} \quad 1 = (\mathtt{s}_{m,i} - \mathtt{s}'_{m,i})\tilde{g} + (\mathtt{s}_{r,i} - \mathtt{s}'_{r,i})\tilde{h}\,.$$

It follows that $\mathtt{s}_{m,i} = \mathtt{s}'_{m,i}$ and $\mathtt{s}_{r,i} = \mathtt{s}'_{r,i}$ for all i with overwhelming probability, as otherwise A could compute $\log_{\tilde{g}} \tilde{h} = -\frac{\mathtt{s}_{m,i} - \mathtt{s}'_{m,i}}{\mathtt{s}_{r,i} - \mathtt{s}'_{r,i}}$. Therefore, also $\mathtt{s}_m = \mathtt{s}'_m$, and thus $\mathtt{s}_d = \mathtt{s}'_d$ because $\tilde{z}$ is a generator.

High-entropy commitments. This follows immediately from the fact that each element sent in the first message is independently and uniformly random in its domain. □