Efficient Lattice-Based Zero-Knowledge Arguments with Standard Soundness: Construction and Applications

Rupeng Yang[1,2], Man Ho Au[2 ⋆], Zhenfei Zhang[3], Qiuliang Xu[4 ⋆], Zuoxia Yu[2], and William Whyte[5]

[1] School of Computer Science and Technology, Shandong University, Jinan, 250101, China
orbbyrp@gmail.com

[2] Department of Computing, The Hong Kong Polytechnic University, Hung Hom, Hong Kong
csallen@comp.polyu.edu.hk, zuoxia.yu@gmail.com

[3] Algorand, USA
zhenfei@algorand.com

[4] School of Software, Shandong University, Jinan, 250101, China
xql@sdu.edu.cn

[5] Qualcomm Technologies Incorporated, USA
wwhyte@qti.qualcomm.com

Abstract. We provide new zero-knowledge argument of knowledge systems that work directly for a wide class of language, namely, ones involving the satisfiability of matrix-vector relations and integer relations commonly found in constructions of lattice-based cryptography. Prior to this work, practical arguments for lattice-based relations either have a constant soundness error ($2/3$), or consider a weaker form of soundness, namely, extraction only guarantees that the prover is in possession of a witness that "approximates" the actual witness. Our systems do not suffer from these limitations.

The core of our new argument systems is an efficient zero-knowledge argument of knowledge of a solution to a system of linear equations, where variables of this solution satisfy a set of quadratic constraints. This argument enjoys standard soundness, a small soundness error ($1/poly$), and a complexity linear in the size of the solution. Using our core argument system, we construct highly efficient argument systems for a variety of statements relevant to lattices, including linear equations with short solutions and matrix-vector relations with hidden matrices.

Based on our argument systems, we present several new constructions of common privacy-preserving primitives in the *standard lattice* setting, including a group signature, a ring signature, an electronic cash system, and a range proof protocol. Our new constructions are one to three orders of magnitude more efficient than the state of the art (in standard lattice). This illustrates the efficiency and expressiveness of our argument system.

⋆ Corresponding author.

1 Introduction

Traditional cryptographic schemes based on number theoretic assumptions are at risk due to possible attacks from quantum computers. Among all alternatives, the lattice-based ones appear to be the most promising. To date, we have good candidates to fundamental cryptographic primitives such as public key encryption schemes (e.g., [6, 18, 19, 45]) and signature schemes (e.g., [15, 31, 35, 64]). However, lattice-based privacy-preserving primitives, such as group signatures [26], ring signatures [75], electronic cash (E-cash) [25], etc., are still significantly less efficient than their traditional counterparts, partially due to the lack of suitable lattice-based zero-knowledge proofs. Specifically, current zero-knowledge proofs for lattice-based relations either have a poor efficiency or have great restrictions when employed in constructing advanced applications.

The study of lattice-based zero-knowledge proofs is initialized by Goldreich and Goldwasser in [38]. Goldreich and Goldwasser's proof system, as well as proof systems developed in subsequent works [3, 43, 69, 73], are mainly of theoretical interest. While one can construct applications such as verifiable encryption [40] and group signature [24, 41] from these protocols, their lack of efficiency prevents them from being employed in practice.

For practical lattice-based zero-knowledge proofs, there are two main approaches in current literature.

Stern-type Protocol. One approach, which follows techniques in [47, 77], is proposed by Ling et al. in [58]. They construct an efficient zero-knowledge argument of knowledge (`ZKAoK`) for the basic Inhomogeneous Short Integer Solution (ISIS) relation $\mathcal{R}_{ISIS} = \{(\boldsymbol{A}, \boldsymbol{y}), \boldsymbol{x} : \boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{y} \wedge \|\boldsymbol{x}\| \leq \beta\}$. Focusing on arguing additional relations over witnesses, `ZKAoK`s for a wider class of lattice-based relations are constructed in subsequent works. This gives rise to various applications, such as verifiable encryption [58], group signature [51, 52, 54, 59, 61], ring signature [54], group encryption [53] and E-cash [55].

The major issue for Stern-type protocols is their inherent *large soundness error*. More precisely, a single round Stern-type protocol has a soundness error of $2/3$, i.e., a cheating prover is able to convince an honest verifier with probability $2/3$ even if it does not possess any valid witness. Thus, to achieve a negligible soundness error, the protocol is required to repeat for many (e.g., 219) times, and the final proof consists of proofs generated in all iterations. Consequently, its proof size is usually on the order of tens of megabytes to terabytes.

Fiat-Shamir with Abort. Another line of research follows the identification schemes from [62–64]. Early works in this direction [50, 70] consider `ZKAoK` protocols with binary challenges, which leads to a soundness error of $1/2$ for a single iteration. Thus, multiple (e.g., 128) repetitions are needed to achieve a negligible soundness error. Subsequently, `ZKAoK`s with larger challenge spaces are adopted to reduce the number of rounds required. This results in one-round protocols with inverse-polynomial/negligible soundness error. Thus, we only need to run them a few (e.g., 10 or even 1) time(s) to achieve a negligible soundness error. Consequently, the proof size is usually a few megabytes or less.

We have seen some applications, such as verifiable encryption [13, 65], group signature [20, 21, 29] and ring signature [33] from Fiat-Shamir with abort (FSwA) protocols (with large challenge space). However, it is a complex task to design cryptographic protocols using FSwA. This is mainly due to the so-called *soundness gap*. For instance, for the ISIS relation $\mathcal{R}_{ISIS}$, the FSwA proof only attests the fact that the prover knows a witness for $\mathcal{R}'_{ISIS} = \{(\boldsymbol{A}, \boldsymbol{y}), \boldsymbol{x} : \boldsymbol{A} \cdot \boldsymbol{x} = c \cdot \boldsymbol{y} \wedge \|\boldsymbol{x}\| \leq \beta'\}$, where $\beta' > \beta$ and $c > 1$. Thus, to construct advanced applications from them, we have to use cryptographic primitives that are compatible with such relaxed soundness, e.g., encryption schemes with a relaxed decryption [13, 65], commitment schemes with a relaxed opening [10, 14] and signature schemes with a relaxed verification [21]. Unfortunately, it is usually hard or even impossible to construct primitives with such property. Meanwhile, general frameworks in the literature for advanced applications may not work when we use relaxed versions as building blocks. Thus, the construction and security analysis has to be conducted from scratch. Additionally, we do not have a simple manner to prove the relations over witnesses using Fiat-Shamir with abort protocols. Ad-hoc techniques are used to circumvent this requirement, which introduce additional complexity.

To summarize, we have some "user-friendly" lattice-based ZKAoKs that are less efficient; and some efficient ZKAoKs that are very complicated for advanced applications. [1] The goal of this paper is, therefore, to construct ZKAoKs that are both efficient and easy to use.

On the Difficulty of Achieving Standard Soundness and Small Soundness Error. Before presenting our main results, we would like to discuss why previous works cannot achieve the standard soundness and a small soundness error simultaneously. First, for most (if not all) lattice-based relations, we need to prove that (parts of) the witnesses are small integers. This can be done in two approaches,

1. In a Stern-type protocol, a short integer is decomposed into a binary vector of bounded length. Then the prover proves that the decomposition outputs are correct via a standard Stern protocol, which asks the prover to open 2 out of 3 commitments in the challenge phase. Therefore, the soundness error $2/3$ is inherent for a Stern-type protocol.
2. In a Fiat-Shamir with abort protocol, the prover and the verifier run a Schnorr-type protocol with some tweaks for arguing shortness of the witness. However, the standard extraction procedure for the Schnorr protocol does not work here. This is because the extracted witnesses will be scaled by some large number (more accurately, the inverse of the difference of two challenges) and may be large. To circumvent this problem, the extraction procedure avoids multiplication of inverses. Correspondingly, the definition

[1] In a concurrent and independent work [17], the problem is also addressed, using a similar technique.

of soundness is relaxed in the sense that the extracted witness does not necessarily satisfy the original relation.

1.1 Our Results

In this work, we present a new approach for constructing efficient zero-knowledge arguments of knowledge for a large class of lattice-based relations. The core component of our methodology is an efficient `ZKAoK` for linear equations with additional quadratic constraints over the witnesses.

More concretely, let m, n, and ℓ be positive integers, and q be a large enough integer that is *a power-of-prime*. The `ZKAoK` protocol proves the following relation $\mathcal{R}^*$ in Eq.(1)[2]:

$$\mathcal{R}^* = \{(\boldsymbol{A}, \boldsymbol{y}, \mathcal{M}), (\boldsymbol{x}) \in (\mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^m \times ([1,n]^3)^\ell) \times (\mathbb{Z}_q^n) : \\ \boldsymbol{A}\cdot\boldsymbol{x} = \boldsymbol{y} \ \wedge \forall (h,i,j) \in \mathcal{M}, \boldsymbol{x}[h] = \boldsymbol{x}[i]\cdot\boldsymbol{x}[j]\} \qquad (1)$$

where $\mathcal{M}$ is a set of ℓ triples that defines quadratic constraints over $\boldsymbol{x}$. Usually, ℓ will be linear in n and in any case, we have $\ell \le n^3$.

Building upon our main protocol, we present a variety of `ZKAoK`s for some concrete lattice-based relations. The constructed `ZKAoK`s have standard soundness, yet achieving an inverse polynomial soundness error. We summarize the differences between our approaches and previous results in Table 1.

Table 1: Comparison of Approaches for Lattice-Based `ZKAoK`s.

	Standard Soundness	Soundness Error
Stern-Style	✓	2/3
FSwA	✗	$1/poly$ or $negl$
This work	✓	$1/poly$

To further demonstrate the usefulness of our methodology, we develop several privacy-preserving primitives from these `ZKAoK`s. We illustrate the roadmap to these applications in Figure 1.

In addition, we also examine the concrete efficiency (particularly, communication cost) of our applications. We highlight some of the results in Table 2. For more details, see the appendix (Appendix D.3, E.3, F.3, and G).

We remark that the applications (and the performance data thereof) are to illustrate the usefulness of our framework. They are by no means exhaustive nor optimal. One may extend our results to other privacy-preserving primitives such as anonymous credential, decentralized anonymous credential, group encryption, traceable signature, linkable ring signature, CryptoNote protocol (and thus Monero), k-times anonymous authentication, blacklistable anonymous credential, Zerocoin, etc. Also, one can improve the results of this work via utilizing

[2] In this paper, operations over group elements in $\mathbb{Z}_q$ are modulo q unless otherwise specified.

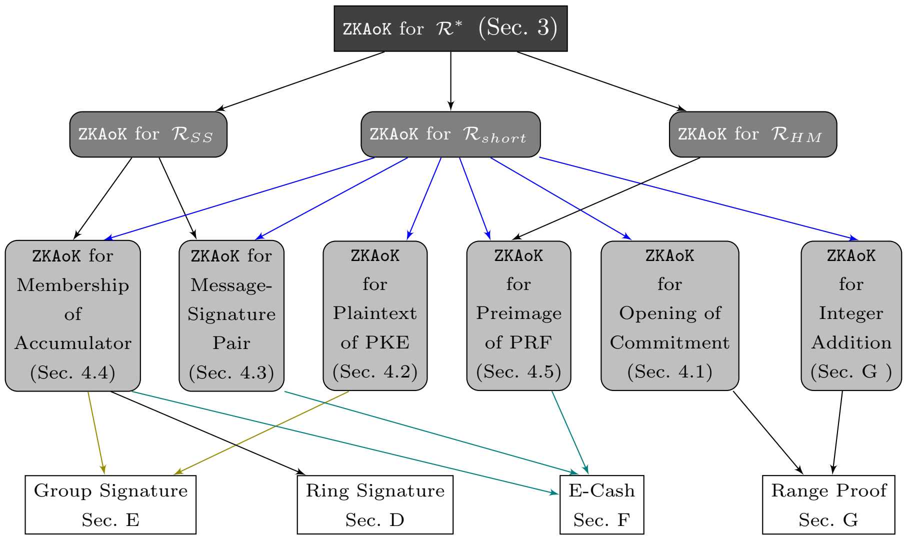
ZKAoK for $\mathcal{R}^*$ (Sec. 3)
ZKAoK for $\mathcal{R}_{SS}$
ZKAoK for $\mathcal{R}_{short}$
ZKAoK for $\mathcal{R}_{HM}$
ZKAoK for Membership of Accumulator (Sec. 4.4)
ZKAoK for Message-Signature Pair (Sec. 4.3)
ZKAoK for Plaintext of PKE (Sec. 4.2)
ZKAoK for Preimage of PRF (Sec. 4.5)
ZKAoK for Opening of Commitment (Sec. 4.1)
ZKAoK for Integer Addition (Sec. G)
Group Signature Sec. E
Ring Signature Sec. D
E-Cash Sec. F
Range Proof Sec. G

Fig. 1 The Roadmap for our `ZKAoK`s and their Applications. The starting point is our core `ZKAoK` for $\mathcal{R}^*$. It is then used to construct `ZKAoK`s for some elementary relations, namely, $\mathcal{R}_{short}$, $\mathcal{R}_{SS}$, and $\mathcal{R}_{HM}$ (we define these elementary relations and explain how to develop `ZKAoK`s for them in Sec. 1.2). Based on these elementary `ZKAoK`s, we further construct `ZKAoK`s for cryptographic schemes. Finally, we construct privacy-preserving primitives from these `ZKAoK`s.

structured lattices (such as ideal lattices or NTRU lattices) and application-specific optimizations. Those extensions and optimizations are beyond the scope of this paper.

Comparisons. Next, we give a brief comparison between the communication cost of applications in this work and that of previous results. Our examples in this section target 80 bits security unless otherwise specified.

We summarize the results in Table 2. Generally, for applications where solutions were only available through Stern-type protocols, our constructions are (much) more efficient than the state of the art. For applications where solutions were also available through Fiat-Shamir with abort protocols, our constructions are less efficient. Note that constructions utilizing Fiat-Shamir with abort are designed from scratch and these state-of-the-art constructions are optimized through the use of structured lattices (ideal lattices); while our solutions are built on standard lattices, which are believed to offer better security.

We stress again that the main advantage of our framework is that it provides a fairly good efficiency yet keeping its user-friendliness. Optimizing toward individual application, as stated earlier, is beyond the scope of this paper.

Ring Signatures. Following the framework of [54], a ring signature scheme can be obtained with our `ZKAoK`. The signature size of [54] is estimated by [33] at 47.3

Table 2: Comparison of Communication Cost for Applications from Different `ZKAoK`s.

Application	This paper	Stern-type	FSwA (ideal lattice)
Ring Signature	4.24MB	47.3MB [54]	**1.41MB** [33]
Group Signature	6.94MB	61.5MB [54]	**0.58MB** [29]
Range Proof	**1.21MB**	3.54MB [56]	N/A
Electronic Cash	**262MB**	$\approx$ 720TB [55]	N/A

MB, for a ring of 2^{10} users. In contrast, the signature size of our ring signature scheme is 4.24 MB in the same setting.

To the best of our knowledge, the most efficient ring signature scheme is from [33], using Fiat-Shamir with abort protocols. For the same number (i.e., 2^{10}) of users, its signature size is about 1.41 MB at 100 bits security level. Using a similar parameter setting, the signature size of our solution is 3.05 MB.[3]

Group Signatures. A group signature can also be obtained following a similar approach in [54] using our `ZKAoK`. The signature size of [54] is 61.5 MB for a group of 2^{10} users. In contrast, the signature size of our solution is 6.94 MB in the same setting.

The most efficient group signature scheme to date is from [29], achieving a signature size of less than 1 MB. Nonetheless, our approach can achieve additional features. For example, one can convert our group signature scheme into a fully dynamic one via the techniques in [60], without increasing its signature size.

Electronic Cash. To the best of our knowledge, the only lattice-based (compact) electronic cash system is from [55], but no concrete estimation of its performance is provided. In Appendix F.3, we provide a rough estimation for the communication cost of their spend protocol, for a wallet of 2^{10} coins. The estimation shows that the communication cost of their spend protocol is at least several terabytes while our spend protocol can achieve a communication cost of 262 MB in the same setting.

There is no E-cash system from Fiat-Shamir with abort protocols in the literature. This is due to the following technical barriers. First, in an E-cash system, we need an argument of correct evaluation for pseudorandom function (PRF). This requires an argument for the learning with rounding (LWR) relation, i.e., proving the (rounded) error terms lie *exactly* in an interval. Due to the aforementioned soundness gap, it is not known how this proof can be done from Fiat-Shamir with abort protocols. Moreover, we also need an adaptively secure signature scheme and an argument of knowledge of a valid message/signature pair for it. To date, signature schemes that admit an argument from Fiat-Shamir

[3] In [33], parameters are set in a slightly mild way, so, the signature size is smaller if we use their criterion to select parameters.

with abort protocols can only achieve selective security. Complexity leveraging trick that converts a selectively secure scheme into an adaptively secure one does not work here either, since the message space, which contains all possible PRF keys, are exponentially large.

Range Proof. Prior to our work, the most efficient lattice-based range proof is from [56]. When arguing knowledge of a 1000-bits committed value in a given range, its proof size is 3.54 MB. In contrast, the proof led by our solution is only 1.21 MB in the same setting.

1.2 Technical Overview

Warm-Up: An Argument for $\mathcal{R}_{ISIS}$ *.* Before explaining the idea of our approach, we would like to give a simple intuition on how one can argue the ISIS relation, with standard soundness and small soundness error simultaneously. Our solution can be viewed as a somewhat mix of the Stern-type protocol and the Fiat-Shamir with abort protocol. In particular, we will first use the bit-decomposition technique to deal with small integers. Then we prove that the decomposition outputs are binary via proving some quadratic constraint over them (i.e., arguing $x = x^2$ for each bit x of the output). As shown in [42] (and its lattice variant [33]), this can be proved via arguing linear relations over commitments and thus can be instantiated with known commitment with a relaxed opening and Fiat-Shamir with abort protocols. Since we do not argue shortness of witnesses explicitly in the latter argument, soundness gap is not introduced.[4] Surprisingly, this simple strategy can produce much more than merely arguing shortness of witnesses. We elaborate this next.

Building ZKAoK for $\boldsymbol{\mathcal{R}^*}$. We start with a protocol that proves

$$\mathcal{R}_0 = \{(\boldsymbol{A}, \boldsymbol{y}), (\boldsymbol{x}) \in (\mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^m) \times (\mathbb{Z}_q^n) : \boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{y}\} \tag{2}$$

which is the linear equation part of $\mathcal{R}^*$. The protocol can be viewed as an extension of the Schnorr protocol to the linear algebra setting. It proceeds as follows:

1. The prover samples a vector $\boldsymbol{r} \xleftarrow{\$} \mathbb{Z}_q^n$ and sends $\boldsymbol{t} = \boldsymbol{A} \cdot \boldsymbol{r}$ to the verifier.
2. The verifier samples a challenge $\alpha \in \mathcal{C}$ and sends it to the prover. Here $\mathcal{C} \subset \mathbb{Z}$ is the challenge space of the protocol and will be specified later.
3. The prover sends $\boldsymbol{z} = \alpha \cdot \boldsymbol{x} + \boldsymbol{r}$ to the verifier.
4. The verifier accepts the proof iff $\boldsymbol{A} \cdot \boldsymbol{z} = \alpha \cdot \boldsymbol{y} + \boldsymbol{t}$.

Given two valid transcripts with distinct challenges, i.e., $(\boldsymbol{t}, \alpha, \boldsymbol{z})$ and $(\boldsymbol{t}, \alpha', \boldsymbol{z}')$, one can extract a vector $\bar{\boldsymbol{x}} = (\alpha - \alpha')^{-1} \cdot (\boldsymbol{z} - \boldsymbol{z}')$ that satisfies Eq. (2). In the meantime, a cheating prover cannot pass the verification unless it successfully

[4] There exists a soundness gap in the proof, but it will not affect the proved argument due to the commitment with a relaxed opening.

guesses the challenge α. Thus, the protocol achieves a soundness error of $1/\|\mathcal{C}\|$. Hence, we can obtain an inverse-polynomial soundness error if $\mathcal{C}$ contains polynomial many distinct challenges.

In the remaining part of this section, we explain how to additionally prove the quadratic constraints over the witnesses.

Let (h, i, j) be an item in $\mathcal{M}$, our goal is to prove that $\boldsymbol{x}[h] = \boldsymbol{x}[i] \cdot \boldsymbol{x}[j]$. First, from the response $\boldsymbol{z} = \alpha \cdot \boldsymbol{x} + \boldsymbol{r}$, the verifier can compute

$$\begin{aligned} d &= \alpha \cdot \boldsymbol{z}[h] - \boldsymbol{z}[i] \cdot \boldsymbol{z}[j] \\ &= (\boldsymbol{x}[h] - \boldsymbol{x}[i] \cdot \boldsymbol{x}[j]) \cdot \alpha^2 + (\boldsymbol{r}[h] - \boldsymbol{r}[i] \cdot \boldsymbol{x}[j] - \boldsymbol{r}[j] \cdot \boldsymbol{x}[i]) \cdot \alpha - \boldsymbol{r}[i] \cdot \boldsymbol{r}[j] \\ &:= (\boldsymbol{x}[h] - \boldsymbol{x}[i] \cdot \boldsymbol{x}[j]) \cdot \alpha^2 + a \cdot \alpha - b \end{aligned}$$

where $a = \boldsymbol{r}[h] - \boldsymbol{r}[i] \cdot \boldsymbol{x}[j] - \boldsymbol{r}[j] \cdot \boldsymbol{x}[i]$ and $b = \boldsymbol{r}[i] \cdot \boldsymbol{r}[j]$. Note that $\boldsymbol{x}[h] = \boldsymbol{x}[i] \cdot \boldsymbol{x}[j]$ iff d is linear in α. Therefore, the main task is reduced to proving that the quadratic polynomial d is indeed linear in α, or alternatively $d - a \cdot \alpha + b$ is a *zero polynomial.*

To prove this, we can ask the prover to additionally send a and b in Step 1. Correspondingly, in Step 4, the verifier computes d and further checks if $d = a \cdot \alpha - b$. Since the prover does not know α in advance, a and b must be independent from α. Therefore, if the verification is successful, d is linear in α.

However, sending a and b in plaintext may leak information about the witness. To solve this problem, we adopt a homomorphic commitment scheme $\texttt{Commit}(m; r) \mapsto c$ that commits a message m to a commitment c using randomness r. More precisely, in Step 1, the prover generates $C_a = \texttt{Commit}(a; s_a)$ and $C_b = \texttt{Commit}(b; s_b)$ for some s_a and s_b, and send them to the verifier. In Step 3, the prover also computes $s = \alpha \cdot s_a - s_b$ and send s to the verifier. The verifier then checks if $\texttt{Commit}(d; s) = \alpha \cdot C_a - C_b$.

Remark 1.1. In this work, we will use the commitment scheme in [10], which is both additive homomorphic and supports multiplication by small constants. Therefore, we require the challenge space $\mathcal{C}$ to be a set of polynomially-many small integers. The commitment scheme also requires the randomness to be drawn from some distributions with bounded norm. Here, we instantiate it with the Gaussian distribution.

Since new variables C_a, C_b and s are introduced in the proof, we also need to make sure that they will not compromise the privacy of a and b. First, C_b is determined by α, d, s and C_a, thus, we only need to consider s and C_a in the analysis. Recall that both s_a and s_b are drawn from the Gaussian distributions. According to the rejection sampling lemma [64], we can use s_b to mask $\alpha \cdot s_a$, and enforce the output s to follow a specific distribution that is independent from s_a. Then, by the hiding property of the commitment scheme, C_a reveals nothing about a. As a result, the commitments C_a, C_b and the randomness s do not leak additional information to the verifier.

There is an additional subtlety that we need to deal with. Note that in the aforementioned protocol, we try to argue that the quadratic polynomial

$d - a\cdot\alpha - b$ is a zero polynomial. Thus, in the proof for soundness, we need three valid transcripts with distinct challenges after rewinding (note that a quadratic polynomial with three distinct roots must be a zero polynomial). So, to fix the extracted witnesses from these transcripts, the prover should also commit the witness $\boldsymbol{x}$ and proves that the witness is properly committed (using a Fiat-Shamir with abort protocol).

In summary, our `ZKAoK` protocol contains three parts.

1. A Schnorr-type protocol that proves possession of a witness for $\mathcal{R}_0$.
2. A commitment of witness $\boldsymbol{x}$ and a Fiat-Shamir with abort protocol proving that the committed value is actually $\boldsymbol{x}$.
3. A proof for the quadratic constraints over the witnesses.

Building `ZKAoK` for More Relations. Next, we show how to develop `ZKAoK`s for relations relevant to lattice-based cryptographic schemes. As we illustrated in Figure 1, such relations can be viewed as combinations of some elementary relations, namely, linear equations with short solutions ($\mathcal{R}_{short}$), subset sum of linear equations ($\mathcal{R}_{SS}$), and linear equations with hidden matrices ($\mathcal{R}_{HM}$). Thus, here we focus on how to deal with these elementary relations.[5]

Linear Equation with Short Solution. This is a primary lattice-based relation and appears in (almost) all applications. Concretely, let m, n, and k be positive integers, q be a large enough power-of-prime, and $\beta = 2^k - 1$. The relation $\mathcal{R}_{short}$ is given as

$$\mathcal{R}_{short} = \{(\boldsymbol{P}, \boldsymbol{v}), (\boldsymbol{w}) \in (\mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^m) \times ([0,\beta]^n) : \boldsymbol{P}\cdot\boldsymbol{w} = \boldsymbol{v}\}$$

The reduction from $\mathcal{R}_{short}$ to $\mathcal{R}^*$ takes the following steps:

- set a new witness $\boldsymbol{x}$ as the binary decomposition of the original witness $\boldsymbol{w}$, i.e., each element w in $\boldsymbol{w}$ is decomposed into k bits $x_1, \ldots, x_k$ such that $w = \sum_{i=1}^{k} x_i \cdot 2^{i-1}$ (note that a positive integer can be decomposed into k bits iff it is in $[0, 2^k - 1]$);
- set $\boldsymbol{A} = \boldsymbol{P}\cdot\boldsymbol{G}$ where the gadget matrix $\boldsymbol{G} := \boldsymbol{I}_n \otimes (1 \quad 2 \quad 4 \ldots 2^{k-1})$ (thus, we have $\boldsymbol{G}\cdot\boldsymbol{x} = \boldsymbol{w}$);
- set $\boldsymbol{y} = \boldsymbol{v}$;
- set $\mathcal{M} = \{(i,i,i)\}_{i\in[1,nk]}$;

In doing so, we obtain a new relation in the form of $\mathcal{R}^*$ where both the length of witness and the size of $\mathcal{M}$ are nk.

Note that since q is a power-of-prime, for any $x \in \mathbb{Z}_q$, $x^2 = x$ iff $x = 0$ or $x = 1$. Thus, the new relation is equivalent to the original relation $\mathcal{R}_{short}$.

There are two common variants to $\mathcal{R}_{short}$. First, for simplicity, we have set $\beta + 1$ to be a power-of-2. The first variant removes this unnecessary constraint

[5] Detailed constructions of ZKAoKs for elementary relations can be found in Sec. 4, e.g, the `ZKAoK` for lattice-based PKE is in fact a `ZKAoK` for a variant of $\mathcal{R}_{short}$.

and deals with arbitrary positive integer β. This is achieved by applying the refined decomposition technique proposed in [58] and the length of the decomposed witness is $n \cdot (\lfloor \log \beta \rfloor + 1)$.

The second variant is to argue knowledge of a witness $\boldsymbol{w} \in [-\beta, \beta]^n$ that satisfies a linear equation. This can be reduced to the relation $\mathcal{R}_{short}$ via adding β to each element of $\boldsymbol{w}$. Note that the linear equation will also need to be modified accordingly.

Optimized arguments for Linear Equation with Short Solution. In some cases, it is desirable to prove a relation $\mathcal{R}_{short}$ with a large n, which makes it inefficient to decompose all elements in $\boldsymbol{x}$. We propose an alternative relation, given by Eq. (3), to argue equations with short solutions more efficiently in this case, at a cost of re-introducing some soundness gap for the argument. More precisely, to argue a linear equation $\boldsymbol{P}\boldsymbol{w} = \boldsymbol{v}$ with β-bounded solution $\boldsymbol{w}$, the argument can only guarantee that the prover possesses a $n \cdot \beta$-bounded solution $\boldsymbol{w}'$ that satisfies $\boldsymbol{P}\boldsymbol{w}' = \boldsymbol{v}$.

$$\begin{aligned} \mathcal{R}'_{short} = \{ & (\boldsymbol{P}, \boldsymbol{v}, \boldsymbol{H}, \boldsymbol{c}), (\boldsymbol{w}, \boldsymbol{u}, \boldsymbol{r}) \in \\ & (\mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m \times [0,1]^{\lambda \times n} \times \mathtt{C}) \times (\mathbb{Z}_q^n \times [0, n \cdot \beta]^\lambda \times \mathtt{R}) : \\ & \boldsymbol{P} \cdot \boldsymbol{w} = \boldsymbol{v} \ \wedge \ \boldsymbol{H} \cdot \boldsymbol{w} - \boldsymbol{u} = 0 \ \wedge \ \boldsymbol{c} = \mathtt{Commit}(\boldsymbol{w}; \boldsymbol{r}) \} \end{aligned} \tag{3}$$

where

- $\mathtt{Commit}$ is a commitment scheme and $\boldsymbol{c} = \mathtt{Commit}(\boldsymbol{w}; \boldsymbol{r})$ is the commitment;
- $\mathtt{C}$ and $\mathtt{R}$ are the output space and the randomness space of $\mathtt{Commit}$;
- $\boldsymbol{H} \leftarrow H(\boldsymbol{c}) \in [0,1]^{\lambda \times n}$, where λ is the security parameter and H is modeled as a random oracle.

To see why $\mathcal{R}'_{short}$ could guarantee that all elements in $\boldsymbol{w}$ are in $[0, n \cdot \beta]$, assume there exists $i \in [n]$ such that $|\ \boldsymbol{w}[i]\ | > n \cdot \beta$. Let $\boldsymbol{h}_1$ and $\boldsymbol{h}_2$ be two n-dimension binary vectors that are identical in all positions except that $\boldsymbol{h}_1[i] \neq \boldsymbol{h}_2[i]$. Then we have $|\ \boldsymbol{h}_1^\intercal \cdot \boldsymbol{w} - \boldsymbol{h}_2^\intercal \cdot \boldsymbol{w}\ | = |\ \boldsymbol{w}[i]\ | > n \cdot \beta$. Thus, either $\boldsymbol{h}_1^\intercal \cdot \boldsymbol{w}$ or $\boldsymbol{h}_2^\intercal \cdot \boldsymbol{w}$ must be outside the interval $[0, n \cdot \beta]$. Therefore, for a vector $\boldsymbol{h}$ sampled uniformly from $[0,1]^n$, with a probability of at least $1/2$, $\boldsymbol{h}^\intercal \cdot \boldsymbol{w} > n \cdot \beta$. Therefore, the probability that all elements in $\boldsymbol{H} \cdot \boldsymbol{w}$ are in $[0, n \cdot \beta]$ is negligible.

It remains to show how to argue the relation $\mathcal{R}'_{short}$. Our strategy is to reduce the relation to an instance of relation $\mathcal{R}^*$ and then argue the instance via our main protocol. Looking ahead, in our main protocol, the prover also generates a commitment of the witness in the first step and will argue that the witness is properly committed during the proof. In addition, the commitment scheme allows one to commit part of the witness first, and then commit the remaining part later, where the partial commitment generated in the first stage is also included in the complete commitment. Consequently, the commitment and the argument for the opening of the commitment are free[6]. The remaining

[6] In fact, we only obtain a relaxed argument for the opening of the commitment. This is sufficient for our purpose.

part of relation $\mathcal{R}'_{short}$ are equations with short solutions, and thus can be straightforwardly reduced to $\mathcal{R}^*$.

In more detail, to argue $\mathcal{R}'_{short}$, the prover first generates the commitment $\boldsymbol{c} = \texttt{Commit}(\boldsymbol{w})$ and computes the matrix $\boldsymbol{H} = H(\boldsymbol{c})$ and $\boldsymbol{u} = \boldsymbol{H}\boldsymbol{w}$. Then, it commits $\boldsymbol{u}$ and appends the commitment to $\boldsymbol{c}$. Finally, it runs the remaining part of our main protocol, arguing that there exists a small vector $\boldsymbol{u}$ and a vector $\boldsymbol{w}$ that satsfies $\boldsymbol{u} = H(\boldsymbol{c})\boldsymbol{w}$ and $\boldsymbol{v} = \boldsymbol{P}\boldsymbol{w}$.

To summarize, we can prove equations with short solutions via our main protocol on $\mathcal{R}^*$, where the length of the witness is $n + \lambda \cdot (\lfloor \log(n \cdot \beta) \rfloor + 1)$ and the size of $\mathcal{M}$ is $\lambda \cdot (\lfloor \log(n \cdot \beta) \rfloor + 1)$.

Subset Sum of Linear Equations. Let m, n and l be positive integers and q be a large power-of-prime. The relation is given as

$$\begin{aligned}\mathcal{R}_{SS} = \{&(\{\boldsymbol{P}_i\}_{i\in[1,l]}, \boldsymbol{v}), (\{\boldsymbol{w}\}_{i\in[1,l]}, \{b_i\}_{i\in[1,l]}) \in \\ &((\mathbb{Z}_q^{m\times n})^l \times \mathbb{Z}_q^m) \times ((\mathbb{Z}_q^n)^l \times \{0,1\}^l) : \sum_{i=1}^{l} b_i \cdot \boldsymbol{P}_i \cdot \boldsymbol{w}_i = \boldsymbol{v}\}\end{aligned}$$

To reduce $\mathcal{R}_{SS}$ to $\mathcal{R}^*$, we first compute $\boldsymbol{v}_i = \boldsymbol{P}_i \cdot \boldsymbol{w}_i$ and $\boldsymbol{v}'_i = b_i \cdot \boldsymbol{v}_i$ for $i \in [1, l]$. Then we set the new witness vector $\boldsymbol{x} = (b_1, \ldots, b_l, \boldsymbol{v}'_1, \ldots, \boldsymbol{v}'_l, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_l, \boldsymbol{w}_1, \ldots, \boldsymbol{w}_l)$ and set

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{0} & \boldsymbol{0} & -\boldsymbol{I}_{ml} & \boldsymbol{P} \\ \boldsymbol{0} & \boldsymbol{J} & \boldsymbol{0} & \boldsymbol{0} \end{pmatrix} \text{ and } \boldsymbol{y} = \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{v} \end{pmatrix}$$

where

$$\boldsymbol{P} = \begin{pmatrix} \boldsymbol{P}_1 & & & \\ & \boldsymbol{P}_2 & & \\ & & \ddots & \\ & & & \boldsymbol{P}_l \end{pmatrix} \text{ and } \boldsymbol{J} = \begin{pmatrix} \boldsymbol{I}_m & \boldsymbol{I}_m & \ldots & \boldsymbol{I}_m \end{pmatrix}.$$

Here, the first part of the equation $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$ (specified by the first "row" of $\boldsymbol{A}$) indicates that $\boldsymbol{v}_i = \boldsymbol{P}_i \cdot \boldsymbol{w}_i$ for $i \in [1, l]$ and its second part indicates that the sum of all $\boldsymbol{v}'_i$ are $\boldsymbol{v}$.

Finally, we set

$$\mathcal{M} = \{(i, i, i)\}_{i\in[1,l]} \cup \{(l + m\cdot(i-1) + j, l + ml + m\cdot(i-1) + j, i)\}_{i\in[1,l], j\in[1,m]}$$

where $\{(i, i, i)\}_{i\in[1,l]}$ indicates that b_i is binary and the rest indicates that $\boldsymbol{v}'_i = b_i \cdot \boldsymbol{v}_i$. This gives us an $\mathcal{R}^*$ statement where the length of witness becomes $(nl + 2ml + l)$ and the size of $\mathcal{M}$ is $ml + l$.

Linear Equation with Hidden Matrix. Let m and n be positive integers and q be a large power-of-prime, the relation is defined as follows:

$$\mathcal{R}_{HM} = \{(\boldsymbol{v}), (\boldsymbol{P}, \boldsymbol{w}) \in (\mathbb{Z}_q^m) \times (\mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^n) : \boldsymbol{P} \cdot \boldsymbol{w} = \boldsymbol{v}\}$$

To reduce $\mathcal{R}_{HM}$ to $\mathcal{R}^*$, we first obtain a new witness vector $\boldsymbol{x} = (\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{2m})$ as follows:

- $\boldsymbol{x}_0 = \boldsymbol{w}$;
- for $i \in [1, m]$, $\boldsymbol{x}_i$ is the i-th row of $\boldsymbol{P}$;
- for $i \in [1, m]$, $\boldsymbol{x}_{m+i}$ is the Hadamard product between the i-th row of $\boldsymbol{P}$ and $\boldsymbol{w}$ (i.e., $\boldsymbol{x}_{m+i}[j] = \boldsymbol{x}_i[j] \cdot \boldsymbol{w}[j]$).

Then we set $\boldsymbol{A} = \begin{pmatrix} \boldsymbol{0}^{m\times n} & \boldsymbol{0}^{m\times mn} & \boldsymbol{M} \end{pmatrix}$ and $\boldsymbol{y} = \boldsymbol{v}$ where $\boldsymbol{M} = \boldsymbol{I}_m \otimes \begin{pmatrix} 1 & 1 & \dots & 1 \end{pmatrix} \in \mathbb{Z}_q^{m\times mn}$.

Finally, we set $\mathcal{M} = \{((m+i)\cdot n + j, i \cdot n + j, j)\}_{i\in[1,m], j\in[1,n]}$, which indicates that $\boldsymbol{x}_{m+i}[j] = \boldsymbol{x}_i[j] \cdot \boldsymbol{w}[j]$. In this way, we obtain a new relation in the form of $\mathcal{R}^*$, where the length of witness is $(2m+1)\cdot n$ and the size of $\mathcal{M}$ is mn.

2 Preliminaries

Notations. In this paper, we will use bold lower-case letters (e.g., $\boldsymbol{v}$) to denote vectors, and use bold upper-case letters (e.g., $\boldsymbol{A}$) to denote matrices. All elements in vectors and matrices are integers unless otherwise specified. For a vector $\boldsymbol{v}$ of length n, we use $\boldsymbol{v}[i]$ to denote the ith element of $\boldsymbol{v}$ for $i \in [1, n]$ and for an m-by-n matrix $\boldsymbol{A}$, we use $\boldsymbol{A}[i, j]$ to denote the element on the i-th row and the j-th columon of $\boldsymbol{A}$ for $i \in [1, m]$ and $j \in [1, n]$. For a vector $\boldsymbol{v}$, we use $\texttt{bin}(\boldsymbol{v})$ to denote the binary decomposition of $\boldsymbol{v}$, i.e., $\boldsymbol{v}[i] = \sum_{j=1}^{k} 2^{j-1} \cdot \bar{\boldsymbol{v}}[(i-1)\cdot k + j]$, where $\bar{\boldsymbol{v}} = \texttt{bin}(\boldsymbol{v})$ and $k = \lceil \log(\|\boldsymbol{v}\|_\infty) \rceil$. We use $\boldsymbol{I}_n$ to denote an n-by-n identity matrix. We use $\otimes$ to denote the Kronecker product of two matrices.

For a string a, we use $\|a\|$ to denote the length of a. For a finite set $\mathcal{S}$, we use $\|\mathcal{S}\|$ to denote the size of $\mathcal{S}$ and use $s \overset{\$}{\leftarrow} \mathcal{S}$ to denote sampling an element s uniformly from set $\mathcal{S}$. For a distribution $\mathcal{D}$, we use $d \leftarrow \mathcal{D}$ to denote sampling d according to $\mathcal{D}$.

For integers $a \le b$, we write $[a, b]$ to denote all integers from a to b. We write $negl(\cdot)$ to denote a negligible function and write $poly(\cdot)$ to denote a polynomial.

2.1 Discrete Gaussian Distribution

We recall the discrete Gaussian distribution and some results from [64].

Definition 2.1 (Discrete Gaussian Distribution). *The continuous Gaussian distribution over $\mathbb{R}^m$ centered at $\boldsymbol{v} \in \mathbb{R}^m$ with standard deviation σ is defined by the function $\rho^m_{\boldsymbol{v},\sigma}(\boldsymbol{x}) = (\frac{1}{\sqrt{2\pi\sigma^2}})^m e^{\frac{-\|\boldsymbol{x}-\boldsymbol{v}\|^2}{2\sigma^2}}$.*

The discrete Gaussian distribution over $\mathbb{Z}^m$ centered at $\boldsymbol{v} \in \mathbb{Z}^m$ with standard deviation σ is defined as $D^m_{\boldsymbol{v},\sigma}(\boldsymbol{x}) = \rho^m_{\boldsymbol{v},\sigma}(\boldsymbol{x}) / \rho^m_\sigma(\mathbb{Z}^m)$, where $\rho^m_\sigma(\mathbb{Z}^m) = \sum_{\boldsymbol{x}\in\mathbb{Z}^m} \rho^m_\sigma(\boldsymbol{x})$.

We write $D^m_\sigma(\boldsymbol{x}) = D^m_{\boldsymbol{0},\sigma}(\boldsymbol{x})$ for short.

Lemma 2.1 ([64, Full Version, Lemma 4.4]).

1. *For any $k > 0$, $\Pr[\|z\| > k\sigma : \boldsymbol{z} \leftarrow D^1_\sigma] \le 2e^{\frac{-k^2}{2}}$.*
2. *For any $\boldsymbol{z} \in \mathbb{Z}^m$, and $\sigma \ge 3/\sqrt{2\pi}$, $D^m_\sigma(\boldsymbol{z}) \le 2^{-m}$.*
3. *For any $k > 1$, $\Pr[\|\boldsymbol{z}\| > k\sigma\sqrt{m} : \boldsymbol{z} \leftarrow D^m_\sigma] < k^m e^{\frac{m}{2}(1-k^2)}$.*

2.2 Rejection Sampling

In this work, we will also use the celebrated "rejection sampling lemma" from [63, 64] to argue the zero-knowledge property of our protocol.

Lemma 2.2 ([64, Full Version, Theorem 4.6]). *Let $\mathcal{V}$ be a subset of $\mathbb{Z}^m$ in which all elements have norms less than T. Let h be a probability distribution over $\mathcal{V}$. Let σ be a real number that $\sigma = \omega(T\sqrt{\log m})$. Then there exists a constant M such that the distribution of the following algorithm $\mathcal{A}$ and that of the following algorithm $\mathcal{F}$ are within statistical distance $\frac{2^{-\omega(\log m)}}{M}$.*

$\mathcal{A}$:

1. *$\boldsymbol{v} \leftarrow h$*
2. *$\boldsymbol{z} \leftarrow D^m_{\boldsymbol{v},\sigma}$*
3. *Output $(\boldsymbol{v}, \boldsymbol{z})$ with probability $\min(1, \frac{D^m_\sigma(\boldsymbol{z})}{M D^m_{\boldsymbol{v},\sigma}(\boldsymbol{z})})$*

$\mathcal{F}$:

1. *$\boldsymbol{v} \leftarrow h$*
2. *$\boldsymbol{z} \leftarrow D^m_\sigma$*
3. *Output $(\boldsymbol{v}, \boldsymbol{z})$ with probability $\frac{1}{M}$*

Moreover, the probability that $\mathcal{A}$ outputs something is at least $\frac{1-2^{-\omega(\log m)}}{M}$.

As a concrete example (suggested in [46]), if $\sigma = \alpha T$ for some positive α, then $M = e^{13.3/\alpha + 1/(2\alpha^2)}$, the output of algorithm $\mathcal{A}$ is within statistical distance $\frac{2^{-128}}{M}$ of the output of $\mathcal{F}$, and the probability that $\mathcal{A}$ outputs something is at least $\frac{1-2^{-128}}{M}$.

2.3 Hardness Assumptions

The security of our main protocol relies on the short integer solution (SIS) assumption and the learning with errors (LWE) assumption. For both assumptions, we will use the normal form (as defined in [72]).

Definition 2.2 (SIS$_{n,m,q,\beta}$, Normal Form). *Given a random matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n\times(m-n)}$, find a nonzero integer vector $\boldsymbol{z} \in \mathbb{Z}^m$ such that $\|\boldsymbol{z}\| \leq \beta$ and $[\boldsymbol{I}_n \mid \boldsymbol{A}] \cdot \boldsymbol{z} = 0$.*

As hardness of the SIS assumption usually depends only on n, q, β (assuming m is large enough), in this work, we write SIS$_{n,m,q,\beta}$ as SIS$_{n,q,\beta}$ for short.

Lemma 2.3 ([2, 37, 66, 68, 72]). *For any $m = poly(n)$, any $\beta > 0$, and any sufficiently large $q \geq \beta \cdot \tilde{O}(\sqrt{n})$, solving (normal form) SIS$_{n,m,q,\beta}$ with non-negligible probability is at least as hard as solving the decisional approximate shortest vector problem GapSVP$_\gamma$ and the approximate shortest independent vectors problems SIVP$_\gamma$ (among others) on arbitrary n-dimensional lattices (i.e., in the worst case) with overwhelming probability, for some $\gamma = \beta \cdot \tilde{O}(\sqrt{n})$.*

Definition 2.3 (Decision-LWE$_{n,m,q,\chi}$, Normal Form). *Given a random matrix $\boldsymbol{A} \in \mathbb{Z}_q^{(m-n)\times n}$, and a vector $\boldsymbol{b} \in \mathbb{Z}_q^{m-n}$, where $\boldsymbol{b}$ is generated according to either of the following two cases:*

1. $\boldsymbol{b} = \boldsymbol{A} \cdot \boldsymbol{s} + \boldsymbol{e}$, *where* $\boldsymbol{s} \leftarrow \chi^n$ *and* $\boldsymbol{e} \leftarrow \chi^{m-n}$
2. $\boldsymbol{b} \overset{\$}{\leftarrow} \mathbb{Z}_q^{m-n}$
distinguish which is the case with non-negligible advantage.

If χ is a discrete Gaussian distribution with standard deviation σ, we write the problem as $LWE_{n,m,q,\alpha}$ where $\alpha = \sigma \cdot \sqrt{2\pi}/q$. Also, as the hardness of the LWE assumption usually depends only on n, q, α (assuming m is large enough), in this work, we write $LWE_{n,m,q,\alpha}$ as $LWE_{n,q,\alpha}$ for short.

Lemma 2.4 ([7, 72, 74]). *For any $m = poly(n)$, any modulus $q \leq 2^{poly(n)}$, and any (discrete) Gaussian error distribution χ with standard deviation σ (i.e., $\chi = D_\sigma$), where $\sigma = \alpha q/\sqrt{2\pi} \geq \sqrt{2n/\pi}$ and $0 < \alpha < 1$, solving the (normal form) decision-$LWE_{n,m,q,\chi}$ problem is at least as hard as (quantumly) solving $GapSVP_\gamma$ and $SIVP_\gamma$ on arbitrary n-dimensional lattices, for some $\gamma = \tilde{O}(n/\alpha)$.*

To build applications on top of our proof system, we require some additional variants of the LWE assumption. We also need to evaluate the concrete hardness of those problems, in order to derive parameters for our system. They are presented in Appendix B.

2.4 Zero-Knowledge Arguments of Knowledge

In a zero-knowledge argument of knowledge system [39], a prover proves to a verifier that he possesses the witness for a statement without revealing any additional information.

More formally, let $\mathtt{R} = \{(x, w)\} \in \{0,1\}^* \times \{0,1\}^*$ be a statements-witnesses set for an NP relation. The ZKAoK for $\mathtt{R}$ is an interactive protocol $\langle \mathcal{P}, \mathcal{V} \rangle$ run between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$ that satisfies:

- **Completeness.** For any $(x, w) \in \mathtt{R}$, $\Pr[\langle \mathcal{P}(x, w), \mathcal{V}(x) \rangle \neq 1] \leq \delta_c$.
- **Proof of Knowledge.** There exists an extractor $\mathcal{E}$ that for any x, for any probabilistic polynomial time (PPT) cheating prover $\hat{\mathcal{P}}$, if $\Pr[\langle \hat{\mathcal{P}}, \mathcal{V}(x) \rangle = 1] > \delta_s + \epsilon$ for some non-negligible ϵ, then $\mathcal{E}$ can extract in polynomial time a witness w such that $(x, w) \in \mathtt{R}$ via accessing $\hat{P}$ in a black-box manner.
- **(Honest-Verifier) Zero-Knowledge.** There exists a simulator $\mathcal{S}$ that for any $(x, w) \in \mathtt{R}$, the two distributions are computationally indistinguishable:
  1. The view of an honest verifier $\mathcal{V}$ in an interaction $\langle \mathcal{P}(x, w), \mathcal{V}(x) \rangle$.
  2. The output of $\mathcal{S}(x)$.

where δ_c is the completeness error and δ_s is the soundness error.

In this work, we also consider non-interactive ZKAoKs (NIZKAoK). They can be obtained by applying the Fiat-Shamir heuristic [34] to public coin ZKAoKs. One advantage led by the Fiat-Shamir transform is that the transformed NIZKAoKs additionally admit a message as input, thus it is also called signature proof of knowledge (SPK), and is usually written as $SPK\{(x, w) : (x, w) \in \mathtt{R}\}[m]$, where m is the additional message.

2.5 Commitment with A Relaxed Opening

In our main construction, we will employ the commitment scheme presented in [10][7], which admits a relaxed opening.

Let λ be the security parameter. Let l_1 and l_2 be positive integers that are polynomials in the security parameter λ. Let σ be a small positive integer that satisfies $\sigma \geq \sqrt{2l_2/\pi}$. Also, let n be the length of the comitted vector. The public parameter of the commitment scheme is a matrix $\boldsymbol{B} \in \mathbb{Z}_q^{(l_1+n)\times(l_1+n+l_2)}$ defined as follows:

$$\boldsymbol{B} = \left(\begin{array}{c|cc} \boldsymbol{I}_{l_1} & \multicolumn{2}{c}{\boldsymbol{B}_1} \\ 0^{n \times l_1} & \boldsymbol{I}_n & \boldsymbol{B}_2 \end{array} \right)$$

where $\boldsymbol{B}_1$ and $\boldsymbol{B}_2$ are random matrices sampled from $\mathbb{Z}_q^{l_1\times(l_2+n)}$ and $\mathbb{Z}_q^{n\times l_2}$ respectively.

To commit to a message $\boldsymbol{m} \in \{0,1\}^n$, the commit algorithm first samples $\boldsymbol{s} \in D_\sigma^{l_1+n+l_2}$. Then it outputs a commitment $\boldsymbol{c} = \boldsymbol{B} \cdot \boldsymbol{s} + (\boldsymbol{0}^\intercal \| \boldsymbol{m}^\intercal)^\intercal$ and the opening $\boldsymbol{s}$.

The open algorithm outputs 1 on input $\boldsymbol{B}, \boldsymbol{m}, \boldsymbol{c}, \boldsymbol{s}$ iff $\boldsymbol{c} = \boldsymbol{B} \cdot \boldsymbol{s} + (\boldsymbol{0}^\intercal \| \boldsymbol{m}^\intercal)^\intercal$ and $\boldsymbol{s}$ is small. Besides, it admits a relaxed opening, where the input of the algorithm includes $\boldsymbol{B}, \boldsymbol{m}, \boldsymbol{c}, \boldsymbol{s}$ and a small integer f, and the algorithm outputs 1 iff $f \cdot \boldsymbol{c} = \boldsymbol{B} \cdot \boldsymbol{s} + f \cdot (\boldsymbol{0}^\intercal \| \boldsymbol{m}^\intercal)^\intercal$ and $\boldsymbol{s}, f$ are small.

3 Main Construction

In this section, we present our main construction, namely, an efficient zero-knowledge argument of knowledge for linear equations with quadratic constraints over the witness.

More concretely, let m, n, ℓ be positive integers, q be a large enough integer that is *a power-of-prime*, i.e., $q = q_0^e$ for some prime q_0 and some positive integer e. Also, let $\boldsymbol{A}$ be a matrix in $\mathbb{Z}_q^{m\times n}$, $\boldsymbol{x}$ and $\boldsymbol{y}$ be vectors in $\mathbb{Z}_q^n$ and $\mathbb{Z}_q^m$ respectively, and $\mathcal{M}$ be a set of ℓ 3-tuples, each of which consists of 3 integers in $[1, n]$. We will construct a ZKAoK for the following relation:

$$\mathcal{R}^* = \{(\boldsymbol{A}, \boldsymbol{y}, \mathcal{M}), (\boldsymbol{x}) : \boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{y} \ \wedge \forall (h,i,j) \in \mathcal{M}, \boldsymbol{x}[h] = \boldsymbol{x}[i] \cdot \boldsymbol{x}[j]\} \qquad (4)$$

Specifically, in Sec. 3.1, we give a basic version of the ZKAoK protocol for $\mathcal{R}^*$ as defined in Eq. (4). This protocol achieves an inverse polynomial soundness error and a constant completeness error. Then, in Sec. 3.2, we transform the basic protocol into a NIZKAoK with negligible soundness error and completeness error.

[7] In fact, we will use its variant in the standard lattice setting. For completeness, we will restate its security in the security proof of our main construction.

3.1 The Basic Protocol

Let `aCommit` be an auxiliary bit commitment scheme with randomness space $\{0,1\}^{\kappa}$ and a suitable message space. As no additional requirement is desired for `aCommit`, we can safely assume it to be a random oracle G, i.e., given an input x and a random string ρ as randomness, the commitment is $G(x\|\rho)$. Nonetheless, `aCommit` can be instantiated by any secure commitment scheme.

Let λ be the security parameter. Let l_1 and l_2 be positive integers that are polynomials in the security parameter λ. Let $\boldsymbol{B}_{1,1}$, $\boldsymbol{B}_{1,2}$, $\boldsymbol{B}_{2,1}$ and $\boldsymbol{B}_{2,2}$ be random matrices sampled from $\mathbb{Z}_q^{l_1\times(l_2+n)}$, $\mathbb{Z}_q^{n\times l_2}$, $\mathbb{Z}_q^{l_1\times(l_2+\ell)}$ and $\mathbb{Z}_q^{\ell\times l_2}$ respectively. Also let

$$\boldsymbol{B}_1=\left(\begin{array}{c|c}\boldsymbol{I}_{l_1} & \boldsymbol{B}_{1,1}\\ 0^{n\times l_1} & \boldsymbol{I}_n \ \ \boldsymbol{B}_{1,2}\end{array}\right),\quad \boldsymbol{B}_2=\left(\begin{array}{c|c}\boldsymbol{I}_{l_1} & \boldsymbol{B}_{2,1}\\ 0^{\ell\times l_1} & \boldsymbol{I}_\ell \ \ \boldsymbol{B}_{2,2}\end{array}\right)$$

Here $\boldsymbol{B}_1$ and $\boldsymbol{B}_2$ are public parameters of the underlying homomorphic commitment scheme, and we assume that they are honestly generated (via some public coin) and are shared by all parties in the protocol.

Let σ_1 be small positive integer that satisfies $\sigma_1 \geq \sqrt{2l_2/\pi}$. Let p be small positive integer that is polynomial in λ. Let $l = 2l_1 + 2l_2 + n + \ell$. Let $\sigma_2 = 2p\cdot\sqrt{l}\cdot\log l\cdot\sigma_1$. Let $M = e^{13.3/\log l + 1/(2\log^2 l)}$. For any l-dimension vectors $\boldsymbol{v}$ and $\boldsymbol{z}$, let $\mathfrak{p}(\boldsymbol{v},\boldsymbol{z}) = \min(1, \frac{D^l_{\sigma_2}(\boldsymbol{z})}{MD^l_{\boldsymbol{v},\sigma_2}(\boldsymbol{z})})$.

The basic protocol P_1 for $\mathcal{R}^*$ is described in Figure 2.

Theorem 3.1. *Assume the worst-case hardness of GapSVP$_\gamma$ (or SIVP$_\gamma$) for some polynomial γ, if $q \geq 16p\cdot\max(\sqrt{l_1+l_2+n}, \sqrt{l_1+l_2+\ell})\cdot(\sigma_2+p\cdot\sigma_1)\cdot\tilde{O}(\sqrt{l_1})$, q/σ_1 is a polynomial, $q_0 > 2p$, and* `aCommit` *is a secure bit commitment scheme, then the protocol P_1, which is described in Figure 2, is a secure zero-knowledge argument of knowledge with completeness error $1-1/M$ and soundness error $2/(2p+1)$.*

We give the detailed proof for Theorem 3.1 in Appendix C.

3.2 NIZKAoK for $\mathcal{R}^*$

In this section, we show how to transform our basic protocol in Sec. 3.1 into a non-interactive zero-knowledge arguments of knowledge with negligible soundness error and completeness error. Generally, this can be done via some standard techniques such as repetition and Fiat-Shamir transform. Nonetheless, we will employ a few tricks (developed in previous works) to reduce the efficiency loss in the transformations. In particular, to minimize the number of repetitions, we will employ the tweaks in [33] when repeating the basic protocol. In a nutshell, it applies one rejection sampling on all (repeated) instances simultaneously, which avoids completeness error increasing caused by repetition.

$\mathcal{P}[\boldsymbol{A}, \boldsymbol{y}, \mathcal{M}, \boldsymbol{B}_1, \boldsymbol{B}_2; \boldsymbol{x}]$ $\qquad\qquad$ $\mathcal{V}[\boldsymbol{A}, \boldsymbol{y}, \mathcal{M}, \boldsymbol{B}_1, \boldsymbol{B}_2]$

$\boldsymbol{r} \xleftarrow{\$} \mathbb{Z}_q^n$; $\boldsymbol{t} = \boldsymbol{A} \cdot \boldsymbol{r}$

$\boldsymbol{s}_1 \leftarrow D_{\sigma_1}^{l_2+n+l_1}$; $\boldsymbol{s}_2 \leftarrow D_{\sigma_2}^{l_2+n+l_1}$

$\boldsymbol{s}_3 \leftarrow D_{\sigma_1}^{l_2+\ell+l_1}$; $\boldsymbol{s}_4 \leftarrow D_{\sigma_2}^{l_2+\ell+l_1}$

$\boldsymbol{c}_1 = \boldsymbol{B}_1 \cdot \boldsymbol{s}_1 + (\boldsymbol{0}^\intercal \| \boldsymbol{x}^\intercal)^\intercal$

$\boldsymbol{c}_2 = \boldsymbol{B}_1 \cdot \boldsymbol{s}_2 + (\boldsymbol{0}^\intercal \| \boldsymbol{r}^\intercal)^\intercal$

Let $\boldsymbol{a}$ and $\boldsymbol{b}$ be two ℓ-dimension vectors.

For $k \in [1, \ell]$, let (h, i, j) be the k-th element in $\mathcal{M}$:

$\qquad \boldsymbol{a}[k] = \boldsymbol{r}[h] - \boldsymbol{r}[i] \cdot \boldsymbol{x}[j] - \boldsymbol{r}[j] \cdot \boldsymbol{x}[i]$

$\qquad \boldsymbol{b}[k] = \boldsymbol{r}[i] \cdot \boldsymbol{r}[j]$

$\boldsymbol{c}_3 = \boldsymbol{B}_2 \cdot \boldsymbol{s}_3 + (\boldsymbol{0}^\intercal \| \boldsymbol{a}^\intercal)^\intercal$

$\boldsymbol{c}_4 = \boldsymbol{B}_2 \cdot \boldsymbol{s}_4 + (\boldsymbol{0}^\intercal \| \boldsymbol{b}^\intercal)^\intercal$

$\rho \xleftarrow{\$} \{0,1\}^\kappa$

$C_{aux} = \texttt{aCommit}(\boldsymbol{t} \| \boldsymbol{c}_1 \| \boldsymbol{c}_2 \| \boldsymbol{c}_3 \| \boldsymbol{c}_4; \rho)$

$$\xrightarrow{\qquad\qquad C_{aux} \qquad\qquad}$$

$\alpha \xleftarrow{\$} [-p, p]$

$$\xleftarrow{\qquad\qquad \alpha \qquad\qquad}$$

$\boldsymbol{z}_0 = \alpha \cdot \boldsymbol{x} + \boldsymbol{r}$

$\boldsymbol{z}_1 = \alpha \cdot \boldsymbol{s}_1 + \boldsymbol{s}_2$

$\boldsymbol{z}_2 = \alpha \cdot \boldsymbol{s}_3 - \boldsymbol{s}_4$

Abort with probability $1 - \mathfrak{p}((\alpha \cdot \boldsymbol{s}_1^\intercal \| \alpha \cdot \boldsymbol{s}_3^\intercal)^\intercal, (\boldsymbol{z}_1^\intercal \| \boldsymbol{z}_2^\intercal)^\intercal)$

$$\xrightarrow{\qquad \boldsymbol{t}, \boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{c}_3, \boldsymbol{c}_4, \rho, \boldsymbol{z}_0, \boldsymbol{z}_1, \boldsymbol{z}_2 \qquad}$$

Let $\boldsymbol{d}$ be an ℓ-dimension vector.

For $k \in [1, \ell]$, let (h, i, j) be the k-th element in $\mathcal{M}$:

$\qquad \boldsymbol{d}[k] = \alpha \cdot \boldsymbol{z}_0[h] - \boldsymbol{z}_0[i] \cdot \boldsymbol{z}_0[j]$

Accept if:

1. $C_{aux} = \texttt{aCommit}(\boldsymbol{t} \| \boldsymbol{c}_1 \| \boldsymbol{c}_2 \| \boldsymbol{c}_3 \| \boldsymbol{c}_4; \rho)$
2. $\|\boldsymbol{z}_1\| \leq 2\sqrt{l_1 + l_2 + n} \cdot (\sigma_2 + p \cdot \sigma_1)$
3. $\|\boldsymbol{z}_2\| \leq 2\sqrt{l_1 + l_2 + \ell} \cdot (\sigma_2 + p \cdot \sigma_1)$
4. $\boldsymbol{A} \cdot \boldsymbol{z}_0 = \alpha \cdot \boldsymbol{y} + \boldsymbol{t}$
5. $\boldsymbol{B}_1 \cdot \boldsymbol{z}_1 + (\boldsymbol{0}^\intercal \| \boldsymbol{z}_0^\intercal)^\intercal = \alpha \cdot \boldsymbol{c}_1 + \boldsymbol{c}_2$
6. $\boldsymbol{B}_2 \cdot \boldsymbol{z}_2 + (\boldsymbol{0}^\intercal \| \boldsymbol{d}^\intercal)^\intercal = \alpha \cdot \boldsymbol{c}_3 - \boldsymbol{c}_4$

Fig. 2 The Basic Protocol P_1: A Zero-Knowledge Arguments of Knowledge for $\mathcal{R}^*$ with Inverse Polynomial Soundness Error and Constant Completeness Error.

The Construction. Let $\texttt{aCommit}$, λ, l_1, l_2, $\boldsymbol{B}_{1,1}$, $\boldsymbol{B}_{1,2}$, $\boldsymbol{B}_{2,1}$ and $\boldsymbol{B}_{2,2}$, σ_1, p, l and M be identical to those of P_1. We highlight the differences.

In the new scheme, a proof is generated by repeating the basic protocol $N = \lambda / \log p$ times. Then we set $\sigma_2 = 2p \cdot \sqrt{N \cdot l} \cdot \log(N \cdot l) \cdot \sigma_1$, and for any $N \cdot l$-dimension vectors $\boldsymbol{v}$ and $\boldsymbol{z}$, we set $\mathfrak{p}(\boldsymbol{v}, \boldsymbol{z}) = \min(1, \frac{D_{\sigma_2}^{N \cdot l}(\boldsymbol{z})}{M D_{\boldsymbol{v}, \sigma_2}^{N \cdot l}(\boldsymbol{z})})$. We will additionally use a hash function H with output space $[-p,p]^N$, which is modelled as a random oracle. Also, let AUX be some application-dependent auxiliary information (e.g., the signed message in a group signature) that is specified as an input to H.

The prove algorithm and the verify algorithm of the NIZKAoK P_2 for $\mathcal{R}^*$ is described in Figure 3 and 4 respectively.

```
                    $\mathtt{Prove}(\boldsymbol{A},\boldsymbol{x},\boldsymbol{y},\mathcal{M},\boldsymbol{B}_1,\boldsymbol{B}_2,AUX)$:
For $\jmath \in [1,\lambda]$:
  1. $\boldsymbol{s}_1 \leftarrow D_{\sigma_1}^{l_2+n+l_1}$, $\boldsymbol{c}_1 = \boldsymbol{B}_1 \cdot \boldsymbol{s}_1 + (\mathbf{0}^\intercal \| \boldsymbol{x}^\intercal)^\intercal$
  2. For $\imath \in [1,N]$:
     (a) $\boldsymbol{r}_\imath \xleftarrow{\$} \mathbb{Z}_q^n$, $\boldsymbol{t}_\imath = \boldsymbol{A} \cdot \boldsymbol{r}_\imath$
     (b) $\boldsymbol{s}_{2,\imath} \leftarrow D_{\sigma_2}^{l_2+n+l_1}$, $\boldsymbol{s}_{3,\imath} \leftarrow D_{\sigma_1}^{l_2+\ell+l_1}$, $\boldsymbol{s}_{4,\imath} \leftarrow D_{\sigma_2}^{l_2+\ell+l_1}$
     (c) $\boldsymbol{c}_{2,\imath} = \boldsymbol{B}_1 \cdot \boldsymbol{s}_{2,\imath} + (\mathbf{0}^\intercal \| \boldsymbol{r}_\imath^\intercal)^\intercal$
     (d) Let $\boldsymbol{a}_\imath$ and $\boldsymbol{b}_\imath$ be two $\ell$-dimension vectors and for $k \in [1,\ell]$, let $(h,i,j)$
         be the $k$-th element in $\mathcal{M}$:
          i. $\boldsymbol{a}_\imath[k] = \boldsymbol{r}_\imath[h] - \boldsymbol{r}_\imath[i] \cdot \boldsymbol{x}[j] - \boldsymbol{r}_\imath[j] \cdot \boldsymbol{x}[i]$
         ii. $\boldsymbol{b}_\imath[k] = \boldsymbol{r}_\imath[i] \cdot \boldsymbol{r}_\imath[j]$
     (e) $\boldsymbol{c}_{3,\imath} = \boldsymbol{B}_2 \cdot \boldsymbol{s}_{3,\imath} + (\mathbf{0}^\intercal \| \boldsymbol{a}_\imath^\intercal)^\intercal$, $\boldsymbol{c}_{4,\imath} = \boldsymbol{B}_2 \cdot \boldsymbol{s}_{4,\imath} + (\mathbf{0}^\intercal \| \boldsymbol{b}_\imath^\intercal)^\intercal$
     (f) $\rho_\imath \xleftarrow{\$} \{0,1\}^\kappa$
     (g) $C_{aux,\imath} = \mathtt{aCommit}(\boldsymbol{t}_\imath \| \boldsymbol{c}_1 \| \boldsymbol{c}_{2,\imath} \| \boldsymbol{c}_{3,\imath} \| \boldsymbol{c}_{4,\imath}; \rho_\imath)$
  3. $\{\alpha_\imath\}_{\imath\in[1,N]} = H(\boldsymbol{A},\boldsymbol{y},\mathcal{M},\{C_{aux,\imath}\}_{\imath\in[1,N]},AUX)$
  4. For $\imath \in [1,N]$:
     (a) $\boldsymbol{z}_{0,\imath} = \alpha_\imath \cdot \boldsymbol{x} + \boldsymbol{r}_\imath$, $\boldsymbol{z}_{1,\imath} = \alpha_\imath \cdot \boldsymbol{s}_1 + \boldsymbol{s}_{2,\imath}$, $\boldsymbol{z}_{2,\imath} = \alpha_\imath \cdot \boldsymbol{s}_{3,\imath} - \boldsymbol{s}_{4,\imath}$
  5. Smaple a real number $\tau \xleftarrow{\$} [0,1]$ (Here, we use $[0,1]$ to denote all real numbers
     between 0 and 1)
  6. If $\tau < \mathfrak{p}((\alpha_1 \cdot \boldsymbol{s}_1^\intercal \| \ldots \| \alpha_N \cdot \boldsymbol{s}_1^\intercal \| \alpha_1 \cdot \boldsymbol{s}_{3,1}^\intercal \| \ldots \| \alpha_N \cdot \boldsymbol{s}_{3,N}^\intercal)^\intercal,$
     $(\boldsymbol{z}_{1,1}^\intercal \| \ldots \| \boldsymbol{z}_{1,N}^\intercal \| \boldsymbol{z}_{2,1}^\intercal \| \ldots \| \boldsymbol{z}_{2,N}^\intercal)^\intercal)$:
     (a) Abort the algorithm with output $\pi = (\boldsymbol{c}_1, \{\alpha_\imath, \rho_\imath, \boldsymbol{c}_{3,\imath}, \boldsymbol{z}_{0,\imath}, \boldsymbol{z}_{1,\imath},$
         $\boldsymbol{z}_{2,\imath}\}_{\imath\in[1,N]})$
Output $\perp$ if the algorithm does not abort in the loop above.
```

Fig. 3 The Prove Algorithm of P_2.

Theorem 3.2. *Assume the worst-case hardness of GapSVP$_\gamma$ (or SIVP$_\gamma$) for some polynomial γ, if $q \geq 16p \cdot \max(\sqrt{l_1+l_2+n}, \sqrt{l_1+l_2+\ell}) \cdot (\sigma_2 + p \cdot \sigma_1) \cdot \tilde{O}(\sqrt{l_1})$, q/σ_1 is a polynomial, $q_0 > 2p$, aCommit is a secure bit commitment scheme, and H is modelled as a random oracle, then the scheme P_2 is a secure non-interactive zero-knowledge argument of knowledge with negligible completeness error and soundness error.*

Proof of Theorem 3.2 follows proof of Theorem 3.1 and well-known results, we omit the details here.

Efficiency. In P_2, a proof π contains a commitment and a set of N elements, where each element consists of a challenge, a κ-bit string, a commitment and

```
$\texttt{Verify}(\boldsymbol{A}, \boldsymbol{y}, \mathcal{M}, \boldsymbol{B}_1, \boldsymbol{B}_2, AUX, \pi = (\boldsymbol{c}_1, \{\alpha_\imath, \rho_\imath, \boldsymbol{c}_{3,\imath}, \boldsymbol{z}_{0,\imath}, \boldsymbol{z}_{1,\imath}, \boldsymbol{z}_{2,\imath}\}_{\imath \in [1,\mathtt{N}]}))$ :
For $\imath \in [1, N]$ :
  1. Let $\boldsymbol{d}_\imath$ be an $\ell$-dimension vector and for $k \in [1, \ell]$, let $(h, i, j)$ be the $k$-th element in $\mathcal{M}$ :
     (a) $\boldsymbol{d}_\imath[k] = \alpha_\imath \cdot \boldsymbol{z}_{0,\imath}[h] - \boldsymbol{z}_{0,\imath}[i] \cdot \boldsymbol{z}_{0,\imath}[j]$
  2. $\boldsymbol{t}_\imath = \boldsymbol{A} \cdot \boldsymbol{z}_{0,\imath} - \alpha_\imath \cdot \boldsymbol{y}$
  3. $\boldsymbol{c}_{2,\imath} = \boldsymbol{B}_1 \cdot \boldsymbol{z}_{1,\imath} + (\boldsymbol{0}^\intercal \| \boldsymbol{z}_{0,\imath}^\intercal)^\intercal - \alpha_\imath \cdot \boldsymbol{c}_1$
  4. $\boldsymbol{c}_{4,\imath} = \alpha_\imath \cdot \boldsymbol{c}_{3,\imath} - \boldsymbol{B}_2 \cdot \boldsymbol{z}_{2,\imath} - (\boldsymbol{0}^\intercal \| \boldsymbol{d}_\imath^\intercal)^\intercal$
  5. $C_{aux,\imath} = \texttt{aCommit}(\boldsymbol{t}_\imath \| \boldsymbol{c}_1 \| \boldsymbol{c}_{2,\imath} \| \boldsymbol{c}_{3,\imath} \| \boldsymbol{c}_{4,\imath}; \rho_\imath)$
  6. If $\|\boldsymbol{z}_{1,\imath}\| > 2\sqrt{l_1 + l_2 + n} \cdot (\sigma_2 + p \cdot \sigma_1) \vee \|\boldsymbol{z}_{2,\imath}\| > 2\sqrt{l_1 + l_2 + \ell} \cdot (\sigma_2 + p \cdot \sigma_1)$ :
     (a) Abort the algorithm with output "Reject"
Output "Accept" if $\{\alpha_\imath\}_{\imath \in [1,N]} = H(\boldsymbol{A}, \boldsymbol{y}, \mathcal{M}, \{C_{aux,\imath}\}_{\imath \in [1,N]}, AUX)$ :
```

Fig. 4 The `Verify` Algorithm of P_2.

three vectors. Thus, we have

$$\|\pi\| = (\log(2p+1) + \kappa + (3l_1 + 2l_2 + 2n + 2\ell) \cdot \log q) \cdot N + (l_1 + n) \cdot \log q$$

4 ZKAoKs for Various Cryptographic Schemes

In this section, we build several tools that are useful for constructing privacy-preserving primitives. This includes an argument of knowledge of committed value, an argument of knowledge of plaintext, an argument of knowledge of signature, an argument for cryptogrphic accumulator and an argument for pseudorandom function.

4.1 ZKAoK of Committed Value

We start with an argument of knowledge of the committed value for the commitment scheme in [47], which is recalled in Appendix A.1.

Let l_1, l_2, L be positive integers and q be a power-of-prime. We propose a `ZKAoK` for the following relation:

$$\mathcal{R}_{com} = \{(\boldsymbol{B}_1, \boldsymbol{B}_2, \boldsymbol{c}), (\boldsymbol{r}, \boldsymbol{w}) \in (\mathbb{Z}_q^{l_1 \times l_2} \times \mathbb{Z}_q^{l_1 \times L} \times \mathbb{Z}_q^{l_1}) \times (\{0,1\}^{l_2} \times \{0,1\}^L) : \boldsymbol{B}_1 \cdot \boldsymbol{r} + \boldsymbol{B}_2 \cdot \boldsymbol{w} = \boldsymbol{c}\}$$

$\mathcal{R}_{com}$ contains linear equations with binary witness. We construct the argument via reducing $\mathcal{R}_{com}$ to an instance of $\mathcal{R}^*$ through the following steps:

1. Set the new witness $\boldsymbol{x} = (\boldsymbol{r}^\intercal \| \boldsymbol{w}^\intercal)^\intercal$;
2. Set $\boldsymbol{A} = (\boldsymbol{B}_1 \| \boldsymbol{B}_2)$ and $\boldsymbol{y} = \boldsymbol{c}$;
3. Set $\mathcal{M} = (i, i, i)_{i \in [1, l_2 + L]}$.

Note that since q is a power of prime, for any $x \in \mathbb{Z}_q$, $x^2 = x$ iff $x = 0$ or $x = 1$. Thus, the new relation $\mathcal{R}^*$ over $(\boldsymbol{A}, \boldsymbol{y}, \mathcal{M}), (\boldsymbol{x})$ is equivalent to the original relation $\mathcal{R}_{com}$. Also, both $\|\boldsymbol{x}\|$ and $\|\mathcal{M}\|$ are $l_2 + L$ for $\mathcal{R}^*$.

4.2 ZKAoK of Plaintext

Next, we give an argument of knowledge of the plaintext for the encryption scheme proposed in [57], which is recalled in Appendix A.2.

More precisely, let l_1, l_2, L and β be positive integers and q be a power-of-prime, we propose a `ZKAoK` for the following relation:

$$\begin{aligned}\mathcal{R}_{enc} = \{&(\boldsymbol{B}_1, \boldsymbol{B}_2, \boldsymbol{c}_1, \boldsymbol{c}_2), (\boldsymbol{r}, \boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{w}) \in \\ &(\mathbb{Z}_q^{l_1 \times l_2} \times \mathbb{Z}_q^{L \times l_2} \times \mathbb{Z}_q^{l_1} \times \mathbb{Z}_q^{L}) \times (\mathbb{Z}_q^{l_2} \times \mathbb{Z}_q^{l_1} \times \mathbb{Z}_q^{L} \times \{0,1\}^L) : \\ &\|\boldsymbol{r}\|_\infty \le \beta \wedge \|\boldsymbol{e}_1\|_\infty \le \beta \wedge \|\boldsymbol{e}_2\|_\infty \le \beta \wedge \\ &\boldsymbol{B}_1 \cdot \boldsymbol{r} + \boldsymbol{e}_1 = \boldsymbol{c}_1 \wedge \boldsymbol{B}_2 \cdot \boldsymbol{r} + \boldsymbol{e}_2 + \lfloor \frac{q}{2} \rceil \cdot \boldsymbol{w} = \boldsymbol{c}_2\}\end{aligned}$$

We construct the argument via reducing the relation $\mathcal{R}_{enc}$, which contains linear equations with short solutions, to an instance of the relation $\mathcal{R}^*$.

First, we define vectors $\boldsymbol{\beta}_1 = (\beta \quad \beta \ldots \beta)^\intercal \in \mathbb{Z}_q^{l_2}$, $\boldsymbol{\beta}_2 = (\beta \quad \beta \ldots \beta)^\intercal \in \mathbb{Z}_q^{l_1}$, $\boldsymbol{\beta}_3 = (\beta \quad \beta \ldots \beta)^\intercal \in \mathbb{Z}_q^{L}$ and define $\boldsymbol{r}' = \boldsymbol{r} + \boldsymbol{\beta}_1$, $\boldsymbol{e}_1' = \boldsymbol{e}_1 + \boldsymbol{\beta}_2$ and $\boldsymbol{e}_2' = \boldsymbol{e}_2 + \boldsymbol{\beta}_3$.

Then, we decompose vectors $\boldsymbol{r}'$, $\boldsymbol{e}_1'$ and $\boldsymbol{e}_2'$ into binary vectors $\bar{\boldsymbol{r}}$, $\bar{\boldsymbol{e}}_1$ and $\bar{\boldsymbol{e}}_2$ using the decomposition technique proposed in [58]. More precisely, let $k = \lfloor \log 2\beta \rfloor + 1$ and let $\boldsymbol{g} = (\lfloor (2\beta+1)/2 \rfloor \| \lfloor (2\beta+2)/4 \rfloor \| \ldots \| \lfloor (2\beta + 2^{i-1})/2^i \rfloor \| \ldots \| \lfloor (2\beta + 2^{k-1})/2^k \rfloor)$ be a row vector. It is claimed in [58] that 1) an integer $a \in [0, 2\beta]$ iff there exists a binary vecotr $\boldsymbol{a} \in \{0,1\}^k$ that $\boldsymbol{g} \cdot \boldsymbol{a} = a$; 2) one can decompose the integer $a \in [0, 2\beta]$ into the k-dimension binary vector $\boldsymbol{a}$ efficiently.

Next, we define the gadget matrix $\boldsymbol{G}_1 = \boldsymbol{I}_{l_2} \otimes \boldsymbol{g}$, $\boldsymbol{G}_2 = \boldsymbol{I}_{l_1} \otimes \boldsymbol{g}$, $\boldsymbol{G}_3 = \boldsymbol{I}_L \otimes \boldsymbol{g}$ and they satisfy that $\boldsymbol{G}_1 \cdot \bar{\boldsymbol{r}} = \boldsymbol{r}'$, $\boldsymbol{G}_2 \cdot \bar{\boldsymbol{e}}_1 = \boldsymbol{e}_1'$ and $\boldsymbol{G}_3 \cdot \bar{\boldsymbol{e}}_2 = \boldsymbol{e}_2'$.

Finally, we set

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{B}_1 \cdot \boldsymbol{G}_1 & \boldsymbol{G}_2 & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{B}_2 \cdot \boldsymbol{G}_1 & \boldsymbol{0} & \boldsymbol{G}_3 & \lfloor \frac{q}{2} \rceil \cdot \boldsymbol{I}_L \end{pmatrix}$$

$$\boldsymbol{x} = (\bar{\boldsymbol{r}}^\intercal \quad \bar{\boldsymbol{e}}_1^\intercal \quad \bar{\boldsymbol{e}}_2^\intercal \quad \boldsymbol{w}^\intercal)^\intercal, \quad \boldsymbol{y} = \begin{pmatrix} \boldsymbol{c}_1 + \boldsymbol{B}_1 \cdot \boldsymbol{\beta}_1 + \boldsymbol{\beta}_2 \\ \boldsymbol{c}_2 + \boldsymbol{B}_2 \cdot \boldsymbol{\beta}_1 + \boldsymbol{\beta}_3 \end{pmatrix}$$

and set $\mathcal{M} = (i, i, i)_{i \in [1, (l_1 + l_2 + L) \cdot k + L]}$. Here, both $\|\boldsymbol{x}\|$ and $\|\mathcal{M}\|$ are $(l_1 + l_2 + L) \cdot k + L$.

One common variant of the encryption scheme in [57] is to use binary secrets and errors rather than sampling them from β bounded distributions. To generate arguments of knowledge of plaintexts for this variant, we can use an almost identical construction as above, except that we do not need to decompose the vectors $\boldsymbol{r}$, $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$. Thus, when reducing the relation to $\mathcal{R}^*$ in this case, both $\|\boldsymbol{x}\|$ and $\|\mathcal{M}\|$ will be $l_1 + l_2 + 2L$.

4.3 ZKAoK of Message-Signature Pair

Next, we give an argument of knowledge of a valid message/signature pair for the signature scheme proposed in [55], which is recalled in Appendix A.3.

Let l_1, l_2, l_3, L, and β be positive integers and q be a power-of-prime. Also let $k_q = \lceil \log q \rceil$. We propose a ZKAoK that proves knowledge of

$$\begin{cases} \{\tau_i\}_{i\in[1,l_3]} \in \{0,1\}^{l_3}; \boldsymbol{v}_1 \in \mathbb{Z}_q^{l_2}; \boldsymbol{v}_2 \in \mathbb{Z}_q^{l_2}; \\ \boldsymbol{w} \in \{0,1\}^{k_q l_1}; \boldsymbol{s} \in \mathbb{Z}_q^{2l_2}; \boldsymbol{m} \in \{0,1\}^L \end{cases}$$

that satisfies

$$\begin{cases} \boldsymbol{B}\cdot\boldsymbol{v}_1 + (\boldsymbol{B}_0 + \sum_{i=1}^{l_3} \tau_i \cdot \boldsymbol{B}_i)\cdot\boldsymbol{v}_2 = \boldsymbol{u} + \boldsymbol{D}\cdot\boldsymbol{w} \\ \boldsymbol{H}\cdot\boldsymbol{w} = \boldsymbol{D}_0\cdot\boldsymbol{s} + \boldsymbol{D}_1\cdot\boldsymbol{m} \\ \|\boldsymbol{v}_1\|_\infty \le \beta; \|\boldsymbol{v}_2\|_\infty \le \beta; \|\boldsymbol{s}\|_\infty \le \beta \end{cases}$$

for public

$$\begin{cases} \boldsymbol{B} \in \mathbb{Z}_q^{l_1\times l_2}; \{\boldsymbol{B}_i\}_{i\in[0,l_3]} \in (\mathbb{Z}_q^{l_1\times l_2})^{l_3+1}; \boldsymbol{u} \in \mathbb{Z}_q^{l_1} \\ \boldsymbol{D} \in \mathbb{Z}_q^{l_1\times k_q l_1}; \boldsymbol{D}_0 \in \mathbb{Z}_q^{l_1\times 2l_2}; \boldsymbol{D}_1 \in \mathbb{Z}_q^{l_1\times L} \end{cases}$$

where $\boldsymbol{H} = \boldsymbol{I}_{l_1} \otimes (1 \quad 2 \quad 4 \ldots 2^{k_q-1})$.

Again, we construct the argument via reducing the relation, which contains a subset sum of linear equations and linear equations with short solutions, to an instance of the relation $\mathcal{R}^*$.

First, we define vectors $\boldsymbol{\beta}_1 = (\beta \quad \beta \ldots \beta)^\intercal \in \mathbb{Z}_q^{l_2}$, $\boldsymbol{\beta}_2 = (\beta \quad \beta \ldots \beta)^\intercal \in \mathbb{Z}_q^{2l_2}$, and define $\boldsymbol{v}_1' = \boldsymbol{v}_1 + \boldsymbol{\beta}_1$, $\boldsymbol{v}_2' = \boldsymbol{v}_2 + \boldsymbol{\beta}_1$ and $\boldsymbol{s}' = \boldsymbol{s} + \boldsymbol{\beta}_2$.

Then, we decompose vectors $\boldsymbol{v}_1'$, $\boldsymbol{v}_2'$ and $\boldsymbol{s}'$ into binary vectors $\bar{\boldsymbol{v}}_1$, $\bar{\boldsymbol{v}}_2$, and $\bar{\boldsymbol{s}}$ using the decomposition technique proposed in [58]. Let $k = \lfloor \log 2\beta \rfloor + 1$, then the vectors $\bar{\boldsymbol{v}}_1$, $\bar{\boldsymbol{v}}_2$ and $\bar{\boldsymbol{s}}$ are of length kl_2, kl_2 and $2kl_2$ respectively.

Also, let $\boldsymbol{g} = (\lfloor (2\beta+1)/2 \rfloor \| \ldots \| \lfloor (2\beta + 2^{i-1})/2^i \rfloor \| \ldots \| \lfloor (2\beta + 2^{k-1})/2^k \rfloor)$ be a row vector. Then, we define the gadget matrix $\boldsymbol{G}_1 = \boldsymbol{I}_{l_2} \otimes \boldsymbol{g}$, $\boldsymbol{G}_2 = \boldsymbol{I}_{2l_2} \otimes \boldsymbol{g}$, and they satisfy that $\boldsymbol{G}_1 \cdot \bar{\boldsymbol{v}}_1 = \boldsymbol{v}_1'$, $\boldsymbol{G}_1 \cdot \bar{\boldsymbol{v}}_2 = \boldsymbol{v}_2'$ and $\boldsymbol{G}_2 \cdot \bar{\boldsymbol{s}} = \boldsymbol{s}'$.

Next, for $i \in [1, l_3]$, let $\boldsymbol{u}_i = \boldsymbol{B}_i \cdot \boldsymbol{v}_2$ and let $\boldsymbol{u}_i' = \tau_i \cdot \boldsymbol{u}_i$. Also, we define $\hat{\boldsymbol{u}} = (\boldsymbol{u}_1^\intercal \| \boldsymbol{u}_2^\intercal \| \ldots \| \boldsymbol{u}_{l_3}^\intercal)^\intercal$ and $\hat{\boldsymbol{u}}' = (\boldsymbol{u}_1'^\intercal \| \boldsymbol{u}_2'^\intercal \| \ldots \| \boldsymbol{u}_{l_3}'^\intercal)^\intercal$. Moreover, define $\boldsymbol{\tau} = (\tau_1 \quad \tau_2 \ldots \tau_{l_3})^\intercal$.

Finally, we set

$$\boldsymbol{A} = \begin{pmatrix} \mathbf{0} & \mathbf{0} & -\boldsymbol{I}_{l_1 l_3} & \bar{\boldsymbol{B}} \cdot \boldsymbol{G}_1 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{J} & \mathbf{0} & \boldsymbol{B}_0 \cdot \boldsymbol{G}_1 & \boldsymbol{B} \cdot \boldsymbol{G}_1 & -\boldsymbol{D} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\boldsymbol{H} & \boldsymbol{D}_0 \cdot \boldsymbol{G}_2 & \boldsymbol{D}_1 \end{pmatrix}$$

$$\boldsymbol{x} = \begin{pmatrix} \boldsymbol{\tau}^\mathsf{T} & \hat{\boldsymbol{u}}'^\mathsf{T} & \hat{\boldsymbol{u}}^\mathsf{T} & \bar{\boldsymbol{v}}_2^\mathsf{T} & \bar{\boldsymbol{v}}_1^\mathsf{T} & \boldsymbol{w}^\mathsf{T} & \bar{\boldsymbol{s}}^\mathsf{T} & \boldsymbol{m}^\mathsf{T} \end{pmatrix}^\mathsf{T}, \quad \boldsymbol{y} = \begin{pmatrix} \bar{\boldsymbol{B}} \cdot \boldsymbol{\beta}_1 \\ \boldsymbol{u} + \boldsymbol{B}_0 \cdot \boldsymbol{\beta}_1 + \boldsymbol{B} \cdot \boldsymbol{\beta}_1 \\ \boldsymbol{D}_0 \cdot \boldsymbol{\beta}_2 \end{pmatrix}$$

where

$$\bar{\boldsymbol{B}} = \begin{pmatrix} \boldsymbol{B}_1 \\ \vdots \\ \boldsymbol{B}_{l_3} \end{pmatrix}, \quad \boldsymbol{J} = (\boldsymbol{I}_{l_1} \quad \boldsymbol{I}_{l_1} \quad \dots \quad \boldsymbol{I}_{l_1})$$

Besides, let $N = l_3 + 2l_1l_3 + 2kl_2 + k_ql_1 + 2kl_2 + L$, we define

$$\begin{cases} \mathcal{M}_1 = \{(i,i,i)\}_{i\in[1,l_3]} \\ \mathcal{M}_2 = \{(i,i,i)\}_{i\in[l_3+2l_1l_3+1,N]} \\ \mathcal{M}_3 = \{(l_3 + l_1 \cdot (i-1) + j, i, l_3 + l_1l_3 + l_1 \cdot (i-1) + j)\}_{i\in[1,l_3],j\in[1,l_1]} \end{cases}$$

where $\mathcal{M}_1$ indicates that each τ_i is binary, $\mathcal{M}_2$ indicates that $\bar{\boldsymbol{v}}_2, \bar{\boldsymbol{v}}_1, \boldsymbol{w}, \bar{\boldsymbol{s}}, \boldsymbol{m}$ are binary vectors, and $\mathcal{M}_3$ indicates that $\boldsymbol{u}'_i = \tau_i \cdot \boldsymbol{u}_i$ for $i \in [1, l_3]$. Then we set $\mathcal{M} = \mathcal{M}_1 \cup \mathcal{M}_2 \cup \mathcal{M}_3$. In the new relation, the length of the witness is N and the size of $\mathcal{M}$ is $N - l_1l_3$.

We can also use the fast mode (mentioned in Sec. 1.2) to argue that $\boldsymbol{v}_1$, $\boldsymbol{v}_2$ and $\boldsymbol{s}$ are short. This will lead to an instance of $\mathcal{R}^*$, where the length of the witness is $l_3 + 2l_1l_3 + 4l_2 + k_ql_1 + L + \lambda \cdot (\lfloor \log(2 \cdot 4l_2 \cdot \beta) \rfloor + 1)$, and the size of $\mathcal{M}$ is $l_3 + l_1l_3 + k_ql_1 + L + \lambda \cdot (\lfloor \log(2 \cdot 4l_2 \cdot \beta) \rfloor + 1)$.

4.4 ZKAoK of Accumulated Value

In this section, we give an argument of knowledge of an accumulated value for the accumulator scheme presented in [54], which is recalled in Appendix A.4.

More precisely, let l_1, L be positive integers and q be a power-of-prime. Also, let $k_q = \lceil \log q \rceil$ and $l_2 = l_1k_q$. We propose a zero knowledge argument of knowledge that proves knowledge of

$$\{\{\tau_i\}_{i\in[1,L]} \in \{0,1\}^L; \{\boldsymbol{v}_i\}_{i\in[1,L]} \in ([0,1]^{l_2})^L; \{\boldsymbol{w}_i\}_{i\in[1,L]} \in ([0,1]^{l_2})^L;\}$$

that satisfies

$$\begin{cases} \boldsymbol{B}_{1+\tau_1} \cdot \boldsymbol{v}_1 + \boldsymbol{B}_{2-\tau_1} \cdot \boldsymbol{w}_1 = \boldsymbol{H} \cdot \boldsymbol{u} \\ \forall i \in [2, L], \boldsymbol{B}_{1+\tau_i} \cdot \boldsymbol{v}_i + \boldsymbol{B}_{2-\tau_i} \cdot \boldsymbol{w}_i = \boldsymbol{H} \cdot \boldsymbol{v}_{i-1} \end{cases}$$

for public

$$\{\boldsymbol{B}_1 \in \mathbb{Z}_q^{l_1 \times l_2}; \boldsymbol{B}_2 \in \mathbb{Z}_q^{l_1 \times l_2}; \boldsymbol{u} \in [0,1]^{l_1 k_q}\}$$

where $\boldsymbol{H} = \boldsymbol{I}_{l_1} \otimes (1 \quad 2 \quad 4 \ldots 2^{k_q-1})$.

We construct the argument via reducing the relation to an instance of the relation $\mathcal{R}^*$. Note that the relation contains L parts, each of which is a disjunction of two equations, namely, $\boldsymbol{B}_1 \cdot \boldsymbol{v}_i + \boldsymbol{B}_2 \cdot \boldsymbol{w}_i = \boldsymbol{H} \cdot \boldsymbol{v}_{i-1}$ and $\boldsymbol{B}_1 \cdot \boldsymbol{w}_i + \boldsymbol{B}_2 \cdot \boldsymbol{v}_i = \boldsymbol{H} \cdot \boldsymbol{v}_{i-1}$ (here, we define $\boldsymbol{v}_0 = \boldsymbol{u}$). As shown in [54], each part can be transformed into a subset sum of these two equations via setting the coefficients as $(1-\tau_i, \tau_i)$. Next, we describe the reduction in more details.

First, for $i \in [2, L]$, we define $\boldsymbol{z}_{i,0} = \boldsymbol{B}_1 \cdot \boldsymbol{v}_i + \boldsymbol{B}_2 \cdot \boldsymbol{w}_i - \boldsymbol{H} \cdot \boldsymbol{v}_{i-1}$, $\boldsymbol{z}_{i,1} = \boldsymbol{B}_1 \cdot \boldsymbol{w}_i + \boldsymbol{B}_2 \cdot \boldsymbol{v}_i - \boldsymbol{H} \cdot \boldsymbol{v}_{i-1}$, $\boldsymbol{z}'_{i,0} = (1-\tau_i) \cdot \boldsymbol{z}_{i,0}$ and $\boldsymbol{z}'_{i,1} = \tau_i \cdot \boldsymbol{z}_{i,1}$. Moreover, we set $\boldsymbol{z}_{1,0} = \boldsymbol{B}_1 \cdot \boldsymbol{v}_1 + \boldsymbol{B}_2 \cdot \boldsymbol{w}_1$, $\boldsymbol{z}_{1,1} = \boldsymbol{B}_1 \cdot \boldsymbol{w}_1 + \boldsymbol{B}_2 \cdot \boldsymbol{v}_1$, $\boldsymbol{z}'_{1,0} = (1-\tau_1) \cdot \boldsymbol{z}_{1,0}$ and $\boldsymbol{z}'_{1,1} = \tau_1 \cdot \boldsymbol{z}_{1,1}$.

Then, we set $\boldsymbol{\tau}_0 = (1-\tau_1 \quad 1-\tau_2 \ldots 1-\tau_L)^\intercal$ and $\boldsymbol{\tau}_1 = (\tau_1 \quad \tau_2 \ldots \tau_L)^\intercal$. Also, we define $\hat{\boldsymbol{z}}'_0 = (\boldsymbol{z}'^\intercal_{1,0} \| \boldsymbol{z}'^\intercal_{2,0} \| \ldots \| \boldsymbol{z}'^\intercal_{L,0})^\intercal$, $\hat{\boldsymbol{z}}'_1 = (\boldsymbol{z}'^\intercal_{1,1} \| \boldsymbol{z}'^\intercal_{2,1} \| \ldots \| \boldsymbol{z}'^\intercal_{L,1})^\intercal$, $\hat{\boldsymbol{z}}_0 = (\boldsymbol{z}^\intercal_{1,0} \| \boldsymbol{z}^\intercal_{2,0} \| \ldots \| \boldsymbol{z}^\intercal_{L,0})^\intercal$, $\hat{\boldsymbol{z}}_1 = (\boldsymbol{z}^\intercal_{1,1} \| \boldsymbol{z}^\intercal_{2,1} \| \ldots \| \boldsymbol{z}^\intercal_{L,1})^\intercal$, $\hat{\boldsymbol{v}} = (\boldsymbol{v}_1^\intercal \| \boldsymbol{v}_2^\intercal \| \ldots \| \boldsymbol{v}_L^\intercal)^\intercal$, $\hat{\boldsymbol{w}} = (\boldsymbol{w}_1^\intercal \| \boldsymbol{w}_2^\intercal \| \ldots \| \boldsymbol{w}_L^\intercal)^\intercal$. Besides, we define $\hat{\boldsymbol{u}} = ((\boldsymbol{H} \cdot \boldsymbol{u})^\intercal \| \boldsymbol{0}^{1 \times (L-1) \cdot l_1})^\intercal$.

Finally, we set

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{I}_L & \boldsymbol{I}_L & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I}_{l_1 L} & \boldsymbol{I}_{l_1 L} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I}_{l_1 L} & \boldsymbol{0} & \boldsymbol{M}_1 & \boldsymbol{N}_2 \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I}_{l_1 L} & \boldsymbol{M}_2 & \boldsymbol{N}_1 \end{pmatrix}$$

$$\boldsymbol{x} = \begin{pmatrix} \boldsymbol{\tau}_0^\intercal & \boldsymbol{\tau}_1^\intercal & \hat{\boldsymbol{z}}_0'^\intercal & \hat{\boldsymbol{z}}_1'^\intercal & \hat{\boldsymbol{z}}_0^\intercal & \hat{\boldsymbol{z}}_1^\intercal & \bar{\boldsymbol{v}}^\intercal & \bar{\boldsymbol{w}}^\intercal \end{pmatrix}^\intercal, \quad \boldsymbol{y} = \begin{pmatrix} \boldsymbol{1}^L & \hat{\boldsymbol{u}}^\intercal & \boldsymbol{0} & \boldsymbol{0} \end{pmatrix}^\intercal$$

where

$$\boldsymbol{M}_1 = \begin{pmatrix} -\boldsymbol{B}_1 & & & \\ \boldsymbol{H} & -\boldsymbol{B}_1 & & \\ & \ddots & \ddots & \\ & & \boldsymbol{H} & -\boldsymbol{B}_1 \end{pmatrix}, \quad \boldsymbol{M}_2 = \begin{pmatrix} -\boldsymbol{B}_2 & & & \\ \boldsymbol{H} & -\boldsymbol{B}_2 & & \\ & \ddots & \ddots & \\ & & \boldsymbol{H} & -\boldsymbol{B}_2 \end{pmatrix}$$

$$\boldsymbol{N}_1 = -\boldsymbol{I}_L \otimes \boldsymbol{B}_1, \quad \boldsymbol{N}_2 = -\boldsymbol{I}_L \otimes \boldsymbol{B}_2$$

Besides, we define

$$\begin{cases} \mathcal{M}_1 = \{(i,i,i)\}_{i \in [1,L]} \\ \mathcal{M}_2 = \{(i,i,i)\}_{i \in [2L+4l_1L+1, 2L+4l_1L+2l_2L]} \\ \mathcal{M}_3 = \{(2L + l_1 \cdot (i-1) + j, 2L + 2l_1L + l_1 \cdot (i-1) + j, i)\}_{i \in [1,L], j \in [1,l_1]} \\ \mathcal{M}_4 = \{(2L + l_1L + l_1 \cdot (i-1) + j, 2L + 3l_1L + l_1 \cdot (i-1) + j, L+i)\}_{i \in [1,L], j \in [1,l_1]} \end{cases}$$

where $\mathcal{M}_1$ indicates that $\boldsymbol{\tau}_0$ is a binary vector, $\mathcal{M}_2$ indicates that $\bar{\boldsymbol{v}}$ and $\bar{\boldsymbol{w}}$ are binary vectors, $\mathcal{M}_3$ and $\mathcal{M}_4$ indicate $\boldsymbol{z}'_{i,0} = (1-\tau_i)\cdot \boldsymbol{z}_{i,0}$ and $\boldsymbol{z}'_{i,1} = \tau_i \cdot \boldsymbol{z}_{i,1}$ for $i \in [1, L]$ respectively. Then we set $\mathcal{M} = \mathcal{M}_1 \cup \mathcal{M}_2 \cup \mathcal{M}_3 \cup \mathcal{M}_4$. Note that as in the linear equation $\boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{y}$, it is proved that $\boldsymbol{\tau}_0[i] + \boldsymbol{\tau}_1[i] = 1$ for $i \in [1, L]$, the fact that $\boldsymbol{\tau}_0[i]$ is binary implies that $\boldsymbol{\tau}_1[i]$ is also binary. In the new relation, the length of the witness is $2L + 4l_1L + 2l_2L$ and the size of $\mathcal{M}$ is $L + 2l_1L + 2l_2L$.

4.5 ZKAoK of PRF Preimage

In this section, we give an argument for the weak pseudorandom function constructed implicitly in [9], which is recalled in Appendix A.5. In paticular, the argument claims knowledge of a key/input pair that evaluates to a public output.

More precisely, let l_1, l_2 be positive integers, q_0 be a prime and $p = q_0^{e_1}$, $q = q_0^{e_2}$, where $1 \le e_1 < e_2$, we propose a `ZKAoK` for the following relation:

$$\mathcal{R}_{PRF} = \{(\boldsymbol{c}), (\boldsymbol{B}, \boldsymbol{k}) \in (\mathbb{Z}_p^{l_1}) \times (\mathbb{Z}_q^{l_1 \times l_2} \times \mathbb{Z}_q^{l_2}) : \boldsymbol{c} = \lfloor \boldsymbol{B} \cdot \boldsymbol{k} \rceil_p \mod p\}$$

We construct the argument via reducing the relation $\mathcal{R}_{PRF}$ to an instance of the relation $\mathcal{R}^*$. First, we rewrite the equation $\boldsymbol{c} = \lfloor \boldsymbol{B} \cdot \boldsymbol{k} \rceil_p \mod p$ as follows:

$$\begin{cases} \boldsymbol{B} \cdot \boldsymbol{k} = \boldsymbol{u} \mod q \\ \lfloor \dfrac{p}{q} \cdot \boldsymbol{u} \rfloor = \boldsymbol{c} \mod p \end{cases}$$

The first equation is a linear equation with hidden matrix. The second equation, as shown in [55, 79], holds iff each element of the vector $\boldsymbol{u} - \frac{q}{p}\boldsymbol{c}$ is in $[0, \frac{q}{p})$, and thus can be transformed into a linear equation with short solution. Next, we describe the reduction in more details. We remark that in the remaining part of this section, all arithmetic operations are under the modulus q, so we omit the moduli in the remaining part of this section.

First, for $i \in [1, l_1]$, we define $\boldsymbol{b}_i$ as the i-th row of $\boldsymbol{B}$ and define $\boldsymbol{v}_i$ as the Hadamard product between $\boldsymbol{b}_i$ and $\boldsymbol{k}$, i.e., $\boldsymbol{v}_i[j] = \boldsymbol{b}_i[j] \cdot \boldsymbol{k}[j]$ for $j \in [1, l_2]$.

Let $\boldsymbol{e} = \boldsymbol{u} - \frac{q}{p}\boldsymbol{c}$, then we decompose the vector $\boldsymbol{e}$ into a binary vector $\bar{\boldsymbol{e}}$ using the decomposition technique proposed in [58]. Let $\gamma = \frac{q}{p} - 1$ and $k = \lfloor \log \gamma \rfloor + 1$, then the length of $\bar{\boldsymbol{e}}$ is $k \cdot l_1$.

Also, let $\boldsymbol{g} = (\lfloor (\gamma+1)/2 \rfloor \| \ldots \| \lfloor (\gamma + 2^{i-1})/2^i \rfloor \| \ldots \| \lfloor (\gamma + 2^{k-1})/2^k \rfloor)$ be a row vector. Then, we define the gadget matrix $\boldsymbol{G} = \boldsymbol{I}_{l_1} \otimes \boldsymbol{g}$, and it satisfies that $\boldsymbol{G} \cdot \bar{\boldsymbol{e}} = \boldsymbol{e}$.

Next, we define $\boldsymbol{b} = (\boldsymbol{b}_1^\intercal \| \ldots \| \boldsymbol{b}_{l_1}^\intercal)^\intercal \in \mathbb{Z}_q^{l_1 \cdot l_2}$ and define $\boldsymbol{v} = (\boldsymbol{v}_1^\intercal \| \ldots \| \boldsymbol{v}_{l_1}^\intercal)^\intercal \in \mathbb{Z}_q^{l_1 \cdot l_2}$.

Finally, we set

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{M} & -\boldsymbol{I}_{l_1} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I}_l & -\boldsymbol{G} \end{pmatrix}$$

$$\boldsymbol{x} = (\boldsymbol{k}^\mathsf{T} \quad \boldsymbol{b}^\mathsf{T} \quad \boldsymbol{v}^\mathsf{T} \quad \boldsymbol{u}^\mathsf{T} \quad \bar{\boldsymbol{e}}^\mathsf{T})^\mathsf{T}, \quad \boldsymbol{y} = \begin{pmatrix} \boldsymbol{0} & \frac{q}{p} \cdot \boldsymbol{c}^\mathsf{T} \end{pmatrix}^\mathsf{T}$$

where $\boldsymbol{M} = \boldsymbol{I}_{l_1} \otimes (1\ 1 \ldots 1) \in \mathbb{Z}_q^{l_1 \times l_1 \cdot l_2}$.

Besides, we define

$$\begin{cases} \mathcal{M}_1 = \{(i, i, i)\}_{i \in [l_2 + 2l_1 l_2 + l_1 + 1, l_2 + 2l_1 l_2 + l_1 + kl_1]} \\ \mathcal{M}_2 = \{(l_2 + l_1 l_2 + (i-1) \cdot l_2 + j, l_2 + (i-1) \cdot l_2 + j, j)\}_{i \in [1, l_1], j \in [1, l_2]} \end{cases}$$

where $\mathcal{M}_1$ indicates that $\bar{\boldsymbol{e}}$ is a binary vector and $\mathcal{M}_2$ indicates that $\boldsymbol{v}_i$ is the Hadamard product between $\boldsymbol{b}_i$ and $\boldsymbol{k}$. Then we set $\mathcal{M} = \mathcal{M}_1 \cup \mathcal{M}_2$. In the new relation, the length of the witness is $l_2 + 2l_1 l_2 + l_1 + kl_1$, and the size of $\mathcal{M}$ is $kl_1 + l_1 l_2$.

Remark 4.1. We remark that besides privacy-preserving primitives, our `ZKAoK` for weak PRF also implies a lattice-based verifiable random function (VRF) with trusted uniqueness (as formally defined in [71]).

More precisely, let λ be the security parameter. Let m, n, p, q be positive integers that are polynomial in λ, where $m \geq n(\log q + 1)/(\log p - 1)$. Let $\boldsymbol{A}$ be a random matrix in $\mathbb{Z}_q^{m \times n}$ and serves as a public parameter. The secret key of the VRF is a random vector $\boldsymbol{s} \in \mathbb{Z}_q^n$ and the public key is a vector $\boldsymbol{b} = \lfloor \boldsymbol{A} \cdot \boldsymbol{s} \rfloor_p \bmod p$. The evaluation algorithm outputs $\boldsymbol{y} = \lfloor H(x) \cdot \boldsymbol{s} \rfloor_p \bmod p$ on input a bitstring x, where H is a hash function that maps an arbitrary-length bitstrings onto a matrix in $\mathbb{Z}_q^{m \times n}$ and is modeled as a random oracle. The proof for the correct evaluation of the VRF on an input x is a `ZKAoK` that argues knowledge of a secret key $\boldsymbol{s}$ s.t. $\boldsymbol{b} = \lfloor \boldsymbol{A} \cdot \boldsymbol{s} \rfloor_p \wedge \boldsymbol{y} = \lfloor \boldsymbol{B} \cdot \boldsymbol{s} \rfloor_p$, where $\boldsymbol{B} = H(x)$ (Note that, we do not need to hide the matrices in this argument.).

First, as proved in [79], with all but negligible probability over the choise of $\boldsymbol{A}$, the secret key and the public key are bijective. Then the trusted uniqueness of the VRF follows directly from the soundness of the underlying arguments.

Acknowledgement. We appreciate the anonymous reviewers for their valuable suggestions. Part of this work was supported by the National Natural Science Foundation of China (Grant No. 61602396, 61572294, 61632020), Early Career Scheme research grant (ECS Grant No. 25206317) from the Research Grant Council of Hong Kong, the Innovation and Technology Support Programme of Innovation and Technology Fund of Hong Kong (Grant No. ITS/356/17), and the MonashU-PolyU-Collinstar Capital Joint Lab on Blockchain.

References

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

[2] Miklós Ajtai. Generating hard instances of lattice problems. In *STOC*, pages 99–108. ACM, 1996.

[3] Navid Alamati, Chris Peikert, and Noah Stephens-Davidowitz. New (and old) proof systems for lattice problems. In *PKC*, pages 619–643. Springer, 2018.

[4] Martin R Albrecht, Benjamin R Curtis, Amit Deo, Alex Davidson, Rachel Player, Eamonn W Postlethwaite, Fernando Virdia, and Thomas Wunderer. Estimate all the {LWE, NTRU} schemes! In *SCN*, pages 351–367. Springer, 2018.

[5] Martin R Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015.

[6] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange-a new hope. In *USENIX Security Symposium*, volume 2016, 2016.

[7] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *CRYPTO*, pages 595–618. Springer, 2009.

[8] Abhishek Banerjee and Chris Peikert. New and improved key-homomorphic pseudorandom functions. In *CRYPTO*, pages 353–370. Springer, 2014.

[9] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In *EUROCRYPT*, pages 719–737. Springer, 2012.

[10] Carsten Baum, Ivan Damgård, Vadim Lyubashevsky, Sabine Oechsner, and Chris Peikert. More efficient commitments from structured lattice assumptions. In *SCN*, pages 368–385. Springer, 2018.

[11] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *EUROCRYPT*, pages 614–629. Springer, 2003.

[12] Adam Bender, Jonathan Katz, and Ruggero Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. In *TCC*, pages 60–79. Springer, 2006.

[13] Fabrice Benhamouda, Jan Camenisch, Stephan Krenn, Vadim Lyubashevsky, and Gregory Neven. Better zero-knowledge proofs for lattice encryption and their application to group signatures. In *ASIACRYPT*, pages 551–572. Springer, 2014.

[14] Fabrice Benhamouda, Stephan Krenn, Vadim Lyubashevsky, and Krzysztof Pietrzak. Efficient zero-knowledge proofs for commitments from learning with errors over rings. In *ESORICS*, pages 305–325. Springer, 2015.

[15] Nina Bindel, Sedat Akeylek, Erdem Alkim, Paulo SLM Barreto, Johannes Buchmann, Edward Eaton, Gus Gutoski, Julaine Kramer, Patrick Longa, Harun Polat, et al. qtesla. submission to the nist?s post-quantum cryptography standardization process.(2018), 2018.

[16] Dan Boneh, Kevin Lewi, Hart Montgomery, and Ananth Raghunathan. Key homomorphic prfs and their applications. In *Advances in Cryptology–CRYPTO 2013*, pages 410–428. Springer, 2013.

[17] Jonathan Bootle, Vadim Lyubashevsky, and Gregor Seiler. Algebraic techniques for short(er) exact lattice-based zero-knowledge proofs. Cryptology ePrint Archive, Report 2019/642, 2019. To Appear at CRYPTO, 2019.

[18] Joppe Bos, Craig Costello, Léo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan, and Douglas Stebila. Frodo: Take off the ring! practical, quantum-secure key exchange from lwe. In *CCS*, pages 1006–1018. ACM, 2016.

[19] Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Kyber: a cca-secure module-lattice-based kem. In *EuroS&P*, pages 353–367. IEEE, 2018.

[20] Cecilia Boschini, Jan Camenisch, and Gregory Neven. Floppy-sized group signatures from lattices. In *ACNS*, pages 163–182. Springer, 2018.

[21] Cecilia Boschini, Jan Camenisch, and Gregory Neven. Relaxed lattice-based signatures with short zero-knowledge proofs. In *ISC*, pages 3–22. Springer, 2018.

[22] Johannes Buchmann, Florian Göpfert, Rachel Player, and Thomas Wunderer. On the hardness of lwe with binary error: revisiting the hybrid lattice-reduction and meet-in-the-middle attack. In *AFRICACRYPT*, pages 24–43. Springer, 2016.

[23] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact E-cash. In *EUROCRYPT*, pages 302–321. Springer, 2005.

[24] Jan Camenisch, Gregory Neven, and Markus Rückert. Fully anonymous attribute tokens from lattices. In *SCN*, pages 57–75. Springer, 2012.

[25] David Chaum. Blind signatures for untraceable payments. In *CRYPTO*, pages 199–203. Springer, 1982.

[26] David Chaum and Eugène Van Heyst. Group signatures. In *EUROCRYPT*, pages 257–265. Springer, 1991.

[27] Yuanmi Chen and Phong Q Nguyen. BKZ 2.0: Better lattice security estimates. In *ASIACRYPT*, pages 1–20. Springer, 2011.

[28] Ronald Cramer and Ivan Damgård. On the amortized complexity of zero-knowledge protocols. In *CRYPTO*, pages 177–191. Springer, 2009.

[29] Rafael del Pino, Vadim Lyubashevsky, and Gregor Seiler. Lattice-based group signatures and zero-knowledge proofs of automorphism stability. In *CCS*, pages 574–591. ACM, 2018.

[30] Yevgeniy Dodis, Aggelos Kiayias, Antonio Nicolosi, and Victor Shoup. Anonymous identification in ad hoc groups. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 609–626. Springer, 2004.

[31] Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal gaussians. In *CRYPTO*, pages 40–56. Springer, 2013.

[32] Leo Ducas, Tancrede Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehle. CRYSTALS – Dilithium: Digital signatures from module lattices. Cryptology ePrint Archive, Report 2017/633, 2017. `https://eprint.iacr.org/2017/633`.

[33] Muhammed F. Esgin, Ron Steinfeld, Amin Sakzad, Joseph K. Liu, and Dongxi Liu. Short lattice-based one-out-of-many proofs and applications to ring signatures. Cryptology ePrint Archive, Report 2018/773, 2018. `https://eprint.iacr.org/2018/773`.

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

[35] Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. Falcon: Fast-fourier lattice-based compact signatures over ntru. submission to the nist?s post-quantum cryptography standardization process.(2018), 2018.

[36] Nicolas Gama and Phong Q Nguyen. Predicting lattice reduction. In *EUROCRYPT*, pages 31–51. Springer, 2008.

[37] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206. ACM, 2008.

[38] Oded Goldreich and Shafi Goldwasser. On the limits of non-approximability of lattice problems. In *STOC*, pages 1–9. ACM, 1998.

[39] S Goldwasser, S Micali, and C Rackoff. The knowledge complexity of interactive proof-systems. In *STOC*, pages 291–304. ACM, 1985.

[40] Shafi Goldwasser and Dmitriy Kharchenko. Proof of plaintext knowledge for the ajtai-dwork cryptosystem. In *TCC*, pages 529–555. Springer, 2005.

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

[42] Jens Groth and Markulf Kohlweiss. One-out-of-many proofs: Or how to leak a secret and spend a coin. In *EUROCRYPT*, pages 253–280. Springer, 2015.

[43] Venkatesan Guruswami, Daniele Micciancio, and Oded Regev. The complexity of the covering radius problem. *Computational Complexity*, 14(2):90–121, 2005.

[44] Jeff Hoffstein, Jill Pipher, John M Schanck, Joseph H Silverman, William Whyte, and Zhenfei Zhang. Choosing parameters for ntruencrypt. In *CT-RSA*, pages 3–18. Springer, 2017.

[45] Jeffrey Hoffstein, Jill Pipher, and Joseph H Silverman. Ntru: A ring-based public key cryptosystem. In *International Algorithmic Number Theory Symposium*, pages 267–288. Springer, 1998.

[46] Jeffrey Hoffstein, Jill Pipher, William Whyte, and Zhenfei Zhang. A signature scheme from learning with truncation. Cryptology ePrint Archive, Report 2017/995, 2017. `https://eprint.iacr.org/2017/995`.

[47] Akinori Kawachi, Keisuke Tanaka, and Keita Xagawa. Concurrently secure identification schemes based on the worst-case hardness of lattice problems. In *ASIACRYPT*, pages 372–389. Springer, 2008.

[48] Ahmed Kosba, Zhichao Zhao, Andrew Miller, Yi Qian, Hubert Chan, Charalampos Papamanthou, Rafael Pass, abhi shelat, and Elaine Shi. C$\emptyset$c$\emptyset$: A framework for building composable zero-knowledge proofs. Cryptology ePrint Archive, Report 2015/1093, 2015. `https://eprint.iacr.org/2015/1093`.

[49] Thijs Laarhoven. Sieving for shortest vectors in lattices using angular locality-sensitive hashing. In *CRYPTO*, pages 3–22. Springer, 2015.

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

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

[52] Benoît Libert, San Ling, Fabrice Mouhartem, Khoa Nguyen, and Huaxiong Wang. Signature schemes with efficient protocols and dynamic group signatures from lattice assumptions. In *ASIACRYPT*, pages 373–403. Springer, 2016.

[53] Benoît Libert, San Ling, Fabrice Mouhartem, Khoa Nguyen, and Huaxiong Wang. Zero-knowledge arguments for matrix-vector relations and lattice-based group encryption. In *ASIACRYPT*, pages 101–131. Springer, 2016.

[54] Benoît Libert, San Ling, Khoa Nguyen, and Huaxiong Wang. Zero-knowledge arguments for lattice-based accumulators: logarithmic-size ring signatures and group signatures without trapdoors. In *EUCRYPT*, pages 1–31. Springer, 2016.

[55] Benoît Libert, San Ling, Khoa Nguyen, and Huaxiong Wang. Zero-knowledge arguments for lattice-based PRFs and applications to E-cash. In *ASIACRYPT*, pages 304–335. Springer, 2017.

[56] Benoît Libert, San Ling, Khoa Nguyen, and Huaxiong Wang. Lattice-based zero-knowledge arguments for integer relations. In *CRYPTO*, pages 700–732. Springer, 2018.

[57] Richard Lindner and Chris Peikert. Better key sizes (and attacks) for lwe-based encryption. In *CT-RSA*, pages 319–339. Springer, 2011.

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

[59] San Ling, Khoa Nguyen, and Huaxiong Wang. Group signatures from lattices: simpler, tighter, shorter, ring-based. In *PKC*, pages 427–449. Springer, 2015.

[60] San Ling, Khoa Nguyen, Huaxiong Wang, and Yanhong Xu. Lattice-based group signatures: Achieving full dynamicity with ease. In *ACNS*, pages 293–312. Springer, 2017.

[61] San Ling, Khoa Nguyen, Huaxiong Wang, and Yanhong Xu. Constant-size group signatures from lattices. In *PKC*, pages 58–88. Springer, 2018.

[62] Vadim Lyubashevsky. Lattice-based identification schemes secure under active attacks. In *PKC*, pages 162–179. Springer, 2008.

[63] Vadim Lyubashevsky. Fiat-shamir with aborts: Applications to lattice and factoring-based signatures. In *ASIACRYPT*, pages 598–616. Springer, 2009.

[64] Vadim Lyubashevsky. Lattice signatures without trapdoors. In *EUROCRYPT*, pages 738–755. Springer, 2012.

[65] Vadim Lyubashevsky and Gregory Neven. One-shot verifiable encryption from lattices. In *EUROCRYPT*, pages 293–323. Springer, 2017.

[66] D Micciancio and O Regev. Worst-case to average-case reductions based on gaussian measures. In *FOCS*, pages 372–381. IEEE, 2004.

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

[68] Daniele Micciancio and Chris Peikert. Hardness of SIS and LWE with small parameters. In *CRYPTO*, pages 21–39. Springer, 2013.

[69] Daniele Micciancio and Salil P Vadhan. Statistical zero-knowledge proofs with efficient provers: Lattice problems and more. In *CRYPTO*, pages 282–298. Springer, 2003.

[70] Phong Q Nguyen, Jiang Zhang, and Zhenfeng Zhang. Simpler efficient group signatures from lattices. In *PKC*, pages 401–426. Springer, 2015.

[71] Dimitrios Papadopoulos, Duane Wessels, Shumon Huque, Moni Naor, Jan Včelák, Leonid Reyzin, and Sharon Goldberg. Making NSEC5 practical for DNSSEC. Cryptology ePrint Archive, Report 2017/099, 2017. `https://eprint.iacr.org/2017/099`.

[72] Chris Peikert. A decade of lattice cryptography. *Foundations and Trends® in Theoretical Computer Science*, 10(4):283–424, 2016.

[73] Chris Peikert and Vinod Vaikuntanathan. Noninteractive statistical zero-knowledge proofs for lattice problems. In *CRYPTO*, pages 536–553. Springer, 2008.

[74] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, pages 84–93. ACM, 2005.

[75] Ronald L Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In *ASIACRYPT*, pages 552–565. Springer, 2001.

[76] John Schanck. Estimator. `https://github.com/jschanck/estimator`.

[77] Jacques Stern. A new identification scheme based on syndrome decoding. In *CRYPTO*, pages 13–21. Springer, 1993.

[78] Gene Tsudik and Shouhuai Xu. Accumulating composites and improved group signing. In *ASIACRYPT*, pages 269–286. Springer, 2003.

[79] Rupeng Yang, Man Ho Au, Junzuo Lai, Qiuliang Xu, and Zuoxia Yu. Lattice-based techniques for accountable anonymity: Composition of abstract sterns protocols and weak PRF with efficient protocols from LWR. Cryptology ePrint Archive, Report 2017/781, 2017. `http://eprint.iacr.org/2017/781`.

A Cryptographic Primitives

In this section, we recall cryptographic primitives employed in our applications.

A.1 Commitment

In this work, we will use the commitment scheme from [47]. Let λ be the security parameter. Let n, q, L be postive integers that are polynomial in λ, and let $k = \lceil \log q \rceil$, $m = n(k+3)$. The public parameter of the commitment scheme is two random matrices $\boldsymbol{B}_1 \in \mathbb{Z}_q^{n\times m}$ and $\boldsymbol{B}_2 \in \mathbb{Z}_q^{n\times L}$. To commit a message $\boldsymbol{m} \in \{0,1\}^L$, the commit algorithm first samples $\boldsymbol{r} \in \{0,1\}^m$. Then it outputs the commitment $\boldsymbol{c} = \boldsymbol{B}_1 \cdot \boldsymbol{r} + \boldsymbol{B}_2 \cdot \boldsymbol{m}$ and the opening $\boldsymbol{r}$. The open algorithm outputs 1 on input $\boldsymbol{B}_1, \boldsymbol{B}_2, \boldsymbol{m}, \boldsymbol{c}, \boldsymbol{r}$ iff $\boldsymbol{c} = \boldsymbol{B}_1 \cdot \boldsymbol{r} + \boldsymbol{B}_2 \cdot \boldsymbol{m}$.

A.2 Public Key Encryption

In this work, we will use the CPA secure public key encryption scheme from [57]. Let λ be the security parameter. Let n_1, n_2, q, L, σ be postive integers that are polynomial in λ. The PKE scheme works as follows:

- **KeyGen.** On input the security parameter, the key generation algorithm first samples $\boldsymbol{D}_0 \xleftarrow{\$} \mathbb{Z}_q^{n_1\times n_2}$, $\boldsymbol{S} \leftarrow D_\sigma^{L\times n_1}$, and $\boldsymbol{E} \leftarrow D_\sigma^{L\times n_2}$. Then it computes $\boldsymbol{D}_1 = \boldsymbol{S}\cdot \boldsymbol{D}_0 + \boldsymbol{E}$. The public key is $(\boldsymbol{D}_0, \boldsymbol{D}_1)$ and the secret key is $\boldsymbol{S}$.
- **Enc.** On input the public key $(\boldsymbol{D}_0, \boldsymbol{D}_1)$ and a message $\boldsymbol{m} \in \{0,1\}^L$, the encryption algorithm first samples $\boldsymbol{r} \leftarrow D_\sigma^{n_2}$, $\boldsymbol{e}_1 \leftarrow D_\sigma^{n_1}$, $\boldsymbol{e}_2 \leftarrow D_\sigma^{L}$. Then it computes $\boldsymbol{c}_1 = \boldsymbol{D}_0 \cdot \boldsymbol{r} + \boldsymbol{e}_1$, $\boldsymbol{c}_2 = \boldsymbol{D}_1 \cdot \boldsymbol{r} + \boldsymbol{e}_2 + \lfloor \frac{q}{2} \rceil \cdot \boldsymbol{m}$ and outputs the ciphertext $(\boldsymbol{c}_1, \boldsymbol{c}_2)$.
- **Dec.** On input the secret key $\boldsymbol{S}$ and a ciphertext $(\boldsymbol{c}_1, \boldsymbol{c}_2)$, the decryption algorithm first computes $\boldsymbol{m}' = \boldsymbol{c}_2 - \boldsymbol{S}\cdot \boldsymbol{c}_1$. Then for $j \in [1, L]$, it sets $m_j = 0$ if $\|\boldsymbol{m}'[j]\| < \lfloor q/4 \rceil$ and $m_j = 1$ otherwise. Finally, it outputs $(m_1, \ldots, m_L)^\intercal$.

We will also use a variant of this scheme for our applications. In particular, in this variant, all elements in $\boldsymbol{S}, \boldsymbol{E}, \boldsymbol{r}, \boldsymbol{e}$ are sampled from $\{0,1\}$ instead of the Gaussian distribution D_σ. Security of the variant relies on the BLWE assumption ranther than the standard LWE assumption.

A.3 Signature

In this work, we will use the signature scheme from [55]. Let λ be the security parameter. Let $m_1, m_2, m_3, \sigma, q, L$ be positive integers that are polynomial in λ. Let $k = \lceil \log q \rceil$. The signautre scheme works as follows:

- **KeyGen.** On input the security parameter, the key generation algorithm first samples a random matrix $\boldsymbol{B} \in \mathbb{Z}_q^{m_1\times m_2}$ together with its trapdoor $\boldsymbol{T}$. Then it samples $\boldsymbol{B}_i \xleftarrow{\$} \mathbb{Z}_q^{m_1\times m_2}$ for $i \in [0, m_3]$ and samples $\tilde{\boldsymbol{B}} \xleftarrow{\$} \mathbb{Z}_q^{m_1\times k m_1}$, $\tilde{\boldsymbol{B}}_0 \xleftarrow{\$} \mathbb{Z}_q^{m_1\times 2m_2}$, $\tilde{\boldsymbol{B}}_1 \xleftarrow{\$} \mathbb{Z}_q^{m_1\times L}$ and $\boldsymbol{u} \xleftarrow{\$} \mathbb{Z}_q^{m_1}$. Finally, it outputs $sk = \boldsymbol{T}$ and $pk = (\boldsymbol{B}, \{\boldsymbol{B}_i\}_{i\in[0,m_3]}, \boldsymbol{u}, \tilde{\boldsymbol{B}}, \tilde{\boldsymbol{B}}_{\boldsymbol{0}}, \tilde{\boldsymbol{B}}_{\boldsymbol{1}})$.
- **Sign.** On input the secret key $\boldsymbol{T}$ and a message $\boldsymbol{m} \in \{0,1\}^L$, the sign algorithm first samples $\boldsymbol{\tau} \xleftarrow{\$} \{0,1\}^{m_3}$. Then it computes the matrix $\boldsymbol{B}_{\boldsymbol{\tau}} = (\boldsymbol{B} \| \boldsymbol{B}_0 + \sum_{i=1}^{m_3} (\boldsymbol{\tau}[i] \cdot \boldsymbol{B}_i))$. Next, it samples $\boldsymbol{r} \leftarrow D_\sigma^{2m_2}$ and computes $\boldsymbol{c} = \tilde{\boldsymbol{B}}_0 \cdot \boldsymbol{r} + \tilde{\boldsymbol{B}}_1 \cdot \boldsymbol{m}$. Then it uses the secret key $\boldsymbol{T}$ to samples a vector $\boldsymbol{v} \in D_\sigma^{2m_2}$ that satisfies $\boldsymbol{B}_{\boldsymbol{\tau}} \cdot \boldsymbol{v} = \boldsymbol{u} + \tilde{\boldsymbol{B}} \cdot \mathtt{bin}(\boldsymbol{c})$. The signature is $(\boldsymbol{\tau}, \boldsymbol{r}, \boldsymbol{v})$.

- **Verify.** On input the public key $(\boldsymbol{B}, \{\boldsymbol{B}_i\}_{i\in[0,m_3]}, \boldsymbol{u}, \tilde{\boldsymbol{B}}, \tilde{\boldsymbol{B}}_{\boldsymbol{0}}, \tilde{\boldsymbol{B}}_{\boldsymbol{1}})$, a message $\boldsymbol{m}$ and a signature $(\boldsymbol{\tau}, \boldsymbol{r}, \boldsymbol{v}) \in \{0,1\}^{m_3} \times \mathbb{Z}^{2m_2} \times \mathbb{Z}^{2m_2}$, the verification algorithm first computes $\boldsymbol{B}_{\boldsymbol{\tau}} = (\boldsymbol{B} \| \boldsymbol{B}_0 + \sum_{i=1}^{m_3}(\boldsymbol{\tau}[i] \cdot \boldsymbol{B}_i))$. Then it checks if

$$\boldsymbol{B}_{\boldsymbol{\tau}} \cdot \boldsymbol{v} = \boldsymbol{u} + \tilde{\boldsymbol{B}} \cdot \mathtt{bin}(\tilde{\boldsymbol{B}}_0 \cdot \boldsymbol{r} + \tilde{\boldsymbol{B}}_1 \cdot \boldsymbol{m})$$

  and $\boldsymbol{v}, \boldsymbol{r}$ are short vectors (here, we say the vector is small if its infinity norm does not exceed 12σ).

One feature of this signature scheme is that it allows one to sign on the committed value in a commitment if the commitment is generated under public parameter $(\tilde{\boldsymbol{B}}_0, \tilde{\boldsymbol{B}}_1)$.

Security of the signature scheme relies on the $SIS_{m_1,q,\beta}$ assumption, where $\beta \approx \sigma^2 \cdot m_2^{3/2} \cdot m_3$. Also, when applying the trapdoor generation algorithm presented in [67], we can set $m_2 = 2km_1$ and $\sigma = 1.6k \cdot \sqrt{m_1}$.

A.4 Cryptographic Accumulator

In this work, we will use the accumulator scheme presented in [54]. Let λ be the security parameter. Let n, q be a postive integer that is polynomial in λ. Let L be a constant. Let $N = 2^L$, $k = \lceil \log q \rceil$ and $m = nk$. The accumulator scheme works as follows:

- **Setup.** The setup algorithm samples matrices $\boldsymbol{B}_1, \boldsymbol{B}_2 \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$ and outputs the public parameter $para = (\boldsymbol{B}_1, \boldsymbol{B}_2)$.
- **Acc.** On input a set $\mathtt{R} = \{\boldsymbol{d}_i\}_{i\in[0,N-1]} \in (\{0,1\}^m)^N$, the accumulate algorithm sets $\boldsymbol{u}_{L,i} = \boldsymbol{d}_i$ for $i \in [0, N-1]$. Then for $j \in [0, L-1]$, it sets $\boldsymbol{u}_{j,i} = \mathtt{bin}(\boldsymbol{B}_1 \cdot \boldsymbol{u}_{j+1,2i} + \boldsymbol{B}_2 \cdot \boldsymbol{u}_{j+1,2i+1})$ for $i \in [0, 2^j - 1]$. Finally, it outputs the accumulator $\boldsymbol{u}_{0,0}$.
- **Witness.** On input a set $\mathtt{R} = \{\boldsymbol{d}_i\}_{i\in[0,N-1]} \in (\{0,1\}^m)^N$ and an element $\boldsymbol{d} = \boldsymbol{d}_{i^*}$, the witness algorithm first generates $\boldsymbol{u}_{j,i}$ as in the accumulate algorithm. Then it outputs $(\mathtt{bin}(i^*), \{\boldsymbol{u}_{j,f(j)}\}_{j\in[1,L]}, \{\boldsymbol{u}_{j,g(j)}\}_{j\in[1,L]})$, where $f(j) = \lfloor i^*/(2^{L-j}) \rfloor$ and $g(j) = 4\lfloor i^*/(2^{L-j-1}) \rfloor - \lfloor i^*/(2^{L-j}) \rfloor + 1$
- **Verify.** On input an accumulator $\boldsymbol{u}$, an element $\boldsymbol{d}$ and a witness $(\boldsymbol{m}, \{\boldsymbol{v}_j\}_{j\in[1,L]}, \{\boldsymbol{w}_j\}_{j\in[1,L]})$, the verification algorithm outputs 1 iff

$$\begin{cases} \mathtt{bin}(\boldsymbol{B}_{1+\boldsymbol{m}[1]} \cdot \boldsymbol{v}_1 + \boldsymbol{B}_{2-\boldsymbol{m}[1]} \cdot \boldsymbol{w}_1) = \boldsymbol{u} \\ \forall i \in [2, L], \mathtt{bin}(\boldsymbol{B}_{1+\boldsymbol{m}[i]} \cdot \boldsymbol{v}_i + \boldsymbol{B}_{2-\boldsymbol{m}[i]} \cdot \boldsymbol{w}_i) = \boldsymbol{v}_{i-1} \end{cases}$$

A.5 (Weak) Pseudorandom Function

In this work, we will use the weak PRF constructed implicitly in [9]. Let λ be the security parameter. Let n_1, n_2, p, q be postive integers that are polynomial in λ. The weak PRF works as follows:

- **KeyGen.** The key generation algorithm samples $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_q^{n_2}$ and outputs the key $\boldsymbol{s}$.
- **Eval.** On input an input $\boldsymbol{A} \xleftarrow{\$} \mathbb{Z}_q^{n_1\times n_2}$, the evaluation algorithm outputs $\boldsymbol{y} = \lfloor \boldsymbol{A} \cdot \boldsymbol{s} \rfloor_p \mod p$.

B Additional Assumptions and Concrete Hardness

We will also use variants of the LWE assumption when constructing our applications. Employed variants include the LWE assumption with binary secrets and errors (BLWE) and the learning with rounding (LWR) assumption. Both assumptions are proved as hard as the standard LWE assumption for specific parameters [9, 68].

Definition B.1 (Decision-BLWE$_{n,m,q}$). *Given a random matrix $\boldsymbol{A} \in \mathbb{Z}_q^{m\times n}$, and a vector $\boldsymbol{b} \in \mathbb{Z}_q^m$, where $\boldsymbol{b}$ is generated according to either of the following two cases:*
1. $\boldsymbol{b} = \boldsymbol{A}\cdot\boldsymbol{s} + \boldsymbol{e}$, where $\boldsymbol{s} \overset{\$}{\leftarrow} \{0,1\}^n$ and $\boldsymbol{e} \leftarrow \{0,1\}^m$
2. $\boldsymbol{b} \overset{\$}{\leftarrow} \mathbb{Z}_q^m$
distinguish which is the case with non-negligible advantage.

As the hardness of the BLWE assumption usually depends only on n, q (assuming m is large enough), in this work, we write $BLWE_{n,m,q}$ as $BLWE_{n,q}$ for short.

Definition B.2 (Decision-LWR$_{n,m,q,p}$). *Given a random matrix $\boldsymbol{A} \in \mathbb{Z}_q^{m\times n}$, and a vector $\boldsymbol{b} \in \mathbb{Z}_p^m$, where $\boldsymbol{b}$ is generated according to either of the following two cases:*
1. $\boldsymbol{b} = \lfloor\boldsymbol{A}\cdot\boldsymbol{s}\rfloor_p$, where $\boldsymbol{s} \overset{\$}{\leftarrow} \mathbb{Z}_q^n$.
2. $\boldsymbol{b} \overset{\$}{\leftarrow} \lfloor\mathbb{Z}_q^m\rfloor_p$
distinguish which is the case with non-negligible advantage. Here, for any $\boldsymbol{x} \in \mathbb{Z}_q^m$, we define $\lfloor\boldsymbol{x}\rfloor_p = \lfloor p/q\cdot\boldsymbol{x}\rfloor \in \mathbb{Z}_p^m$.

As the hardness of the LWR assumption usually depends only on n, q, p (assuming m is large enough), in this work, we write $LWR_{n,m,q,p}$ as $LWR_{n,q,p}$ for short.

B.1 Concrete Hardness of Assumptions

In lattice-based cryptography, a hardness theorem is often used to demonstrate that the overall protocol comes from a provable secure design. Nonetheless, parameters stem from those theorems are usually impractical and it is more desirable to set parameters that are robust against best known cryptanalysis. This approach has been adopted in most practical lattice-based schemes, such as [4, 6, 32, 44].

We adopt the same approach to estimate the security and derive parameters for applications in this paper. Concretely, we examine the root Hermite factor (RHF) [36] for each lattice problem and summarize the required RHF for each problem in Table 3.

Then we use the estimator from [76] to estimate the cost of BKZ [27] algorithm with quantum-core-sieving [4, 6, 49] to arrive those root Hermite factors. In general, to achieve a 80 / 100 / 128-bit security, the corresponding RHF are 1.0048 / 1.0042 / 1.0035, respectively.

Problem	RHF	Reference
$SIS_{n,m,q,\beta}$	$2^{\frac{\log^2 \beta}{4n \log q}}$	[48]
$LWE_{n,m,q,\alpha}$	$2^{\frac{\log^2 \frac{\alpha}{5.31}}{4n \log q}}$	[5]
$BLWE_{n,m,q}$	$2^{\frac{\log^2 (q \cdot 4.24)}{4n \log q}}$	[22]
$LWR_{n,m,q,p}$	$2^{\frac{\log^2 (\frac{\sqrt{\pi/6}}{5.31p})}{4n \log q}}$	[4]

Table 3: The required RHF to solve lattice problems. Note that $LWR_{n,m,q,p}$ is treated as an $LWE_{n,m,q,\alpha}$ instance with $\alpha = \sqrt{\frac{(q/p)^2-1}{12}} \cdot \sqrt{2\pi}/q$.

C Proof of Theorem 3.1

Proof. We prove Theorem 3.1 by proving the completeness, the proof of knowledge property and the honest-verifier zero-knowledge property of P_1.

Completeness. Let $\boldsymbol{A}, \boldsymbol{y}, \mathcal{M}$ be the valid statement to be proved, let $\boldsymbol{x}$ be the witness, let $\boldsymbol{B}_1, \boldsymbol{B}_2$ be the public parameter, let $\boldsymbol{r}, \boldsymbol{t}, \boldsymbol{s}_1, \boldsymbol{s}_2, \boldsymbol{s}_3, \boldsymbol{s}_4, \boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{c}_3, \boldsymbol{c}_4, \boldsymbol{a}, \boldsymbol{b}, \rho, C_{aux}, \alpha, \boldsymbol{z}_0, \boldsymbol{z}_1, \boldsymbol{z}_2, \boldsymbol{d}$ be the variables used in an honest executation of the protocol.

First, let $\boldsymbol{v} = (\alpha \cdot \boldsymbol{s}_1^\intercal \| \alpha \cdot \boldsymbol{s}_3^\intercal)^\intercal$, then by Lemma 2.1, we have $\|\boldsymbol{v}\| \le 2p \cdot \sqrt{l} \cdot \sigma_1$ with all but negligible probability. That is, the norm of the masked vector $\boldsymbol{v}$ is bounded by $T = 2p \cdot \sqrt{l} \cdot \sigma_1$. Since $\sigma_2 = 2p \cdot \sqrt{l} \cdot \log l \cdot \sigma_1 = \log l \cdot T$ and $M = e^{12/\log l + 1/(2\log^2 l)}$, by Lemma 2.2, the prover will response in the third move with a probability of at least $\frac{1-2^{-100}}{M}$.

Now, assume the prover response in the third move. It is easy to see that Condition 1, Condition 4 and Condition 5 (in the verifier's checklist) are satisfied. Next, by Lemma 2.1, we have $\|\alpha \cdot \boldsymbol{s}_1\| \le 2p \cdot \sqrt{l_1 + l_2 + n} \cdot \sigma_1$ with all but negligible probability and $\|\boldsymbol{s}_2\| \le 2 \cdot \sqrt{l_1 + l_2 + n} \cdot \sigma_2$ with all but negligible probability. Then by the triangle inequality, Condition 2 is satisfied with all but negligible probability. Similar, we can argue that Condition 3 is satisfied with all but negligible probability.

Finally, we argue that the last condition is also satisfied. First, for any $k \in [1, \ell]$, let (h, i, j) be the k-th element in $\mathcal{M}$, we have

$$\begin{aligned}\boldsymbol{d}[k] =& \alpha \cdot \boldsymbol{z}_0[h] - \boldsymbol{z}_0[i] \cdot \boldsymbol{z}_0[j] \\ =& \alpha^2 \cdot \boldsymbol{x}[h] + \alpha \cdot \boldsymbol{r}[h] - \alpha^2 \cdot \boldsymbol{x}[i] \cdot \boldsymbol{x}[j] - \\ & \alpha \cdot \boldsymbol{x}[i] \cdot \boldsymbol{r}[j] - \alpha \cdot \boldsymbol{x}[j] \cdot \boldsymbol{r}[i] - \boldsymbol{r}[i] \cdot \boldsymbol{r}[j] \\ =& \alpha \cdot (\boldsymbol{r}[h] - \boldsymbol{x}[i] \cdot \boldsymbol{r}[j] - \boldsymbol{x}[j] \cdot \boldsymbol{r}[i]) - \boldsymbol{r}[i] \cdot \boldsymbol{r}[j]\end{aligned}$$

where the last equation holds since the witness $\boldsymbol{x}$ satisfies $\boldsymbol{x}[h] = \boldsymbol{x}[i] \cdot \boldsymbol{x}[j]$. This implies that $\boldsymbol{d} = \alpha \cdot \boldsymbol{a} - \boldsymbol{b}$ and Condition 6 follows.

In summary, for an honest prover and an honest verifier, the prover will response in the third move with a probability that is negligibly close to $1/M$ and once the prover responses, the verifier will accept with all but negligible probability, thus, the protocol is complete with a completeness error of $1-1/M$.

Proof of Knowledge. Let $\boldsymbol{A}, \boldsymbol{y}, \mathcal{M}$ be a statement, let $\boldsymbol{B}_1, \boldsymbol{B}_2$ be the public parameter. Suppose a cheating prover $\hat{P}$ can convince the verifier that he possesses a valid witness for $\boldsymbol{A}, \boldsymbol{y}, \mathcal{M}$ with probability $\frac{2}{2p+1} + \epsilon$ for some non-negligible ϵ, then we construct a knowledge extractor that can extract a valid witness for $\boldsymbol{A}, \boldsymbol{y}, \mathcal{M}$ via invoking $\hat{P}$.

First, note that the output of $\hat{P}$ is determined by its inner randomness and the challenge α, which is sampled from $2p+1$ possible values. Since $\hat{P}$ can convince the verifier with a probability that is non-negligibly larger than $\frac{2}{2p+1}$, with a non-negligible probability over the choise of $\hat{P}$'s inner randomness, $\hat{P}$ can answer at least 3 challenges correctly. Thus, via invoking the cheating prover $\hat{P}$ polynomial times, the extractor could extract three valid proofs with different challenges and the same first-move response. Moreover, by the binding property of the auxiliary commitment, the extractor is able to obtain

$$\begin{aligned}
&(\alpha, \boldsymbol{t}, \boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{c}_3, \boldsymbol{c}_4, \boldsymbol{z}_0, \boldsymbol{z}_1, \boldsymbol{z}_2)\\
&(\alpha', \boldsymbol{t}, \boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{c}_3, \boldsymbol{c}_4, \boldsymbol{z}'_0, \boldsymbol{z}'_1, \boldsymbol{z}'_2)\\
&(\alpha'', \boldsymbol{t}, \boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{c}_3, \boldsymbol{c}_4, \boldsymbol{z}''_0, \boldsymbol{z}''_1, \boldsymbol{z}''_2)
\end{aligned}$$

for distinct α, α' and α'' that satisfies

$$\begin{cases}
\|\boldsymbol{z}_1\| \leq 2\sqrt{l_1 + l_2 + n} \cdot (\sigma_2 + p \cdot \sigma_1)\\
\|\boldsymbol{z}_2\| \leq 2\sqrt{l_1 + l_2 + \ell} \cdot (\sigma_2 + p \cdot \sigma_1)\\
\boldsymbol{A} \cdot \boldsymbol{z}_0 = \alpha \cdot \boldsymbol{y} + \boldsymbol{t}\\
\boldsymbol{B}_1 \cdot \boldsymbol{z}_1 + (\mathbf{0}^\intercal \| \boldsymbol{z}_0^\intercal)^\intercal = \alpha \cdot \boldsymbol{c}_1 + \boldsymbol{c}_2\\
\boldsymbol{B}_2 \cdot \boldsymbol{z}_2 + (\mathbf{0}^\intercal \| \boldsymbol{d}^\intercal)^\intercal = \alpha \cdot \boldsymbol{c}_3 - \boldsymbol{c}_4
\end{cases}$$

$$\begin{cases}
\|\boldsymbol{z}'_1\| \leq 2\sqrt{l_1 + l_2 + n} \cdot (\sigma_2 + p \cdot \sigma_1)\\
\|\boldsymbol{z}'_2\| \leq 2\sqrt{l_1 + l_2 + \ell} \cdot (\sigma_2 + p \cdot \sigma_1)\\
\boldsymbol{A} \cdot \boldsymbol{z}'_0 = \alpha' \cdot \boldsymbol{y} + \boldsymbol{t}\\
\boldsymbol{B}_1 \cdot \boldsymbol{z}'_1 + (\mathbf{0}^\intercal \| \boldsymbol{z}_0'^\intercal)^\intercal = \alpha' \cdot \boldsymbol{c}_1 + \boldsymbol{c}_2\\
\boldsymbol{B}_2 \cdot \boldsymbol{z}'_2 + (\mathbf{0}^\intercal \| \boldsymbol{d}'^\intercal)^\intercal = \alpha' \cdot \boldsymbol{c}_3 - \boldsymbol{c}_4
\end{cases}$$

$$\begin{cases}
\|\boldsymbol{z}''_1\| \leq 2\sqrt{l_1 + l_2 + n} \cdot (\sigma_2 + p \cdot \sigma_1)\\
\|\boldsymbol{z}''_2\| \leq 2\sqrt{l_1 + l_2 + \ell} \cdot (\sigma_2 + p \cdot \sigma_1)\\
\boldsymbol{A} \cdot \boldsymbol{z}''_0 = \alpha'' \cdot \boldsymbol{y} + \boldsymbol{t}\\
\boldsymbol{B}_1 \cdot \boldsymbol{z}''_1 + (\mathbf{0}^\intercal \| \boldsymbol{z}_0''^\intercal)^\intercal = \alpha'' \cdot \boldsymbol{c}_1 + \boldsymbol{c}_2\\
\boldsymbol{B}_2 \cdot \boldsymbol{z}''_2 + (\mathbf{0}^\intercal \| \boldsymbol{d}''^\intercal)^\intercal = \alpha'' \cdot \boldsymbol{c}_3 - \boldsymbol{c}_4
\end{cases}$$

where for $k \in [1, \ell]$, let (h, i, j) be the k-th element in $\mathcal{M}$,

$$\boldsymbol{d}[k] = \alpha \cdot \boldsymbol{z}_0[h] - \boldsymbol{z}_0[i] \cdot \boldsymbol{z}_0[j]$$

$$\boldsymbol{d}'[k] = \alpha' \cdot \boldsymbol{z}'_0[h] - \boldsymbol{z}'_0[i] \cdot \boldsymbol{z}'_0[j]$$

$$\boldsymbol{d}''[k] = \alpha'' \cdot \boldsymbol{z}''_0[h] - \boldsymbol{z}''_0[i] \cdot \boldsymbol{z}''_0[j]$$

Now, let $\Delta_1 = \alpha' - \alpha$ and $\Delta_2 = \alpha'' - \alpha$. The output of the extractor is the vector $\bar{\boldsymbol{x}} = \Delta_1^{-1} \cdot (\boldsymbol{z}'_0 - \boldsymbol{z}_0)$. [8] Next, we argue that with all but negligible probability $((\boldsymbol{A}, \boldsymbol{y}, \mathcal{M}), (\bar{\boldsymbol{x}})) \in \mathcal{R}^*$ via proving the following lemmas.

Lemma C.1. $\boldsymbol{A} \cdot \bar{\boldsymbol{x}} = \boldsymbol{y}$.

Proof. This is because

$$\begin{aligned}
\boldsymbol{A} \cdot \bar{\boldsymbol{x}} &= \boldsymbol{A} \cdot \Delta_1^{-1} \cdot (\boldsymbol{z}'_0 - \boldsymbol{z}_0) \\
&= \Delta_1^{-1} \cdot (\boldsymbol{A} \cdot \boldsymbol{z}'_0 - \boldsymbol{A} \cdot \boldsymbol{z}_0) \\
&= \Delta_1^{-1} \cdot (\alpha \cdot \boldsymbol{y} + \boldsymbol{t} - \alpha' \cdot \boldsymbol{y} - \boldsymbol{t}) \\
&= \boldsymbol{y}
\end{aligned}$$

□

Claim 1. *Under the SIS assumption, $\bar{\boldsymbol{x}} = \Delta_2^{-1} \cdot (\boldsymbol{z}''_0 - \boldsymbol{z}_0)$ with all but negligible probability.*

Proof. Let $\boldsymbol{e}' = \boldsymbol{z}'_1 - \boldsymbol{z}_1$, $\boldsymbol{e}'' = \boldsymbol{z}''_1 - \boldsymbol{z}_1$, $\boldsymbol{f}' = \boldsymbol{z}'_0 - \boldsymbol{z}_0$, $\boldsymbol{f}'' = \boldsymbol{z}''_0 - \boldsymbol{z}_0$. Then we have

$$\begin{cases} \boldsymbol{B}_1 \cdot \boldsymbol{e}' + (\boldsymbol{0}^\intercal \| \boldsymbol{f}'^\intercal)^\intercal = \Delta_1 \cdot \boldsymbol{c}_1 \\ \boldsymbol{B}_1 \cdot \boldsymbol{e}'' + (\boldsymbol{0}^\intercal \| \boldsymbol{f}''^\intercal)^\intercal = \Delta_2 \cdot \boldsymbol{c}_1 \end{cases}$$

which implies

$$\begin{cases} \Delta_2 \cdot \boldsymbol{B}_1 \cdot \boldsymbol{e}' + (\boldsymbol{0}^\intercal \| (\Delta_2 \cdot \boldsymbol{f}')^\intercal)^\intercal = \Delta_1 \cdot \Delta_2 \cdot \boldsymbol{c}_1 \\ \Delta_1 \cdot \boldsymbol{B}_1 \cdot \boldsymbol{e}'' + (\boldsymbol{0}^\intercal \| (\Delta_1 \cdot \boldsymbol{f}'')^\intercal)^\intercal = \Delta_1 \cdot \Delta_2 \cdot \boldsymbol{c}_1 \end{cases}$$

Thus we have

$$\boldsymbol{B}_1 \cdot (\Delta_2 \cdot \boldsymbol{e}' - \Delta_1 \cdot \boldsymbol{e}'') + (\boldsymbol{0}^\intercal \| (\Delta_2 \cdot \boldsymbol{f}' - \Delta_1 \cdot \boldsymbol{f}'')^\intercal)^\intercal = \boldsymbol{0}$$

i.e.,

$$[\boldsymbol{I}_{l_1} \parallel \boldsymbol{B}_{1,1}] \cdot (\Delta_2 \cdot \boldsymbol{e}' - \Delta_1 \cdot \boldsymbol{e}'') = \boldsymbol{0}$$

Since $\boldsymbol{z}_1$, $\boldsymbol{z}'_1$ and $\boldsymbol{z}''_1$ are all bounded by $\leq 2\sqrt{l_1 + l_2 + n} \cdot (\sigma_2 + p \cdot \sigma_1)$, it is easy to show that $\|\Delta_2 \cdot \boldsymbol{e}' - \Delta_1 \cdot \boldsymbol{e}''\| \leq 16p \cdot \sqrt{l_1 + l_2 + n} \cdot (\sigma_2 + p \cdot \sigma_1)$. Then by the (normal form) SIS assumption, we have $\Delta_2 \cdot \boldsymbol{e}' - \Delta_1 \cdot \boldsymbol{e}'' = \boldsymbol{0}$ with all but negligible probability. This implies $\Delta_2 \cdot \boldsymbol{f}' = \Delta_1 \cdot \boldsymbol{f}''$ and Claim 1 follows. □

[8] As α, α' and α'' are distinct integers in $[-p, p]$ and $q_0 > 2p$, both Δ_1 and Δ_2 are invertible in $\mathbb{Z}_q$.

Claim 2. *Let $\bar{\boldsymbol{r}} = \boldsymbol{z}_0 - \alpha \cdot \bar{\boldsymbol{x}}$, then we have $\boldsymbol{z}_0' - \bar{\boldsymbol{r}} = \alpha' \cdot \bar{\boldsymbol{x}}$ and $\boldsymbol{z}_0'' - \bar{\boldsymbol{r}} = \alpha'' \cdot \bar{\boldsymbol{x}}$ with all but negligible probability.*

Proof. This is because with all but negligible probability, we have

$$\begin{aligned}\boldsymbol{z}_0' - \bar{\boldsymbol{r}} &= \boldsymbol{z}_0' - \boldsymbol{z}_0 + \alpha \cdot \bar{\boldsymbol{x}} \\ &= \Delta_1 \cdot \bar{\boldsymbol{x}} + \alpha \cdot \bar{\boldsymbol{x}} \\ &= \alpha' \cdot \bar{\boldsymbol{x}}\end{aligned}$$

and

$$\begin{aligned}\boldsymbol{z}_0'' - \bar{\boldsymbol{r}} &= \boldsymbol{z}_0'' - \boldsymbol{z}_0 + \alpha \cdot \bar{\boldsymbol{x}} \\ &= \Delta_2 \cdot \bar{\boldsymbol{x}} + \alpha \cdot \bar{\boldsymbol{x}} \\ &= \alpha'' \cdot \bar{\boldsymbol{x}}\end{aligned}$$

where the second equation comes from Claim 1. □

Claim 3. *Under the SIS assumption, $\Delta_1^{-1} \cdot (\boldsymbol{d}' - \boldsymbol{d}) = \Delta_2^{-1} \cdot (\boldsymbol{d}'' - \boldsymbol{d})$ with all but negligible probability.*

Proof. Proof of Claim 3 is similar to the proof of Claim 1, and we omit the details. □

Lemma C.2. *With all but negligible probability, we have $\bar{\boldsymbol{x}}[h] = \bar{\boldsymbol{x}}[i] \cdot \bar{\boldsymbol{x}}[j]$ for all (h, i, j) in $\mathcal{M}$.*

Proof. For any $k \in [1, \ell]$, let (h, i, j) be the k-th element in $\mathcal{M}$, then from Definition of $\bar{\boldsymbol{r}}$ (in Claim 2), we have

$$\begin{aligned}\boldsymbol{d}[k] &= \alpha \cdot \boldsymbol{z}_0[h] - \boldsymbol{z}_0[i] \cdot \boldsymbol{z}_0[j] \\ &= \alpha \cdot (\alpha \cdot \bar{\boldsymbol{x}}[h] + \bar{\boldsymbol{r}}[h]) - (\alpha \cdot \bar{\boldsymbol{x}}[i] + \bar{\boldsymbol{r}}[i]) \cdot (\alpha \cdot \bar{\boldsymbol{x}}[j] + \bar{\boldsymbol{r}}[j]) \\ &= \alpha^2 \cdot (\bar{\boldsymbol{x}}[h] - \bar{\boldsymbol{x}}[i] \cdot \bar{\boldsymbol{x}}[j]) + \alpha \cdot (\bar{\boldsymbol{r}}[h] - \bar{\boldsymbol{x}}[i] \cdot \bar{\boldsymbol{r}}[j] - \bar{\boldsymbol{x}}[j] \cdot \bar{\boldsymbol{r}}[i]) - \bar{\boldsymbol{r}}[i] \cdot \bar{\boldsymbol{r}}[j]\end{aligned}$$

Similar, from Claim 2, we have

$$\begin{aligned}\boldsymbol{d}'[k] =& \alpha'^2 \cdot (\bar{\boldsymbol{x}}[h] - \bar{\boldsymbol{x}}[i] \cdot \bar{\boldsymbol{x}}[j]) + \\ & \alpha' \cdot (\bar{\boldsymbol{r}}[h] - \bar{\boldsymbol{x}}[i] \cdot \bar{\boldsymbol{r}}[j] - \bar{\boldsymbol{x}}[j] \cdot \bar{\boldsymbol{r}}[i]) - \bar{\boldsymbol{r}}[i] \cdot \bar{\boldsymbol{r}}[j]\end{aligned}$$

and

$$\begin{aligned}\boldsymbol{d}''[k] =& \alpha''^2 \cdot (\bar{\boldsymbol{x}}[h] - \bar{\boldsymbol{x}}[i] \cdot \bar{\boldsymbol{x}}[j]) + \\ & \alpha'' \cdot (\bar{\boldsymbol{r}}[h] - \bar{\boldsymbol{x}}[i] \cdot \bar{\boldsymbol{r}}[j] - \bar{\boldsymbol{x}}[j] \cdot \bar{\boldsymbol{r}}[i]) - \bar{\boldsymbol{r}}[i] \cdot \bar{\boldsymbol{r}}[j]\end{aligned}$$

with all but negligible probability. This implies that

$$\begin{aligned}\boldsymbol{d}'[k] - \boldsymbol{d}[k] =& (\alpha'^2 - \alpha^2) \cdot (\bar{\boldsymbol{x}}[h] - \bar{\boldsymbol{x}}[i] \cdot \bar{\boldsymbol{x}}[j]) + \\ & (\alpha' - \alpha) \cdot (\bar{\boldsymbol{r}}[h] - \bar{\boldsymbol{x}}[i] \cdot \bar{\boldsymbol{r}}[j] - \bar{\boldsymbol{x}}[j] \cdot \bar{\boldsymbol{r}}[i])\end{aligned}$$

and

$$\begin{aligned}\boldsymbol{d}''[k]-\boldsymbol{d}[k]=&(\alpha''^2-\alpha^2)\cdot(\bar{\boldsymbol{x}}[h]-\bar{\boldsymbol{x}}[i]\cdot\bar{\boldsymbol{x}}[j])+\\&(\alpha''-\alpha)\cdot(\bar{\boldsymbol{r}}[h]-\bar{\boldsymbol{x}}[i]\cdot\bar{\boldsymbol{r}}[j]-\bar{\boldsymbol{x}}[j]\cdot\bar{\boldsymbol{r}}[i])\end{aligned}$$

Also, from Claim 3, with all but negligible probability, we have $\Delta_1^{-1}\cdot(\boldsymbol{d}'[k]-\boldsymbol{d}[k])=\Delta_2^{-1}\cdot(\boldsymbol{d}''[k]-\boldsymbol{d}[k])$. Thus we have

$$\begin{aligned}&(\alpha'+\alpha)\cdot(\bar{\boldsymbol{x}}[h]-\bar{\boldsymbol{x}}[i]\cdot\bar{\boldsymbol{x}}[j])+(\bar{\boldsymbol{r}}[h]-\bar{\boldsymbol{x}}[i]\cdot\bar{\boldsymbol{r}}[j]-\bar{\boldsymbol{x}}[j]\cdot\bar{\boldsymbol{r}}[i])\\=&(\alpha''+\alpha)\cdot(\bar{\boldsymbol{x}}[h]-\bar{\boldsymbol{x}}[i]\cdot\bar{\boldsymbol{x}}[j])+(\bar{\boldsymbol{r}}[h]-\bar{\boldsymbol{x}}[i]\cdot\bar{\boldsymbol{r}}[j]-\bar{\boldsymbol{x}}[j]\cdot\bar{\boldsymbol{r}}[i])\end{aligned}$$

which implies

$$(\alpha''-\alpha')\cdot(\bar{\boldsymbol{x}}[h]-\bar{\boldsymbol{x}}[i]\cdot\bar{\boldsymbol{x}}[j])=0$$

Since $\alpha''\neq\alpha'$ and $|\alpha''-\alpha'|\leq 2p<q_0$, we have $\bar{\boldsymbol{x}}[h]-\bar{\boldsymbol{x}}[i]\cdot\bar{\boldsymbol{x}}[j]=0$ and that completes the proof of Lemma C.2. □

Now, combining Lemma C.1 and Lemma C.2, proof of knowledge property of P_1 follows.

Honest-Verifier Zero-Knowledge. Let $\boldsymbol{A},\boldsymbol{y},\mathcal{M}$ be a statement, let $\boldsymbol{B}_1,\boldsymbol{B}_2$ be the public parameter. We construct a simulator that can simulate the view of an honest verifier $\hat{V}$ in an interaction with an honest prover, given $\boldsymbol{A},\boldsymbol{y},\mathcal{M}$, $\boldsymbol{B}_1,\boldsymbol{B}_2$ and black-box access to $\hat{V}$.

The simulator first retrieves the challenge α from $\hat{V}$ via feeding it with a commitment of 0 under the auxiliary commitment scheme. Then it:

1. Sample $\boldsymbol{z}_0\xleftarrow{\$}\mathbb{Z}_q^n$ and compute $\boldsymbol{t}=\boldsymbol{A}\cdot\boldsymbol{z}_0-\alpha\cdot\boldsymbol{y}$.
2. Sample $\boldsymbol{z}_1\leftarrow D_{\sigma_2}^{l_2+n+l_1}$ and $\boldsymbol{z}_2\leftarrow D_{\sigma_2}^{l_2+\ell+l_1}$.
3. Sample $\boldsymbol{c}_1\xleftarrow{\$}\mathbb{Z}_q^{l_1+n}$ and $\boldsymbol{c}_3\xleftarrow{\$}\mathbb{Z}_q^{l_1+\ell}$.
4. Compute a ℓ-dimension vector $\boldsymbol{d}$ that for $k\in[1,\ell]$, let (h,i,j) be the k-th element in $\mathcal{M}$, $\boldsymbol{d}[k]=\alpha\cdot\boldsymbol{z}_0[h]-\boldsymbol{z}_0[i]\cdot\boldsymbol{z}_0[j]$.
5. Compute $\boldsymbol{c}_2=\boldsymbol{B}_1\cdot\boldsymbol{z}_1+(\boldsymbol{0}^\intercal\|\boldsymbol{z}_0^\intercal)^\intercal-\alpha\cdot\boldsymbol{c}_1$ and $\boldsymbol{c}_4=\alpha\cdot\boldsymbol{c}_3-\boldsymbol{B}_2\cdot\boldsymbol{z}_2-(\boldsymbol{0}^\intercal\|\boldsymbol{d}^\intercal)^\intercal$.
6. Sample $\rho\xleftarrow{\$}\{0,1\}^\kappa$ and computes $C_{aux}=\mathtt{aCommit}(\boldsymbol{t}\|\boldsymbol{c}_1\|\boldsymbol{c}_2\|\boldsymbol{c}_3\|\boldsymbol{c}_4;\rho)$ and $C'_{aux}=\mathtt{aCommit}(\boldsymbol{0};\rho)$.
7. Finally, with probability $1/M$, output $(C_{aux},\alpha,\boldsymbol{t},\boldsymbol{c}_1,\boldsymbol{c}_2,\boldsymbol{c}_3,\boldsymbol{c}_4,\rho,\boldsymbol{z}_0,\boldsymbol{z}_1,\boldsymbol{z}_2)$ and with probability $1-1/M$, output $(C'_{aux},\alpha,\perp)$.

Next, we argue that the output of the simulator is computationally indistinguishable from $\hat{V}$'s view in an interaction with an honest prover with a valid witness $\boldsymbol{x}$ for $\boldsymbol{A},\boldsymbol{y},\mathcal{M}$. We first define the following auxiliary games.

- Game 0: In this game, a prover runs P_1 honestly with $\hat{V}$.
- Game 1: In this game, the prover retrieves a challenge α from $\hat{V}$ via feeding it with a commitment of 0 under the auxiliary commitment scheme. Then it rewinds $\hat{V}$ to the initial state while fixing its inner randomness and proceeds as follows:

1. Sample $\boldsymbol{r} \xleftarrow{\$} \mathbb{Z}_q^n$ and compute $\boldsymbol{t} = \boldsymbol{A} \cdot \boldsymbol{r}$.
2. Sample $\boldsymbol{s}_1 \leftarrow D_{\sigma_1}^{l_2+n+l_1}$, $\boldsymbol{s}_2 \leftarrow D_{\sigma_2}^{l_2+n+l_1}$, $\boldsymbol{s}_3 \leftarrow D_{\sigma_1}^{l_2+\ell+l_1}$, and $\boldsymbol{s}_4 \leftarrow D_{\sigma_2}^{l_2+\ell+l_1}$.
3. Compute $\boldsymbol{c}_1 = \boldsymbol{B}_1 \cdot \boldsymbol{s}_1 + (\boldsymbol{0}^\intercal \| \boldsymbol{x}^\intercal)^\intercal$ and $\boldsymbol{c}_2 = \boldsymbol{B}_1 \cdot \boldsymbol{s}_2 + (\boldsymbol{0}^\intercal \| \boldsymbol{r}^\intercal)^\intercal$.
4. Compute two ℓ-dimension vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, where for $k \in [1, \ell]$, let (h, i, j) be the k-th element in $\mathcal{M}$, $\boldsymbol{a}[k] = \boldsymbol{r}[h] - \boldsymbol{r}[i] \cdot \boldsymbol{x}[j] - \boldsymbol{r}[j] \cdot \boldsymbol{x}[i]$ and $\boldsymbol{b}[k] = \boldsymbol{r}[i] \cdot \boldsymbol{r}[j]$.
5. Compute $\boldsymbol{c}_3 = \boldsymbol{B}_2 \cdot \boldsymbol{s}_3 + (\boldsymbol{0}^\intercal \| \boldsymbol{a}^\intercal)^\intercal$ and $\boldsymbol{c}_4 = \boldsymbol{B}_2 \cdot \boldsymbol{s}_4 + (\boldsymbol{0}^\intercal \| \boldsymbol{b}^\intercal)^\intercal$.
6. Compute $\boldsymbol{z}_0 = \alpha \cdot \boldsymbol{x} + \boldsymbol{r}$, $\boldsymbol{z}_1 = \alpha \cdot \boldsymbol{s}_1 + \boldsymbol{s}_2$, and $\boldsymbol{z}_2 = \alpha \cdot \boldsymbol{s}_3 - \boldsymbol{s}_4$.
7. Set the binary variable *abort* to be 1 with probability $1 - \mathfrak{p}((\alpha \cdot \boldsymbol{s}_1^\intercal \| \alpha \cdot \boldsymbol{s}_3^\intercal)^\intercal, (\boldsymbol{z}_1^\intercal \| \boldsymbol{z}_2^\intercal)^\intercal)$.
8. Sample $\rho \xleftarrow{\$} \{0,1\}^\kappa$ and compute $C_{aux} = \mathtt{aCommit}(\boldsymbol{t} \| \boldsymbol{c}_1 \| \boldsymbol{c}_2 \| \boldsymbol{c}_3 \| \boldsymbol{c}_4; \rho)$.
9. Send C_{aux} to $\hat{V}$.
10. On receiving a new challenge α', abort if $abort = 1$ and send $(\boldsymbol{t}, \boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{c}_3, \boldsymbol{c}_4, \rho, \boldsymbol{z}_0, \boldsymbol{z}_1, \boldsymbol{z}_2)$ otherwise.

- Game 2: This is identical to Game 1 except that the prover changes the way to generate $\boldsymbol{c}_2$, $\boldsymbol{c}_4$ and $\boldsymbol{t}$. In particular, it first computes a ℓ-dimension vector $\boldsymbol{d}$ that for $k \in [1, \ell]$, let (h, i, j) be the k-th element in $\mathcal{M}$, $\boldsymbol{d}[k] = \alpha \cdot \boldsymbol{z}_0[h] - \boldsymbol{z}_0[i] \cdot \boldsymbol{z}_0[j]$. Then it computes $\boldsymbol{c}_2 = \boldsymbol{B}_1 \cdot \boldsymbol{z}_1 + (\boldsymbol{0}^\intercal \| \boldsymbol{z}_0^\intercal)^\intercal - \alpha \cdot \boldsymbol{c}_1$, $\boldsymbol{c}_4 = \alpha \cdot \boldsymbol{c}_3 - \boldsymbol{B}_2 \cdot \boldsymbol{z}_2 - (\boldsymbol{0}^\intercal \| \boldsymbol{d}^\intercal)^\intercal$ and $\boldsymbol{t} = \boldsymbol{A} \cdot \boldsymbol{z}_0 - \alpha \cdot \boldsymbol{y}$.
- Game 3: This is identical to Game 2 except that the prover changes the way to compute the auxiliary commitment. In particular, $C_{aux} = \mathtt{aCommit}(\boldsymbol{0}; \rho)$ if $abort = 1$ and $C_{aux} = \mathtt{aCommit}(\boldsymbol{t} \| \boldsymbol{c}_1 \| \boldsymbol{c}_2 \| \boldsymbol{c}_3 \| \boldsymbol{c}_4; \rho)$ (i.e., remains identical to Game 1) otherwise.
- Game 4: This is identical to Game 3 except that the prover changes the way to computes $\boldsymbol{z}_1$, $\boldsymbol{z}_2$ and *abort*. In particular, it samples $\boldsymbol{z}_1 \leftarrow D_{\sigma_2}^{l_2+n+l_1}$ and $\boldsymbol{z}_2 \leftarrow D_{\sigma_2}^{l_2+\ell+l_1}$, and sets *abort* to be 1 with probability $1 - 1/M$.
- Game 5: This is identical to Game 4 except that the prover samples $\boldsymbol{c}_1 \xleftarrow{\$} \mathbb{Z}_q^{l_1+n}$.
- Game 6: This is identical to Game 5 except that the prover samples $\boldsymbol{c}_3 \xleftarrow{\$} \mathbb{Z}_q^{l_1+\ell}$.
- Game 7: This is identical to Game 6 except that the prover samples $\boldsymbol{z}_0 \xleftarrow{\$} \mathbb{Z}_q^n$.

Next, we prove the indistinguishability of each consecutive pair of games.

Lemma C.3. *$\hat{V}$'s view in Game 0 and Game 1 are identical.*

Proof. As $\hat{V}$ is an honest verifier, the challenge it outputs is determined by its inner randomness. Thus, we always have $\alpha = \alpha'$. Therefore, all variables, including $\hat{V}$'s views, are identically computed in Game 0 and Game 1. □

Lemma C.4. *$\hat{V}$'s view in Game 1 and Game 2 are identical.*

Proof. Vectors $\boldsymbol{c}_2$, $\boldsymbol{c}_4$ and $\boldsymbol{t}$ are identically computed in Game 1 and Game 2 (as shown in the proof of completeness for P_1), thus, the two games are identical. □

Lemma C.5. *$\hat{V}$'s view in Game 2 and Game 3 are indistinguishable.*

Proof. Lemma C.5 comes from the hiding property of the auxiliary commitment via a direct reduction, and we just omit the details here. □

Lemma C.6. *$\hat{V}$'s view in Game 3 and Game 4 are statistically indistinguishable.*

Proof. Lemma C.6 follows the rejection sampling lemma (Lemma 2.2) directly, and we just omit the details here. □

Lemma C.7. *Under the LWE assumption, $\hat{V}$'s view in Game 4 and Game 5 are computationally indistinguishable.*

Proof. This comes from the hiding property of the homomorphic commitment, which has been proved in [10]. Here we include the detailed proof for completeness. We prove Lemma C.7 by showing that if there exists a distinguisher $\mathcal{A}$ that can distinguish $\hat{V}'s$ view in Game 4 and Game 5, we can construct an algorithm $\mathcal{B}$ solving the (normal form) LWE problem, which works as follows.

On input an instance $(\bar{\boldsymbol{A}}, \bar{\boldsymbol{b}}) \in \mathbb{Z}_q^{(l_1+n)\times l_2} \times \mathbb{Z}_q^{l_1+n}$, where $\bar{\boldsymbol{A}}$ is a random matrix in $\mathbb{Z}_q^{(l_1+n)\times l_2}$, and $\bar{\boldsymbol{b}}$ is either a random vector in $\mathbb{Z}_q^{l_1+n}$ or $\bar{\boldsymbol{b}} = \bar{\boldsymbol{A}} \cdot \bar{\boldsymbol{s}} + \bar{\boldsymbol{e}}$ for $\bar{\boldsymbol{s}} \leftarrow D_{\sigma_1}^{l_2}$ and $\bar{\boldsymbol{e}} \leftarrow D_{\sigma_1}^{l_1+n}$, the algorithm $\mathcal{B}$ first samples a random matrix $\boldsymbol{R} \overset{\$}{\leftarrow} \mathbb{Z}_q^{l_1 \times n}$. Then it proceeds identically to the prover in Game 4 (or Game 5) except that it does not sample $\boldsymbol{s}_1$ and sets

$$\boldsymbol{B}_1 = \begin{pmatrix} \boldsymbol{I}_{l_1} & \boldsymbol{R} \\ 0^{n\times l_1} & \boldsymbol{I}_n \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{I}_{l_1} & 0^{l_1\times n} & \bar{\boldsymbol{A}}_{\boldsymbol{U}} \\ 0^{n\times l_1} & \boldsymbol{I}_n & \bar{\boldsymbol{A}}_{\boldsymbol{L}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{I}_{l_1} & \boldsymbol{R} & \bar{\boldsymbol{A}}_{\boldsymbol{U}} + \boldsymbol{R}\cdot\bar{\boldsymbol{A}}_{\boldsymbol{L}} \\ 0^{n\times l_1} & \boldsymbol{I}_n & \bar{\boldsymbol{A}}_{\boldsymbol{L}} \end{pmatrix}$$

and

$$\boldsymbol{c}_1 = \begin{pmatrix} \boldsymbol{I}_{l_1} & \boldsymbol{R} \\ 0^{n\times l_1} & \boldsymbol{I}_n \end{pmatrix} \cdot \begin{pmatrix} \bar{\boldsymbol{b}}_{\boldsymbol{U}} \\ \bar{\boldsymbol{b}}_{\boldsymbol{L}} \end{pmatrix} + \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{x} \end{pmatrix} = \begin{pmatrix} \bar{\boldsymbol{b}}_{\boldsymbol{U}} + \boldsymbol{R}\cdot\bar{\boldsymbol{b}}_{\boldsymbol{L}} \\ \bar{\boldsymbol{b}}_{\boldsymbol{L}} \end{pmatrix} + \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{x} \end{pmatrix}$$

where

$$\bar{\boldsymbol{A}} = \begin{pmatrix} \bar{\boldsymbol{A}}_{\boldsymbol{U}} \\ \bar{\boldsymbol{A}}_{\boldsymbol{L}} \end{pmatrix}, \quad \bar{\boldsymbol{b}} = \begin{pmatrix} \bar{\boldsymbol{b}}_{\boldsymbol{U}} \\ \bar{\boldsymbol{b}}_{\boldsymbol{L}} \end{pmatrix}$$

Finally, $\mathcal{B}$ feeds $\mathcal{A}$ with $\hat{V}$'s view in its simulated interaction and outputs what $\mathcal{A}$ outputs.

It is easy to see that this $\boldsymbol{B}_1$ is identically distributed as that in Game 4 and Game 5. Also, if $\bar{\boldsymbol{b}}$ is a random vector, $\boldsymbol{c}_1$ is also a random vector in $\mathbb{Z}_q^{l_1+n}$. Thus $\mathcal{B}$ perfectly simulates Game 5 if $\bar{\boldsymbol{b}}$ is a random vector.

In contrast, if $\bar{\boldsymbol{b}} = \bar{\boldsymbol{A}} \cdot \bar{\boldsymbol{s}} + \bar{\boldsymbol{e}}$ for $\bar{\boldsymbol{s}} \leftarrow D_{\sigma_1}^{l_2}$ and $\bar{\boldsymbol{e}} \leftarrow D_{\sigma_1}^{l_1+n}$, we have

$$\begin{aligned}
\boldsymbol{c}_1 &= \begin{pmatrix} \boldsymbol{I}_{l_1} & \boldsymbol{R} \\ 0^{n\times l_1} & \boldsymbol{I}_n \end{pmatrix} \cdot \bar{\boldsymbol{b}} + \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{x} \end{pmatrix} \\
&= \begin{pmatrix} \boldsymbol{I}_{l_1} & \boldsymbol{R} \\ 0^{n\times l_1} & \boldsymbol{I}_n \end{pmatrix} \cdot \left(\boldsymbol{I}_{l_1+n} \; \bar{\boldsymbol{A}} \right) \cdot \begin{pmatrix} \bar{\boldsymbol{e}} \\ \bar{\boldsymbol{s}} \end{pmatrix} + \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{x} \end{pmatrix} \\
&= \boldsymbol{B}_1 \cdot \begin{pmatrix} \bar{\boldsymbol{e}} \\ \bar{\boldsymbol{s}} \end{pmatrix} + \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{x} \end{pmatrix}
\end{aligned}$$

That is, $\boldsymbol{c}_1$ is a valid commitment of $\boldsymbol{x}$ with randomness $(\bar{\boldsymbol{e}}^\intercal \| \bar{\boldsymbol{s}}^\intercal)^\intercal$. Thus, $\mathcal{B}$ can perfectly simulate Game 4 via implicitly setting $\boldsymbol{s}_1 = (\bar{\boldsymbol{e}}^\intercal \| \bar{\boldsymbol{s}}^\intercal)^\intercal$ in this case.

In summary, if $\mathcal{A}$ could distinguish Game 4 and Game 5, $\mathcal{B}$ can distinguish if the instance is a LWE instance or a random one. That completes the proof. $\square$

Lemma C.8. *Under the LWE assumption, $\hat{V}$'s view in Game 5 and Game 6 are computationally indistinguishable.*

Proof. Lemma C.8 also comes from the binding property of the underlying homomorphic commitment and we omit the details here. $\square$

Lemma C.9. *$\hat{V}$'s view in Game 6 and Game 7 are identical.*

Proof. In Game 6, $\boldsymbol{z}_0 = \alpha \cdot \boldsymbol{x} + \boldsymbol{r}$, where $\boldsymbol{r}$ is sampled uniformly at random from $\mathbb{Z}_q^n$ and is not used anywhere else. Thus, in $\hat{V}'s$ view, $\boldsymbol{z}_0$ is also a uniform vecotr in $\mathbb{Z}_q^n$ and that completes the proof. $\square$

It can be easily verified that $\hat{V}$'s view in Game 7 is identical to the output of the simulator and that completes the proof of honest-verifier zero-knowledge for P_1. $\square$

D Ring Signature

D.1 The Definition

In this section, we recall the standard syntax and security requirement of a ring signature scheme, which is defined in [12].

A ring signature scheme consists of four algorithms:

- **Setup.** On input a security parameter 1^λ, the setup algorithm outputs the public parameter *param* for the scheme.
- **KeyGen.** The key generation algorithm outputs a secret key/public key pair (sk, pk).
- **Sign.** On input a message $\mathfrak{m}$, a polynomial-size set R of public keys and a valid public key/secret pair (pk, sk) that $pk \in \mathsf{R}$, the signing algorithm outputs a signature σ.

- **Verify.** On input a message $\mathtt{m}$, a polynomial-size set R of public keys and a signature σ, the verification algorithm outputs a bit indicating whether the signature is acceptable.

Next, we describe security requirements for ring signature schemes. We only give an informal description for each security requirement and refer the readers to previous works, e.g., [12, 54], for the formal definitions.

- **Correctness.** The correctness requires that an honest user who has a valid key pair (pk, sk) is able to generate an acceptable signature on behalf of a set R if $pk \in \mathsf{R}$.
- **Unforgeability.** The unforgeability requires that no user can sign on behalf of a set he does not belong to.
- **Anonymity.** The anonymity requires that no one could locate the real signer given a signature that is signed on behalf of a set of users.

D.2 The Construction

Overview. We construct the ring signature scheme following the generic framework presented in [30]. More precisely, the secret key/public key pair of a user is an input/output pair of a one-way function. To sign on behalf of a set of users, the signer first generates an accumulator that accumulates all public keys of users in this set using a cryptographic accumulator scheme, then he generates an NIZK argument proving that he has a valid secret key/public key pair where the public key is properly accumulated in the accumulator. The signature is the proof (here, the message is embedded in the proof). To verify the signature, the verifier first generates the accumulator for the claimed user set, then she checks if the proof is valid.

Next, we give a detailed description of our ring signature scheme, which is exactly the ring signature scheme constructed in [54], except that we replace the underlying zero-knowledge arguments with the one we constructed in Sec. 4.4.

The Construction. Let λ be the security parameter. Let n be a postive integer that is polynomial in λ. Let q be a large enough prime number and let $k = \lceil \log q \rceil$, $m = nk$. Let $N = 2^L$ be the size of supported user set. Let $\mathsf{Acc} = (\mathsf{Acc}.\mathtt{Acc}, \mathsf{Acc}.\mathtt{Witness}, \mathsf{Acc}.\mathtt{Verify})$ be the accumulator scheme recalled in Sec. A.4. The ring signature scheme RS works as follows:

- **Setup.** The setup algorithm samples matrices $\boldsymbol{B}_1, \boldsymbol{B}_2 \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$ and outputs the public parameter $pp = (\boldsymbol{B}_1, \boldsymbol{B}_2)$.
- **KeyGen.** The key generation algorithm samples $\boldsymbol{s} \xleftarrow{\$} \{0,1\}^{2m}$ and computes $\boldsymbol{t} = \mathtt{bin}((\boldsymbol{B}_1 \| \boldsymbol{B}_2) \cdot \boldsymbol{s})$. Then it outputs $sk = \boldsymbol{s}$ and $pk = \boldsymbol{t}$.
- **Sign.** On input a set R of at most N public keys, a secret key/public key pair $(\boldsymbol{s}, \boldsymbol{t})$ that $\boldsymbol{t} \in \mathsf{R}$ and a message $\mathtt{m}$, the sign algorithm first computes $\boldsymbol{u} \leftarrow \mathsf{Acc}.\mathtt{Acc}_{\boldsymbol{B}_1,\boldsymbol{B}_2}(\mathsf{R})$. Then it generates the witness $(\boldsymbol{m}, \{\boldsymbol{v}_i\}_{i\in[1,L]}, \{\boldsymbol{w}_i\}_{i\in[1,L]}) \leftarrow \mathsf{Acc}.\mathtt{Witness}_{\boldsymbol{B}_1,\boldsymbol{B}_2}(\mathsf{R}, \boldsymbol{t})$. Next, the sign algorithm generates

the proof:

$$\pi = SPK\{(\boldsymbol{B}_1, \boldsymbol{B}_2, \boldsymbol{u}), (\boldsymbol{m}, \{\boldsymbol{v}_i\}_{i\in[1,L]}, \{\boldsymbol{w}_i\}_{i\in[1,L]}, \boldsymbol{s}, \boldsymbol{t}) : \\ \mathsf{Acc}\,.\,\mathtt{Verify}_{\boldsymbol{B}_1,\boldsymbol{B}_2}(\boldsymbol{u}, \boldsymbol{t}, (\boldsymbol{m}, \{\boldsymbol{v}_i\}_{i\in[1,L]}, \{\boldsymbol{w}_i\}_{i\in[1,L]})) = 1 \wedge \\ \boldsymbol{H} \cdot \boldsymbol{t} = (\boldsymbol{B}_1 \| \boldsymbol{B}_2) \cdot \boldsymbol{s}\}[\mathfrak{m}]$$

, where $\boldsymbol{H} = \boldsymbol{I}_n \otimes (1 \quad 2 \quad 4 \ldots 2^{k-1})$. The signature is just the proof π .

- **Verify.** On input a set R and a proof π , the verifier first computes $\boldsymbol{u} \leftarrow \mathsf{Acc}\,.\,\mathtt{Acc}_{\boldsymbol{B}_1,\boldsymbol{B}_2}(\mathsf{R})$. Then it outputs 1 if π is valid and 0 otherwise.

Theorem D.1. *Assuming the worst-case hardness of GapSVP$_\gamma$ (or SIVP$_\gamma$) for some polynomial γ , RS is a secure ring signature scheme in the random oracle model.*

Proof of Theorem D.1 follows directly from the security proof for the ring signature scheme proposed in [54], and we omit its details in this work.

D.3 The Efficiency

We focus on the communication cost, namely, the signature size, of the ring signature scheme. As the signature is just a NIZK proof generated by our main protocol, we estimate the signature size via analyzing the size of the proof.

The proved statement includes two parts. The main part argues knowledge of an element in an accumulator. As analyzed in Sec. 4.4, this part can be reduced to an instance of the relation $\mathcal{R}^*$, where the length of its witness is $2L + 4nL + 2nkL$ and the size of $\mathcal{M}$ is $L + 2nL + 2nkL$. The scond part argues a linear euqation with binary solution. After reducing it to an instance of $\mathcal{R}^*$, the length of the witness is $3nk$ and the size of $\mathcal{M}$ is also $3nk$. So, after combing these two statements, the length of the witness will be

$$\mathfrak{n} = 2L + 4nL + 2nkL + 2nk$$

and the size of $\mathcal{M}$ will be

$$\mathfrak{l} = L + 2nL + 2nkL + 2nk$$

(the overlapped part, namely, $\boldsymbol{t}$, should be counted only once).

Now, let $\hat{l}_1, \hat{l}_2$, $\hat{\sigma}_1$, $\hat{\sigma}_2$, $\hat{p}$, $\hat{\kappa}$, $\hat{N}$ be parameters used in the main protocol. Then the proof contains

$$\|\pi\| = (\log(2\hat{p} + 1) + \hat{\kappa} + (3\hat{l}_1 + 2\hat{l}_2 + 2\mathfrak{n} + 2\mathfrak{l}) \cdot k) \cdot \hat{N} + (\hat{l}_1 + \mathfrak{n}) \cdot k \qquad (5)$$

bits.

Choosing the Parameters. Next, we estimate the concrete size of the signature via setting concrete values of parameters involved in Equation (5). Note that security of the ring signature scheme relies on the following 4 assumptions:

$$\begin{cases} SIS_{\hat{l}_1,q,\beta_1} \\ SIS_{\hat{l}_1,q,\beta_2} \\ LWE_{\hat{l}_2,q,\alpha} \\ SIS_{n,q,\beta_3} \end{cases}$$

where $\beta_1 = 16\hat{p} \cdot \sqrt{\hat{l}_1 + \hat{l}_2 + \mathfrak{n}} \cdot (\hat{\sigma}_2 + \hat{p} \cdot \hat{\sigma}_1)$, $\beta_2 = 16\hat{p} \cdot \sqrt{\hat{l}_1 + \hat{l}_2 + \mathfrak{l}} \cdot (\hat{\sigma}_2 + \hat{p} \cdot \hat{\sigma}_1)$, $\beta_3 = \sqrt{2nk}$ and $\alpha = \frac{\sqrt{2\pi} \cdot \hat{\sigma}_1}{q}$. Thus, we should guaratee that the above assumptions are valid while picking parameters. As discussed in Sec. B.1, this can be achieved by setting the parameters in a way that a small root-Hermite factor is required to break the assumptions.

More concretely, we set these parameters as per Table 4. Our parameters are set to achieve 80-bit security, 100-bit security and 128-bit security respectively in a setting that we fix $L = 10$ (i.e., our ring signature allows the user to sign on behalf of 2^{10} users) and $\hat{\kappa} = 128$. Besides, to better compare the efficiency of our ring signature scheme and that in [33], we also set the parameters according to their criterion in the last column.

Table 4: Concrete Parameters for Our Ring Signature.

Soundness Error	2^{-80}	2^{-100}	2^{-128}	2^{-100}
δ_0	1.0048	1.0042	1.0035	1.007
$\hat{p}$	2^{40}	2^{50}	2^{64}	2^{50}
$\hat{l}_1$	3700	5100	7400	3000
$\hat{l}_2$	4000	5400	7900	3300
$\hat{N}$	2	2	2	2
$\lceil \log q \rceil$	120	140	170	140
n	10	10	11	5
Signature Size	4.24MB	5.9MB	9.7MB	3.05MB

E Group Signature

E.1 The Definition

In this section, we recall the standard syntax and security requirement of a (static) group signature scheme, which is defined in [11].

A group signature scheme consists of four algorithms:

- **Setup.** On input $(1^\lambda, 1^N)$, where λ is the security parameter and N is the number of group users, the setup algorithm outputs a tuple $(gpk, gmsk, \boldsymbol{gsk})$, where gpk is the group public key, $gmsk$ is the group manager's secret key, and $\boldsymbol{gsk} = (\boldsymbol{gsk}[0], \dots, \boldsymbol{gsk}[N-1])$ is an N-dimension vector, where for $i \in [0, N-1]$, $\boldsymbol{gsk}[i]$ is the secret key for user i.
- **Sign.** On input a message $\mathfrak{m}$, a group public key gpk and a user secret key $\boldsymbol{gsk}[i]$, the signing algorithm outputs a signature σ.
- **Verify.** On input a message $\mathfrak{m}$, a group public key gpk and a signature σ, the verification algorithm outputs a bit indicating whether the signature is acceptable.
- **Open.** On input a group public key gpk, a group manager's secret key $gmsk$, a message $\mathfrak{m}$ and a signature σ, the open algorithm outputs an identity $i \in [0, N-1]$ or $\perp$ if the algorithm fails.

Next, we describe security requirements for group signature schemes. We only give an informal description for each security requirement and refer the readers to previous works, e.g., [11, 54], for the formal definitions.

- **Correctness.** The correctness requires that 1) a valid user i is able to sign an acceptable signature and 2) the group manager can open an honestly signed signature correctly.
- **Traceability.** The traceability requires that for any signature that is acceptable by the verification algorithm, the group manager is able to find the real signer. This property holds even when the signatrue is generated by a set of colluded users and in this case the group manager is able to find a member of the coalition.
- **Anonymity.** The anonymity requires that no one except the group manager could locate the real signer given a signature.

E.2 The Construction

Overview. In an accumulator-based group signature [78], a group of users is fiexed in the beginning and the group manager generates an accumulator of public keys of all users in this group. Then the accumulator is used in all sign and verifiaction procedures during the life time of the scheme. To sign on behalf of the group, one needs to generate an NIZK argument proving that he has a valid secret key/public key pair where the public key is properly accumulated in the accumulator. Besides, he also encrypts his identity (under the group manager's public key) using a CCA-2 secure encryption scheme and proves that his identity is properly encrypted. The signature includes the ciphertext and the proofs (here, the message is embedded in the proof). To verify the signature, the verifier just checks the validity of the proofs. When the group manager would like to open a signature, it decrypts the ciphertext attached in the signature and can obtain the identity of the signer.

Next, we give a detailed description of our group signature scheme, which is exactly the group signature scheme constructed in [54], except that we replace the underlying zero-knowledge arguments with the one we constructed in Sec. 4.2 and that in Sec. 4.4.

The Construction. Let λ be the security parameter. Let n, n_1, n_2 be postive integers that are polynomial in λ. Let q be a large enough prime number and let $k = \lceil \log q \rceil$, $m = nk$. Let $N = 2^L$ be the size of supported user set. Let $\mathsf{Acc} = (\mathsf{Acc}.\mathtt{Acc}, \mathsf{Acc}.\mathtt{Witness}, \mathsf{Acc}.\mathtt{Verify})$ be the accumulator scheme recalled in Sec. A.4. Let $\mathsf{PKE} = (\mathsf{PKE}.\mathtt{KeyGen}, \mathsf{PKE}.\mathtt{Enc}, \mathsf{PKE}.\mathtt{Dec})$ be the encryption scheme recalled in Sec. A.2 (here, we use the variants with binary secrets and errors). The group signature scheme GS works as follows:

- **Setup.** The setup algorithm first samples matrices $\boldsymbol{B}_1, \boldsymbol{B}_2 \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$, which serve as the public key of the underlying accumulator scheme. Then it generates the global accumulator and secret keys for users as follows:
  1. For $i \in [0, N-1]$, sample $\boldsymbol{s}_i \xleftarrow{\$} \{0,1\}^{2m}$ and compute $\boldsymbol{t}_i = \mathtt{bin}((\boldsymbol{B}_1 \| \boldsymbol{B}_2)\cdot \boldsymbol{s}_i)$.
  2. Compute the global accumulator $\boldsymbol{u} \leftarrow \mathsf{Acc}.\mathtt{Acc}_{\boldsymbol{B}_1,\boldsymbol{B}_2}(\mathsf{R})$, where $\mathsf{R} = \{\boldsymbol{t}_i\}_{i\in[0,N-1]}$.
  3. For $i \in [0, N-1]$, compute the witness $(\boldsymbol{m}_i, \{\boldsymbol{v}_{i,j}\}_{j\in[1,L]}, \{\boldsymbol{w}_{i,j}\}_{j\in[1,L]}) \leftarrow \mathsf{Acc}.\mathtt{Witness}_{\boldsymbol{B}_1,\boldsymbol{B}_2}(\mathsf{R}, \boldsymbol{t}_i)$ and set the user key for the i-th user as $\boldsymbol{gsk}[i] = (\boldsymbol{s}_i, \boldsymbol{t}_i, \boldsymbol{m}_i, \{\boldsymbol{v}_{i,j}\}_{j\in[1,L]}, \{\boldsymbol{w}_{i,j}\}_{j\in[1,L]})$.

  Next, the algorithm generates the public keys of the used encryption schemes. More precisely, it first samples $\boldsymbol{D}_0 \xleftarrow{\$} \mathbb{Z}_q^{n_1\times n_2}$, $\boldsymbol{S}_1 \xleftarrow{\$} \{0,1\}^{L\times n_1}$, $\boldsymbol{S}_2 \xleftarrow{\$} \{0,1\}^{L\times n_1}$, $\boldsymbol{E}_1 \xleftarrow{\$} \{0,1\}^{L\times n_2}$, $\boldsymbol{E}_2 \xleftarrow{\$} \{0,1\}^{L\times n_2}$. Then it computes

$$\boldsymbol{D}_1 = \boldsymbol{S}_1 \cdot \boldsymbol{D}_0 + \boldsymbol{E}_1$$
$$\boldsymbol{D}_2 = \boldsymbol{S}_2 \cdot \boldsymbol{D}_0 + \boldsymbol{E}_2$$

  Finally, the setup algorithm outputs the group public key $gpk = (\boldsymbol{B}_1, \boldsymbol{B}_2, \boldsymbol{u}, \boldsymbol{D}_0, \boldsymbol{D}_1, \boldsymbol{D}_2)$, the group manager's secret key $gmsk = \boldsymbol{S}_1$, and the users' secret keys $\boldsymbol{gsk} = (\boldsymbol{gsk}[0], \ldots, \boldsymbol{gsk}[N-1])$.
- **Sign.** The input of the sign algorithm includes the group public key $gpk = (\boldsymbol{B}_1, \boldsymbol{B}_2, \boldsymbol{u}, \boldsymbol{D}_0, \boldsymbol{D}_1, \boldsymbol{D}_2)$, a user secret key $\boldsymbol{gsk}[i] = (\boldsymbol{s}_i, \boldsymbol{t}_i, \boldsymbol{m}_i, \{\boldsymbol{v}_{i,j}\}_{j\in[1,L]}, \{\boldsymbol{w}_{i,j}\}_{j\in[1,L]})$ and a message $\mathfrak{m}$.
  First, the algorithm encrypts the user's identity $\boldsymbol{m}_i$ (recall that $\boldsymbol{m}_i = \mathtt{bin}(i)$) using $\boldsymbol{D}_0, \boldsymbol{D}_1$ and $\boldsymbol{D}_0, \boldsymbol{D}_2$. More precisely, it first samples $\boldsymbol{r}_1 \xleftarrow{\$} \{0,1\}^{n_2}$, $\boldsymbol{e}_{1,1} \xleftarrow{\$} \{0,1\}^{n_1}$, $\boldsymbol{e}_{1,2} \xleftarrow{\$} \{0,1\}^{L}$ and $\boldsymbol{r}_2 \xleftarrow{\$} \{0,1\}^{n_2}$, $\boldsymbol{e}_{2,1} \xleftarrow{\$} \{0,1\}^{n_1}$, $\boldsymbol{e}_{2,2} \xleftarrow{\$} \{0,1\}^{L}$. Then it computes

$$\begin{cases} \boldsymbol{c}_{1,1} = \boldsymbol{D}_0 \cdot \boldsymbol{r}_1 + \boldsymbol{e}_{1,1} \\ \boldsymbol{c}_{1,2} = \boldsymbol{D}_1 \cdot \boldsymbol{r}_1 + \boldsymbol{e}_{1,2} + \lfloor \frac{q}{2} \rceil \cdot \boldsymbol{m}_i \\ \boldsymbol{c}_{2,1} = \boldsymbol{D}_0 \cdot \boldsymbol{r}_2 + \boldsymbol{e}_{2,1} \\ \boldsymbol{c}_{2,2} = \boldsymbol{D}_1 \cdot \boldsymbol{r}_2 + \boldsymbol{e}_{2,2} + \lfloor \frac{q}{2} \rceil \cdot \boldsymbol{m}_i \end{cases}$$

Next, the sign algorithm generates the proof:

$$\begin{aligned}\pi = SPK\{&(\boldsymbol{B}_1, \boldsymbol{B}_2, \boldsymbol{u}, \boldsymbol{D}_0, \boldsymbol{D}_1, \boldsymbol{D}_2, \boldsymbol{c}_{1,1}, \boldsymbol{c}_{1,2}, \boldsymbol{c}_{2,1}, \boldsymbol{c}_{2,2}),\\ &(\{\boldsymbol{v}_{i,j}\}_{j\in[1,L]}, \{\boldsymbol{w}_{i,j}\}_{j\in[1,L]}, \boldsymbol{m}_i, \boldsymbol{s}_i, \boldsymbol{t}_i, \boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{e}_{1,1}, \boldsymbol{e}_{1,2}, \boldsymbol{e}_{2,1}, \boldsymbol{e}_{2,2}) :\\ &\quad \mathsf{Acc}.\mathtt{Verify}_{\boldsymbol{B}_1,\boldsymbol{B}_2}(\boldsymbol{u}, \boldsymbol{t}, (\boldsymbol{m}_i, \{\boldsymbol{v}_i\}_{i\in[1,L]}, \{\boldsymbol{w}_i\}_{i\in[1,L]})) = 1\wedge\\ &\quad (\boldsymbol{c}_{1,1}, \boldsymbol{c}_{1,2}) = \mathsf{PKE}.\mathtt{Enc}((\boldsymbol{D}_0, \boldsymbol{D}_1), \boldsymbol{m}_i; (\boldsymbol{r}_1, \boldsymbol{e}_{1,1}, \boldsymbol{e}_{1,2}))\wedge\\ &\quad (\boldsymbol{c}_{2,1}, \boldsymbol{c}_{2,2}) = \mathsf{PKE}.\mathtt{Enc}((\boldsymbol{D}_0, \boldsymbol{D}_2), \boldsymbol{m}_i; (\boldsymbol{r}_2, \boldsymbol{e}_{2,1}, \boldsymbol{e}_{2,2}))\wedge\\ &\qquad\qquad\qquad\qquad \boldsymbol{H}\cdot \boldsymbol{t} = (\boldsymbol{B}_1\|\boldsymbol{B}_2)\cdot \boldsymbol{s}\}[\mathfrak{m}]\end{aligned}$$

, where $\boldsymbol{H} = \boldsymbol{I}_n \otimes (1 \quad 2 \quad 4 \ldots 2^{k-1})$. The signature $\sigma = (\pi, \boldsymbol{c}_{1,1}, \boldsymbol{c}_{1,2}, \boldsymbol{c}_{2,1}, \boldsymbol{c}_{2,2})$.

- **Verify.** On input the group public key $gpk = (\boldsymbol{B}_1, \boldsymbol{B}_2, \boldsymbol{u}, \boldsymbol{D}_0, \boldsymbol{D}_1, \boldsymbol{D}_2)$, a signature $\sigma = (\pi, \boldsymbol{c}_{1,1}, \boldsymbol{c}_{1,2}, \boldsymbol{c}_{2,1}, \boldsymbol{c}_{2,2})$ and a message $\mathfrak{m}$, the verifiaction algorithm checks the proof π and outputs 1 iff π is valid.
- **Open.** On input the group public key $gpk = (\boldsymbol{B}_1, \boldsymbol{B}_2, \boldsymbol{u}, \boldsymbol{D}_0, \boldsymbol{D}_1, \boldsymbol{D}_2)$, the group manager's secret key $gmsk = \boldsymbol{S}_1$, a signature $\sigma = (\pi, \boldsymbol{c}_{1,1}, \boldsymbol{c}_{1,2}, \boldsymbol{c}_{2,1}, \boldsymbol{c}_{2,2})$ and a message $\mathfrak{m}$, the open algorithm decrypts $\boldsymbol{m} = \mathsf{PKE}.\mathtt{Dec}(\boldsymbol{S}_1, (\boldsymbol{c}_{1,1}, \boldsymbol{c}_{1,2}))$. Then, it outputs the identity $i = \sum_{j=1}^{L} 2^{j-1}\cdot \boldsymbol{m}[j]$.

Theorem E.1. *Assuming the worst-case hardness of GapSVP$_\gamma$ (or SIVP$_\gamma$) for some polynomial γ and the hardness of LWE assumption with binary secrets and binary errors for some specific parameters,* GS *is a secure group signature scheme in the random oracle model.*

Proof of Theorem E.1 follows directly from the security proof for the group signature scheme proposed in [54], and we omit its details in this work.

E.3 The Efficiency

We focus on the communication cost, namely, the signature size, of the group signature scheme. The signature consists of a NIZK proof generated by our main protocol and two ciphertexts.

We first analyze the size of the proof. The proved statement includes three parts. The first part argues knowledge of an element in an accumulator. As analyzed in Sec. 4.4, this part can be reduced to an instance of the relation $\mathcal{R}^*$, where the length of its witness is $2L + 4nL + 2nkL$ and the size of $\mathcal{M}$ is $L + 2nL + 2nkL$. The second part argues knowledge of the (same) plaintext encrypted in the two ciphertexts. As analyzed in Sec. 4.2, this part can be reduced to an instance of the relation $\mathcal{R}^*$, where the length of its witness and the size of $\mathcal{M}$ are both $2n_1+2n_2+3L$. The last part argues a linear equation with binary solution. After reducing it to an instance of $\mathcal{R}^*$, the length of the witness is $3nk$ and the size of $\mathcal{M}$ is also $3nk$. So, after combing these statements, the length of the witness will be

$$\mathfrak{n} = 4L + 4nL + 2nkL + 2n_1 + 2n_2 + 2nk$$

and the size of $\mathcal{M}$ will be

$$\mathfrak{l} = 3L + 2nL + 2nkL + 2n_1 + 2n_2 + 2nk$$

(the overlapped part, namely, $\boldsymbol{t}_i$ and $\boldsymbol{m}_i$, should be counted only once).

Now, let $\hat{l}_1, \hat{l}_2$, $\hat{\sigma}_1$, $\hat{\sigma}_2$, $\hat{p}$, $\hat{\kappa}$, $\hat{N}$ be parameters used in the main protocol. Then the proof contains

$$\|\pi\| = (\log(2\hat{p}+1) + \hat{\kappa} + (3\hat{l}_1 + 2\hat{l}_2 + 2\mathfrak{n} + 2\mathfrak{l}) \cdot k) \cdot \hat{N} + (\hat{l}_1 + \mathfrak{n}) \cdot k$$

bits.

Next, we analyze the size of the ciphertexts. Each ciphertext contains two vectors, one of which is of length n_1 and the other is of length L. So, the two ciphertexts contain $2(n_1 + L) \cdot k$ bits.

In summary, each signature in our gorup siganture scheme contains:

$$\begin{aligned}\|\sigma\| = (\log(2\hat{p}+1) + \hat{\kappa} + (3\hat{l}_1 + 2\hat{l}_2 + 2\mathfrak{n} + 2\mathfrak{l}) \cdot k) \cdot \hat{N} \\ + (\hat{l}_1 + \mathfrak{n}) \cdot k + 2(n_1 + L) \cdot k \quad (6)\end{aligned}$$

bits.

Choosing the Parameters. Next, we estimate the concrete size of the signature via setting concrete values of parameters involved in Equation (6). Note that security of the group signature scheme relies on the following 6 assumptions:

$$\begin{cases} SIS_{\hat{l}_1,q,\beta_1} \\ SIS_{\hat{l}_1,q,\beta_2} \\ LWE_{\hat{l}_2,q,\alpha} \\ \\ SIS_{n,q,\beta_3} \\ \\ BLWE_{n_2,q} \\ BLWE_{n_1,q} \end{cases}$$

where $\beta_1 = 16\hat{p} \cdot \sqrt{\hat{l}_1 + \hat{l}_2 + \mathfrak{n}} \cdot (\hat{\sigma}_2 + \hat{p} \cdot \hat{\sigma}_1)$, $\beta_2 = 16\hat{p} \cdot \sqrt{\hat{l}_1 + \hat{l}_2 + \mathfrak{l}} \cdot (\hat{\sigma}_2 + \hat{p} \cdot \hat{\sigma}_1)$, $\beta_3 = \sqrt{2nk}$, $\alpha = \frac{\sqrt{2\pi} \cdot \hat{\sigma}_1}{q}$. Thus, we should guaratee that the above assumptions are valid while picking parameters. As discussed in Sec. B.1, this can be achieved by setting the parameters in a way that a small root-Hermite factor is required to break the assumptions.

More concretely, we set these parameters as per Table 5. Our parameters are set to achieve 80-bit security, 100-bit security and 128-bit security respectively in a setting that we fix $L = 10$ (i.e., our group signature supports a group of 2^{10} users) and $\hat{\kappa} = 128$.

Table 5: Concrete Parameters for Our Group Signature.

Soundness Error	2^{-80}	2^{-100}	2^{-128}
δ_0	1.0048	1.0042	1.0035
$\hat{p}$	2^{20}	2^{25}	2^{32}
$\hat{l}_1$	2300	3100	4400
$\hat{l}_2$	2600	3400	4700
$\hat{N}$	4	4	4
$\lceil \log q \rceil$	80	90	105
n	14	15	16
n_1	3100	3900	5400
n_2	3100	3900	5400
Signature Size	6.94MB	9.61MB	14.57MB

F Electronic Cash

F.1 The Definition

In this section, we recall the standard syntax and security requirement of a (compact) electronic cash system, which is defined in [23].

An electronic cash system consists of the following algorithms and protocols:

- **Setup.** On input a security parameter 1^λ, the setup algorithm outputs the public parameter $param$ for the system.
- **BKGen.** The bank key generation algorithm outputs a secret key/public key pair (sk_B, pk_B) for the bank.
- **UKGen.** The user key generation algorithm outputs a secret key/public key pair (sk_B, pk_B) for a user.
- **Withdraw.** This is an interactive protocol run between a user and the bank, where the user withdraws N coins from the bank. In this protocol, the user's input is his secret key, his public key and the bank's public key, while the bank's input is its secret key, its public key, its internal state and the user's public key. After executing the protocol, the user's output is either a wallet $\mathcal{W}$ of N coins or an error message, and the bank's output is either an updated state, which allows it to trace users that double-spends some coin, or an error message.
- **Spend.** This is an interactive protocol run between a user and a merchant, where the user spends a coin from his wallet to the merchant. In this protocol, the user's input is his wallet $\mathcal{W}$, his public key, some auxiliary information $info \in \{0,1\}^*$ containing transaction information from the merchant to the user, and the bank's public key, while the merchant's input is the bank's public key. After executing the protocol, the user's output is an updated

wallet $\mathcal{W}'$, and the merchant's output is a coin, which contains a serial number and a proof of validity.

- **Deposit.** This is an interactive protocol run between a merchant and the bank, where the merchant deposits a coin received from a user into her account at the bank. In this protocol, the merchant's input is a coin and the bank's public key while the bank's input is its secret key. After executing the protocol, the merchant outputs either an error message or nothing. The bank updates the merchant's account and its state if the coin is valid and no double-spending is detected. Otherwise, it outputs an error message.
- **Identify.** The identify algorithm takes as input two coins with the same serial number, and outputs the double-spender's public key together with a proof of guilt for the detected user.
- **VerifyGuilt.** The guilt verification algorithm takes as input a user public key pk and a proof of guilt, and outputs a bit indicting if the user with public key pk is a double-spender.

Next, we describe security requirements for E-cash systems. We only give an informal description for each security requirement and refer the readers to previous works, e.g., [23, 55], for the formal definitions.

- **Correctness.** The correctness requires that the withdraw protocol, the deposit protocol and the spend protocol will output the correct results when executed by honest parties. In particular, an honest user is able to obtain a wallet from the withdraw protocol and is able to spend a coin in the spend protocol if his wallet is not empty. Also, an honest merchant is able to deposit the coin if she sends a coin received from an honest user to the bank.
- **Compactness.** Assume each new wallet in the system has N coins. The compactness requires that the communication costs in both the withdraw protocol and the spend protocol are proportional to $\log N$. Also, it requires that the size of the wallet is proportional to $\log N$.
- **Balance.** The balance property requires that no collection of users and merchants can produce more serial numbers than they have withdrawn.
- **Identification of double spender.** This property requires that once a user outputs two coins with the same serial number, he will be caught by the identify algorithm.
- **Anonymity of users.** The anonymity property requires that no one can locate the spender of a coin if the spender does not double-spend.
- **Exculpability.** The exculpability property requires that no honest user will be falsely identified as a double-spender.

F.2 The Construction

Overview. We construct the compact electronic cash system following the blueprint presented in [23]. More precisely, the secret key of the bank is a secret key of a signature scheme and the secret key/public key pair of a user is an input/output pair of a one-way function.

To withdraw money (N coins) from the bank, the user first generates two independent secret keys for the underlying PRF. Then he sends the two PRF keys as well as his own secret key to the bank and gets a signature on them (actually, he will send a commitment of the keys and thus needs the signature scheme to be able to sign on the commited value given a commitment).

To spend one coin to a merchant, who sends a "challenge" c in the beginning, the user first chooses a fresh input from $[1, N]$. Then he computes the PRF function with certificated keys on this input and gets two outputs z_1 and z_2. Next, he uses z_1 as the serial number of the coin and generates a security tag for the coin via masking his public key with $c \cdot z_2$. Finally, he generates a proof proving that both the serial number and the security tag are properly generated and sends the serial number, the security tag and the proof to the merchant. The merchant accepts the coin if the proof sent is valid.

Double-spending can be detected once there exist repeated serial numbers and it is easy to recover the double-spender's public key from the security tags in the two transcripts containing the same serial number.

Building Blocks. We will use the signature scheme presented in [52], which can sign on a commitment. We will use a full-rank difference function H_{FRD} [1, 28] to map the challenge in a way that for any different challenges c_1 and c_2, $H_{RFD}(c_1) - H_{RFD}(c_2)$ is a full rank matrix.

We will use the weak pseudorandom function F in Appendix A.5 and upgrade it to a standard one in the randome oralce model. More precisely, let H be a hash function, which is modeled as a random oracle. The new PRF function $\mathsf{F}'(k, x) = \mathsf{F}(k, H(x))$. However, we do not know how to efficiently argue the correct evaluation of F while hiding the input x directly.

To solve this problem, we use a cryptographic accumulator scheme $\mathsf{Acc} = (\mathsf{Acc}.\mathtt{Acc}, \mathsf{Acc}.\mathtt{Witness}, \mathsf{Acc}.\mathtt{Verify})$. In more detail, to aruge knowledge of k, x satisfying $y = \mathsf{F}(k, H(x))$ for a public y, the prover first accumulates all $H(i)$ for $i \in [1, N]$ (recall that we only allows the prover to compute the PRF on inputs from $[0, N-1]$ in the system). Then he proves knowledge of z, k that $y = \mathsf{F}(k, z)$ and z is properly accumulated.

Note that, the main reason we do not use current lattice-based pseudorandom functions (e.g., [8, 9, 16]) in our construction is that they have an extremely low concrete efficiency.

The Construction. Next, we give a detailed description of our E-cash system. Our system is similar to the one presented in [55], but we replace the underlying zero-knowledge arguments with the ones developped in this work. We also introduce several tricks to improve the efficiency of the system.

Let λ be the security parameter. Let e_1, e_2 be postive integers that $1 \leq e_1 < e_2$. Let q_0 be a prime number and let $p = q_0^{e_1}$, $q = q_0^{e_2}$. Let $k = \lceil \log q \rceil$. Let $m_1, m_2, m_3, \sigma, \beta$ be positive integers that $m_2 = 2km_1$, $m_3 = \lambda$, $\sigma = 1.6k \cdot \sqrt{m_1}$ and $\beta = 12\sigma$. Let n_1, n_2 be positive integers. Let l_1, l_2 be positive integers that $l_2 = kl_1$. Let n_3 be positive integers that $n_3 = n_1 \cdot (\lceil \log p \rceil + 1)$. Let l_3, l_4, l_5, ι

be positive integers and $k' = k/\iota$. Let $N = 2^L$ be the number of coins in a wallet.

Let $\mathsf{Sig} = (\mathsf{Sig}.\mathtt{KeyGen}, \mathsf{Sig}.\mathtt{Sign}, \mathsf{Sig}.\mathtt{Verify})$ be the signature scheme recalled in Sec. A.3. Let F be the weak PRF recalled in Sec. A.5. Let $\mathsf{Acc} = (\mathsf{Acc}.\mathtt{Acc}, \mathsf{Acc}.\mathtt{Witness}, \mathsf{Acc}.\mathtt{Verify})$ be the accumulator scheme recalled in Sec. A.4. Let $H_{FRD} : \mathbb{Z}_p^{n_1} \to \mathbb{Z}_p^{n_1 \times n_1}$ be a full-rank difference function. Let $H : [0, N-1] \to \mathbb{Z}_q^{n_1 \times n_2}$ be hash function that is modeled as a random oracle. Let $H' : \{0,1\}^* \to \mathbb{Z}_p^{n_1}$ be a hash function that is modeled as a random oracle.

The electronic cash system EC works as follows:

- **Setup.** The setup algorithm first samples $\boldsymbol{D}_1, \boldsymbol{D}_2 \xleftarrow{\$} \mathbb{Z}_q^{l_1 \times l_2}$, which are the public parameter for the accumulator scheme. Then it samples $\boldsymbol{E} \xleftarrow{\$} \mathbb{Z}_p^{n_1 \times n_3}$, which maps a user secret key to a user public key. It also samples $\boldsymbol{F}_1 \xleftarrow{\$} \mathbb{Z}_q^{2l_1 \times kl_3}$, $\boldsymbol{F}_2 \xleftarrow{\$} \mathbb{Z}_q^{l_3 \times k' n_1 n_2}$, $\boldsymbol{F}_3 \xleftarrow{\$} \mathbb{Z}_q^{l_4 \times 2k' n_2}$, and $\boldsymbol{F}_4 \xleftarrow{\$} \mathbb{Z}_q^{l_5 \times k' n_2}$, which is used to connect different ingredients in the system.
  Finally, the setup algorithm outputs the public parameter $param = (\boldsymbol{D}_1, \boldsymbol{D}_2, \boldsymbol{E}, \boldsymbol{F}_1, \boldsymbol{F}_2, \boldsymbol{F}_3)$.
- **BKGen.** The bank generates a secret key/public key pair for Sig. More precisely, it runs

$$(\boldsymbol{T}, (\boldsymbol{B}, \{\boldsymbol{B}_i\}_{i \in [0,m_3]}, \boldsymbol{u}, \tilde{\boldsymbol{B}}, \tilde{\boldsymbol{B}}_{\boldsymbol{0}}, \tilde{\boldsymbol{B}}_{\boldsymbol{1}})) \leftarrow \mathsf{Sig}.\mathtt{KeyGen}(1^\lambda)$$

  where $\boldsymbol{B} \in \mathbb{Z}_q^{m_1 \times m_2}$, $\forall i \in [0, m_3]$, $\boldsymbol{B}_i \in \mathbb{Z}_q^{m_1 \times m_2}$, $\boldsymbol{u} \in \mathbb{Z}_q^{m_1}$, $\tilde{\boldsymbol{B}} \in \mathbb{Z}_q^{m_1 \times km_1}$, $\tilde{\boldsymbol{B}}_0 \in \mathbb{Z}_q^{m_1 \times 2m_2}$, $\tilde{\boldsymbol{B}}_1 \in \mathbb{Z}_q^{m_1 \times (kl_4 + kl_5 + n_3)}$, and $\boldsymbol{T}$ is the trapdoor for $\boldsymbol{B}$.
  Here, let $\tilde{\boldsymbol{B}}_0 = (\tilde{\boldsymbol{B}}_0' \| \tilde{\boldsymbol{B}}_0'') \in \mathbb{Z}_q^{m_1 \times m_2} \times \mathbb{Z}_q^{m_1 \times m_2}$ and $\tilde{\boldsymbol{B}}_1 = (\tilde{\boldsymbol{B}}_1' \| \tilde{\boldsymbol{B}}_1'') \in \mathbb{Z}_q^{m_1 \times (kl_4 + n_3)} \times \mathbb{Z}_q^{m_1 \times kl_5}$, then $\tilde{\boldsymbol{B}}_0'$ and $\tilde{\boldsymbol{B}}_1'$ also serve as the public key of the commitment scheme used in the withdraw protocol
  Finally, the setup algorithm outputs the bank's public key $pk_B = (\boldsymbol{B}, \{\boldsymbol{B}_i\}_{i \in [0,m_3]}, \boldsymbol{u}, \tilde{\boldsymbol{B}}, \tilde{\boldsymbol{B}}_{\boldsymbol{0}}, \tilde{\boldsymbol{B}}_{\boldsymbol{1}})$ and its secret key $sk_B = \boldsymbol{T}$.
- **UKGen.** The user key generation algorithm samples $\boldsymbol{s} \xleftarrow{\$} \{0,1\}^{n_3}$ and computes $\boldsymbol{t} = \boldsymbol{E} \cdot \boldsymbol{s}) \in \mathbb{Z}_p^{n_1} \mod p$. Then it outputs $sk = \boldsymbol{s}$ and $pk = \boldsymbol{t}$.
- **Withdraw.** The withdraw protocol is run between a user and the bank. Here, the user's input is his secret key $\boldsymbol{s}$, his public key $\boldsymbol{t}$ and the bank's public key $pk_B = (\boldsymbol{B}, \{\boldsymbol{B}_i\}_{i \in [0,m_3]}, \boldsymbol{u}, \tilde{\boldsymbol{B}}, \tilde{\boldsymbol{B}}_{\boldsymbol{0}}, \tilde{\boldsymbol{B}}_{\boldsymbol{1}})$. The bank's input is its secret key $\boldsymbol{T}$, its public key $pk_B = (\boldsymbol{B}, \{\boldsymbol{B}_i\}_{i \in [0,m_3]}, \boldsymbol{u}, \tilde{\boldsymbol{B}}, \tilde{\boldsymbol{B}}_{\boldsymbol{0}}, \tilde{\boldsymbol{B}}_{\boldsymbol{1}})$ and the user's public key $\boldsymbol{t}$. The protocol proceeds as follows:

  1. To withdraw N coins from the bank, the user first samples $\boldsymbol{k}_0 \xleftarrow{\$} \mathbb{Z}_q^{n_2}$ and $\boldsymbol{k}_1 \xleftarrow{\$} \mathbb{Z}_q^{n_2}$, which serve as PRF keys.
     Then he generates a commitment of the two PRF keys as well as his secret key. More precisely, the user first decomposes the two vectors $\boldsymbol{k}_0$ and $\boldsymbol{k}_1$ into vectors $\bar{\boldsymbol{k}}_{\boldsymbol{0}}$ and $\bar{\boldsymbol{k}}_{\boldsymbol{1}}$ respectively, whose elements are ι-bits

integers, i.e.,

$$\forall b \in [0,1], i \in [1, n_2], \quad \boldsymbol{k}_b[i] = \sum_{j=1}^{k'} ((2^\iota)^{j-1} \cdot \bar{\boldsymbol{k}}_b[(i-1)\cdot k' + j])$$

Then he generates $\boldsymbol{d}_0 = \mathtt{bin}(\boldsymbol{F}_3 \cdot (\bar{\boldsymbol{k}}_0^\intercal \| \bar{\boldsymbol{k}}_1^\intercal)^\intercal)$, which is a digest of $\boldsymbol{k}_0$ and $\boldsymbol{k}_1$ Finally, it samples $\boldsymbol{r}_0 \in D_\sigma^{m_2}$ and computes the commitment $\boldsymbol{c}_0 = \tilde{\boldsymbol{B}}_0^{'} \cdot \boldsymbol{r}_0 + \tilde{\boldsymbol{B}}_1^{'} \cdot (\boldsymbol{d}_0^\intercal \| \boldsymbol{s}^\intercal)^\intercal$.
Next, the user generates a proof proving that the commitment is properly generated. More concretely, it computes:

$$\begin{aligned}\pi_w = SPK\{(\tilde{\boldsymbol{B}}_0^{'}, \tilde{\boldsymbol{B}}_1^{'}, \boldsymbol{F}_3, \boldsymbol{E}, \boldsymbol{D}_1, \boldsymbol{D}_2, \boldsymbol{c}_0, \boldsymbol{t}), (\bar{\boldsymbol{k}}_0, \bar{\boldsymbol{k}}_1, \boldsymbol{s}, \boldsymbol{r}_0, \boldsymbol{d}_0) :& \\ \boldsymbol{c}_0 = \tilde{\boldsymbol{B}}_0^{'} \cdot \boldsymbol{r}_0 + \tilde{\boldsymbol{B}}_1^{'} \cdot (\boldsymbol{d}_0^\intercal \| \boldsymbol{s}^\intercal)^\intercal \ \wedge\ \|\boldsymbol{r}_0\|_\infty \le \beta \ \wedge& \\ \boldsymbol{d}_0 \in \{0,1\}^{kl_4} \ \wedge\ \boldsymbol{s} \in \{0,1\}^{n_3} \ \wedge& \\ \frac{q}{p} \cdot \boldsymbol{t} = \frac{q}{p} \cdot \boldsymbol{E} \cdot \boldsymbol{s} \ \wedge\ \boldsymbol{H}_1 \cdot \boldsymbol{d}_0 = \boldsymbol{F}_3 \cdot (\bar{\boldsymbol{k}}_0^\intercal \| \bar{\boldsymbol{k}}_1^\intercal)^\intercal \ \wedge& \\ \bar{\boldsymbol{k}}_{\boldsymbol{0}} \in [0, 2^\iota - 1]^{k'n_2} \ \wedge\ \bar{\boldsymbol{k}}_{\boldsymbol{1}} \in [0, 2^\iota - 1]^{k'n_2}\}&\end{aligned}$$

where $\boldsymbol{H}_1 = \boldsymbol{I}_{l_4} \otimes (1\ 2\ 4 \ldots 2^{k-1})$. Note that for any $a, b \in \mathbb{Z}_p$, $a = b \mod p$ iff $\frac{q}{p} \cdot a = \frac{q}{p} \cdot b \mod q$, so we can lift an equation in $\mathbb{Z}_p$ to an equation in $\mathbb{Z}_q$ via multiplying q/p in both side of the equation.
Finally, the user sends the commitment $\boldsymbol{c}_0$ and the proof π_w to the bank.

2. On receiving the user's commitment $\boldsymbol{c}_0$ and a proof π_w, the bank first verifies if the proof is valid, it aborts if the proof is invalid.
Then, it samples $\boldsymbol{k}_2 \xleftarrow{\$} \mathbb{Z}_q^{n_2}$ and decomposes it into a vector $\bar{\boldsymbol{k}}_{\boldsymbol{2}}$, whose elements are ι-bits integers, i.e., for $i \in [1, n_2]$, $\boldsymbol{k}_2[i] = \sum_{j=1}^{k'} ((2^\iota)^{j-1} \cdot \bar{\boldsymbol{k}}_2[(i-1)\cdot k' + j])$. It also generates a digest $\boldsymbol{d}_1 = \mathtt{bin}(\boldsymbol{F}_4 \cdot \bar{\boldsymbol{k}}_2)$ for $\boldsymbol{k}_2$.
Next, the bank samples $\boldsymbol{r}_1 \in D_\sigma^{m_2}$ and re-randomize the commitment $\boldsymbol{c} = \boldsymbol{c}_0 + \tilde{\boldsymbol{B}}_0^{''} \cdot \boldsymbol{r}_1 + \tilde{\boldsymbol{B}}_1^{''} \cdot \boldsymbol{d}_1$
Then, the bank signs on the commitment $\boldsymbol{c}$ as follows. First, it samples $\boldsymbol{\tau} \xleftarrow{\$} \{0,1\}^{m_3}$ and computes the matrix

$$\boldsymbol{B}_{\boldsymbol{\tau}} = (\boldsymbol{B} \| \boldsymbol{B}_0 + \sum_{i=1}^{m_3} (\boldsymbol{\tau}[i] \cdot \boldsymbol{B}_i))$$

Then it use the secret $\boldsymbol{T}$ to samples a vector $\boldsymbol{v} \in D_\sigma^{2m_2}$ that satisfies $\boldsymbol{B}_{\boldsymbol{\tau}} \cdot \boldsymbol{v} = \boldsymbol{u} + \tilde{\boldsymbol{B}} \cdot \mathtt{bin}(\boldsymbol{c})$.
Finally, the bank sends $\boldsymbol{k}_1, \boldsymbol{\tau}, \boldsymbol{r}_1, \boldsymbol{v}$ back to the user.

3. On receiving vectors $\boldsymbol{k}_1, \boldsymbol{\tau}, \boldsymbol{r}_1, \boldsymbol{v}$, the user first checks if $(\boldsymbol{\tau}, \boldsymbol{r}, \boldsymbol{v})$ is a valid signature for $(\boldsymbol{d}_0^\intercal \| \boldsymbol{s}^\intercal \| \boldsymbol{d}_1^\intercal)^\intercal$, where $\boldsymbol{r} = (\boldsymbol{r}_0^\intercal \| \boldsymbol{r}_1^\intercal)^\intercal$, $\boldsymbol{d}_1 = \mathtt{bin}(\boldsymbol{F}_4 \cdot \bar{\boldsymbol{k}}_2)$ and $\bar{\boldsymbol{k}}_{\boldsymbol{2}}$ decomposes each element $\boldsymbol{k}_2$ into ι-bits integers. If so, the user sets his wallet as

$$\mathcal{W} = (\boldsymbol{s}, \boldsymbol{k}_0, \boldsymbol{k}_1, \boldsymbol{k}_2, \boldsymbol{\tau}, \boldsymbol{r}, \boldsymbol{v}, 0)$$

4. After that, the bank records a debit of N for the account with public key $pk = \boldsymbol{t}$.

- **Spend.** The spend protocol is run between a user and a merchant. Here, the user's input is his wallet $\mathcal{W} = (\boldsymbol{s}, \boldsymbol{k}_0, \boldsymbol{k}_1, \boldsymbol{k}_2, \boldsymbol{\tau}, \boldsymbol{r}, \boldsymbol{v}, J)$, his public key $\boldsymbol{t}$, some auxiliary information $\mathit{info} \in \{0,1\}^*$ containing transaction information from the merchant to the user, and the bank's public key $pk_B = (\boldsymbol{B}, \{\boldsymbol{B}_i\}_{i \in [0,m_3]}, \boldsymbol{u}, \tilde{\boldsymbol{B}}, \tilde{\boldsymbol{B}}_{\boldsymbol{0}}, \tilde{\boldsymbol{B}}_{\boldsymbol{1}})$. The merchant's input is the bank's public key $pk_B = (\boldsymbol{B}, \{\boldsymbol{B}_i\}_{i \in [0,m_3]}, \boldsymbol{u}, \tilde{\boldsymbol{B}}, \tilde{\boldsymbol{B}}_{\boldsymbol{0}}, \tilde{\boldsymbol{B}}_{\boldsymbol{1}})$. The protocol proceeds as follows.

  To spend a coin to the merchant, the user first checks if his wallet is empty. He aborts if $J = N$ and otherwise, he generates matrices $\boldsymbol{R} = H_{FRD}(H'(\mathit{info}))$ and $\boldsymbol{C} = H(J)$. Then he computes $\boldsymbol{k} = \boldsymbol{k}_1 + \boldsymbol{k}_2$ and generates the security tag and the serial number for the coin:

$$\begin{aligned} \boldsymbol{z}_0 &= \boldsymbol{R} \cdot \mathsf{F}(\boldsymbol{k}_0, \boldsymbol{C}) + \boldsymbol{t} \mod p \\ \boldsymbol{z}_1 &= \mathsf{F}(\boldsymbol{k}, \boldsymbol{C}) \end{aligned}$$

  Then, the user generates an accumulator for all valid inputs.[9] More precisely, for $i \in [0, N-1]$, let $\boldsymbol{C}_i = H(i)$ and let $\boldsymbol{c}_i$ be the concatenation of row vectors of $\boldsymbol{C}_i$, i.e., $\boldsymbol{c}_i[(\imath - 1) \cdot n_2 + \jmath] = \boldsymbol{C}_i[\imath, \jmath]$ for $\imath \in [1, n_1]$, $\jmath \in [1, n_2]$. The user first decompose the vector $\boldsymbol{c}_i$ into a vector $\bar{\boldsymbol{c}}_i$ whose elements are ι-bits integers, i.e.,

$$\forall i \in [1, n_1 n_2], \quad \boldsymbol{c}[i] = \sum_{j=1}^{k'} ((2^\iota)^{j-1} \cdot \bar{\boldsymbol{c}}[(i-1) \cdot k' + j])$$

  Then, he generates a digest $\boldsymbol{e}_i$ for each matrix, that is, $\boldsymbol{e}_i = \mathtt{bin}(\boldsymbol{F}_2 \cdot \bar{\boldsymbol{c}}_i)$. He also maps the intput and the digest to the input space of the accumulator and generates $\bar{\boldsymbol{e}}_i = \mathtt{bin}(\boldsymbol{F}_1 \cdot \boldsymbol{e}_i)$ and $\tilde{\boldsymbol{e}}_i = \mathtt{bin}((\boldsymbol{D}_1 \| \boldsymbol{D}_2) \cdot \bar{\boldsymbol{e}})$. Next, the user accumulates all $\tilde{\boldsymbol{e}}_i$ via the accumulator scheme and gets $\tilde{\boldsymbol{u}} = \mathsf{Acc}\,.\,\mathtt{Acc}_{\boldsymbol{D}_1, \boldsymbol{D}_2}(\{\tilde{\boldsymbol{e}}_i\}_{i \in [0,N-1]})$. Let $\boldsymbol{c} = \boldsymbol{c}_J$, $\bar{\boldsymbol{c}} = \bar{\boldsymbol{c}}_J$, $\boldsymbol{e} = \boldsymbol{e}_J$, $\bar{\boldsymbol{e}} = \bar{\boldsymbol{e}}_J$ and $\tilde{\boldsymbol{e}} = \tilde{\boldsymbol{e}}_J$. Then the user generates the witness $\mathfrak{w} \leftarrow \mathsf{Acc}\,.\,\mathtt{Witness}_{\boldsymbol{D}_1, \boldsymbol{D}_2}(\{\tilde{\boldsymbol{e}}_i\}_{i \in [0,N-1]}, \tilde{\boldsymbol{e}})$.
  Next, the user generates a proof π_s as in Figure 5, where $\boldsymbol{G}_1 = \boldsymbol{I}_{n_2} \otimes \boldsymbol{g}$, $\boldsymbol{G}_2 = \boldsymbol{I}_{n_1 n_2} \otimes \boldsymbol{g}$, $\boldsymbol{H}_1 = \boldsymbol{I}_{l_4} \otimes \boldsymbol{h}$, $\boldsymbol{H}_2 = \boldsymbol{I}_{l_5} \otimes \boldsymbol{h}$, $\boldsymbol{H}_3 = \boldsymbol{I}_{l_3} \otimes \boldsymbol{h}$, $\boldsymbol{H}_4 = \boldsymbol{I}_{2l_1} \otimes \boldsymbol{h}$, $\boldsymbol{H}_5 = \boldsymbol{I}_{l_1} \otimes \boldsymbol{h}$, and $\boldsymbol{g} = (1\ 2^\iota\ 2^{2\iota} \ldots 2^{(k'-1)\iota})$, $\boldsymbol{h} = (1\ 2\ 4 \ldots 2^{k-1})$.

  The coin sent to the merchant is $\mathit{coin} = (\boldsymbol{R}, \boldsymbol{z}_0, \boldsymbol{z}_1, \pi_s)$, and the merchant accepts it iff the proof π_s is valid.

  Then the user update his wallet via increasing the counter J by 1.

[9] In practice, to reduce the computational cost of the user and the merchant, we can generate the accumulator and witnesses for all inputs in the setup phase and put it into the public parameter.

$$\pi_w = SPK\left\{\begin{array}{l}(\boldsymbol{k}_0,\boldsymbol{k}_1,\boldsymbol{k}_2,\boldsymbol{k},\boldsymbol{z}_0',\boldsymbol{C},\boldsymbol{t},\bar{\boldsymbol{k}}_0,\bar{\boldsymbol{k}}_1,\bar{\boldsymbol{k}}_2,\boldsymbol{d}_0,\boldsymbol{d}_1,\boldsymbol{\tau},\boldsymbol{r},\boldsymbol{v},\boldsymbol{s},\boldsymbol{c},\bar{\boldsymbol{c}},\boldsymbol{e},\bar{\boldsymbol{e}},\tilde{\boldsymbol{e}},\mathfrak{w}):\\ \\ \boldsymbol{z}_0' = \mathsf{F}(\boldsymbol{k}_0,\boldsymbol{C}) \ \wedge \ \frac{q}{p}\cdot\boldsymbol{z}_0 = \frac{q}{p}\cdot\boldsymbol{R}\cdot\boldsymbol{z}_0' + \frac{q}{p}\cdot\boldsymbol{t} \ \wedge\\ \boldsymbol{z}_1 = \mathsf{F}(\boldsymbol{k},\boldsymbol{C}) \ \wedge \ \boldsymbol{k} = \boldsymbol{k}_1+\boldsymbol{k}_2 \ \wedge\\ \\ \boldsymbol{k}_0 = \boldsymbol{G}_1\cdot\bar{\boldsymbol{k}}_0 \ \wedge \ \boldsymbol{k}_1 = \boldsymbol{G}_1\cdot\bar{\boldsymbol{k}}_1 \ \wedge \ \boldsymbol{k}_2 = \boldsymbol{G}_1\cdot\bar{\boldsymbol{k}}_2 \ \wedge\\ \boldsymbol{H}_1\cdot\boldsymbol{d}_0 = \boldsymbol{F}_3\cdot(\bar{\boldsymbol{k}}_0^\intercal\|\bar{\boldsymbol{k}}_1^\intercal)^\intercal \ \wedge \ \boldsymbol{H}_2\cdot\boldsymbol{d}_1 = \boldsymbol{F}_4\cdot\bar{\boldsymbol{k}}_2 \ \wedge\\ \bar{\boldsymbol{k}}_{\boldsymbol{0}} \in [0,2^\iota-1]^{k'n_2} \ \wedge \ \bar{\boldsymbol{k}}_{\boldsymbol{1}} \in [0,2^\iota-1]^{k'n_2} \ \wedge\\ \bar{\boldsymbol{k}}_{\boldsymbol{2}} \in [0,2^\iota-1]^{k'n_2} \ \wedge\\ \boldsymbol{d}_0 \in \{0,1\}^{kl_4} \ \wedge \ \boldsymbol{d}_1 \in \{0,1\}^{kl_5} \ \wedge\\ \\ \mathsf{Sig}.\mathtt{Verify}((\boldsymbol{\tau},\boldsymbol{r},\boldsymbol{v}),(\boldsymbol{d}_0^\intercal\|\boldsymbol{s}^\intercal\|\boldsymbol{d}_1^\intercal)^\intercal) = 1 \ \wedge\\ \\ \frac{q}{p}\cdot\boldsymbol{t} = \frac{q}{p}\cdot\boldsymbol{E}\cdot\boldsymbol{s} \ \wedge \ \boldsymbol{s}\in\{0,1\}^{n_3} \ \wedge\\ \\ \boldsymbol{c} = \boldsymbol{G}_2\cdot\bar{\boldsymbol{c}} \ \wedge \ \boldsymbol{H}_3\cdot\boldsymbol{e} = \boldsymbol{F}_2\cdot\bar{\boldsymbol{c}} \ \wedge\\ \boldsymbol{H}_4\cdot\bar{\boldsymbol{e}} = \boldsymbol{F}_1\cdot\boldsymbol{e} \ \wedge \ \boldsymbol{H}_5\cdot\tilde{\boldsymbol{e}} = (\boldsymbol{D}_1\|\boldsymbol{D}_2)\cdot\bar{\boldsymbol{e}} \ \wedge\\ \bar{\boldsymbol{c}} \in [0,2^\iota-1]^{k'n_1n_2} \ \wedge \ \boldsymbol{e}\in\{0,1\}^{kl_3} \ \wedge\\ \bar{\boldsymbol{e}}\in\{0,1\}^{2kl_1} \ \wedge \ \tilde{\boldsymbol{e}}\in\{0,1\}^{kl_1} \ \wedge\\ \\ \mathsf{Acc}.\mathtt{Verify}(\tilde{\boldsymbol{u}},\tilde{\boldsymbol{e}},\mathfrak{w}) = 1\end{array}\right\}$$

Fig. 5 The generation of the proof π_s used in the Spend protocol. Here, we omit the public part of the proved statement for simplicity of description.

- **Deposit.** The deposit protocol is run between a merchant and the bank. The merchant sends a coin $coin = (\boldsymbol{R},\boldsymbol{z}_0,\boldsymbol{z}_1,\pi_s)$ to the bank.
  Then the bank checks if π_s is a valid proof and if the serial number $\boldsymbol{z}_1$ has not been appeared in the list $\mathcal{L}$ of all previous coins. The bank accepts the coin, adds $coin$ to the list $\mathcal{L}$ and credits the merchant's account if both checks are passed. Otherwise, it returns an error message.
- **Identify.** The input of the identification algorithm is two coins $coin^{(1)} = (\boldsymbol{R}^{(1)},\boldsymbol{z}_0^{(1)},\boldsymbol{z}_1^{(1)},\pi_s^{(1)})$ and $coin^{(2)} = (\boldsymbol{R}^{(2)},\boldsymbol{z}_0^{(2)},\boldsymbol{z}_1^{(2)},\pi_s^{(2)})$ that $\boldsymbol{R}^{(1)} \neq \boldsymbol{R}^{(2)}$ and $\boldsymbol{z}_0^{(1)} \neq \boldsymbol{z}_0^{(2)}$. First, the algorithm computs $\boldsymbol{z} = (\boldsymbol{R}^{(1)}-\boldsymbol{R}^{(2)})^{-1}\cdot(\boldsymbol{z}_0^{(1)} - \boldsymbol{z}_0^{(2)}) \mod p$ and $\boldsymbol{t} = \boldsymbol{z}_0^{(1)} - \boldsymbol{R}\cdot\boldsymbol{z} \mod p$. Then it outputs the identity $\boldsymbol{t}$ and a proof of guilt $\pi_G = (coin^{(1)}, coin^{(2)})$.
- **VerifyGuilt.** The input of the guilt verification algorithm includes a user public key $\boldsymbol{t}$ and a proof of guilt $\pi_G = (coin^{(1)}, coin^{(2)})$, where $coin^{(1)} =$

$(\boldsymbol{R}^{(1)}, \boldsymbol{z}_0^{(1)}, \boldsymbol{z}_1^{(1)}, \pi_s^{(1)})$ and $coin^{(2)} = (\boldsymbol{R}^{(2)}, \boldsymbol{z}_0^{(2)}, \boldsymbol{z}_1^{(2)}, \pi_s^{(2)})$. The algorithm first gets a user public key $\boldsymbol{t'}$ via running the identification algorithm on $(coin^{(1)}, coin^{(2)})$. Then it outputs "accept" iff $\boldsymbol{t} = \boldsymbol{t'}$ and both $\pi_s^{(1)}$ and $\pi_s^{(2)}$ are valid.

Theorem F.1. *Assuming the worst-case hardness of GapSVP$_\gamma$ (or SIVP$_\gamma$) for some polynomial γ, EC is a secure compact electronic cash system in the random oracle model.*

Proof of Theorem F.1 is similar to the security proof for the LLNW E-cash system proposed in [55], and we omit its details in this work.

Note that we use the "random oracle+weak PRF" to replace the PRF used in the LLNW system. This will not affect the security since the random oracle can map any input into a random one. Also, the use of accumulator allows the user to prove to the merchant that the "random" input is valid. Another difference between our system and the LLNW system is that instead of signing on the PRF keys directly, we first generates digests for the PRF keys and signs on the digest. This will also not compromise the security due to the SIS assumption.

F.3 The Efficiency

We focus on the communication cost of the system. In particular, we will consider the communication cost in the withdraw protocol and that in the spend protocol. Here, we use $\hat{l}_1, \hat{l}_2$, $\hat{\sigma}_1$, $\hat{\sigma}_2$, $\hat{p}$, $\hat{\kappa}$, $\hat{N}$ to denote parameters used in the main protocol.

Communication Cost for the Withdraw Protocol. In the withdraw protocol, the user will first send a commitment and a proof to the bank. Then the bank sends a (partial) signature and a PRF key back to the user.

The size of the commitment is km_1 bits, the size of the PRF key is kn_2 bits, and the size of the partial signature is $m_3 + 3m_2 \cdot \lceil \log \beta \rceil$ bits.

Next, we analyze the size of the proof. The proved statement includes several linear equations with short/binary solutions and can be reduced to an instance of the relation $\mathcal{R}^*$. Here, we use the fast mode to argue that $\boldsymbol{r}_0$ is a short vector [10], and this will lead to an argument for an instance of $\mathcal{R}^*$, where the length of the witness is

$$\mathfrak{n}_1 = m_2 + \mathfrak{b}_1 \cdot \lambda \cdot (\lfloor \log(2m_2 \cdot \beta / \mathfrak{b}_1) \rfloor + 1) + kl_4 + n_3 + 2k'n_2 + \mathfrak{b}_2 \cdot \lambda \cdot (\lfloor \log(2k'n_2 \cdot (2^\iota - 1)/\mathfrak{b}_2) \rfloor + 1)$$

and the size of $\mathcal{M}$ is

$$\mathfrak{l}_1 = \mathfrak{b}_1 \lambda \cdot (\lfloor \log(2m_2 \cdot \beta / \mathfrak{b}_1) \rfloor + 1) + kl_4 + n_3 + \mathfrak{b}_2 \cdot \lambda \cdot (\lfloor \log(2k'n_2 \cdot (2^\iota - 1)/\mathfrak{b}_2) \rfloor + 1)$$

[10] To balance the size of the witness/ $\mathcal{M}$ and the hardness of the underlying SIS problem, we divide the "short" witness into several, say $\mathfrak{b}_1$, blocks and use the fast mode on each block.

Therefore, the proof contains

$$\|\pi\| = (\log(2\hat{p}+1) + \hat{\kappa} + (3\hat{l}_1 + 2\hat{l}_2 + 2\mathfrak{n}_1 + 2\mathfrak{l}_1)\cdot k)\cdot \hat{N} + (\hat{l}_1 + \mathfrak{n}_1)\cdot k$$

bits.

In summary, in each executation of the withdraw protocol, the communication cost is

$$\begin{aligned}\mathfrak{C}_W = (\log(2\hat{p}+1) + \hat{\kappa} + (3\hat{l}_1 + 2\hat{l}_2 + 2\mathfrak{n}_1 + 2\mathfrak{l}_1)\cdot k)\cdot \hat{N} + (\hat{l}_1 + \mathfrak{n}_1)\cdot k \\ + km_1 + kn_2 + m_3 + 3m_2 \cdot \lceil \log \beta \rceil \end{aligned} \quad (7)$$

bits.

Communication Cost for the Spend Protocol. In the spend protocol, the user will send a coin, which contains a matrix $\boldsymbol{R}$, two PRF outputs and a proof.

The size of $\boldsymbol{R}$ is $n_1^2 \lceil \log p \rceil$ bits. The size of a PRF output is $n_1 \lceil \log p \rceil$ bits.

Next, we analyze the size of the proof. The proved statement includes six parts. The first part contains two arguments for the correctness of weak PRF evaluation, and two linear equations. It can be reduced to an instance of $\mathcal{R}^*$, where the length of the witness is $4n_2 + 3n_1n_2 + 4n_1 + 2(\lfloor \log(\frac{q}{p} - 1)\rfloor + 1)\cdot n_1$ and the size of $\mathcal{M}$ is $2(\lfloor \log(\frac{q}{p} - 1)\rfloor + 1)\cdot n_1 + 2n_1n_2$.

The second part includes several linear equations with short/binary solutions and can be reduced to an instance of the relation $\mathcal{R}^*$. Here, we use the fast mode to argue that $\bar{\boldsymbol{k}}_0$, $\bar{\boldsymbol{k}}_1$ and $\bar{\boldsymbol{k}}_2$ are short vectors, and this will lead to an argument for an instance of $\mathcal{R}^*$, where the length of the witness is

$$\begin{aligned}3n_2 + 3k'n_2 + kl_4 + kl_5 + \mathscr{b}_2 \cdot \lambda \cdot (\lfloor \log(2k'n_2 \cdot (2^\iota - 1)/\mathscr{b}_2)\rfloor + 1) + \\ \mathscr{b}_3 \cdot \lambda \cdot (\lfloor \log(k'n_2 \cdot (2^\iota - 1)/\mathscr{b}_3)\rfloor + 1)\end{aligned}$$

and the size of $\mathcal{M}$ is

$$kl_4 + kl_5 + \mathscr{b}_2 \cdot \lambda \cdot (\lfloor \log(2k'n_2 \cdot (2^\iota - 1)/\mathscr{b}_2)\rfloor + 1) + \mathscr{b}_3 \cdot \lambda \cdot (\lfloor \log(k'n_2 \cdot (2^\iota - 1)/\mathscr{b}_3)\rfloor + 1)$$

The third part argues knowledge of a valid message/signature pair. As analyzed in Sec. 4.3, this part can be reduced to an instance of the relation $\mathcal{R}^*$, where the length of its witness is

$$m_3 + 2m_1m_3 + 4m_2 + km_1 + (kl_4 + kl_5 + n_3) + \mathscr{b}_4 \cdot \lambda \cdot (\lfloor \log(8m_2 \cdot \beta/\mathscr{b}_4)\rfloor + 1)$$

and the size of $\mathcal{M}$ is

$$m_3 + m_1m_3 + km_1 + (kl_4 + kl_5 + n_3) + \mathscr{b}_4 \cdot \lambda \cdot (\lfloor \log(8m_2 \cdot \beta/\mathscr{b}_4)\rfloor + 1)$$

where $\mathscr{b}_4 = 4\mathscr{b}_1$.

The fourth part is a linear equation with binary solution. After reducing it to an instance of $\mathcal{R}^*$, the length of the witness is $n_1 + n_3$ and the size of $\mathcal{M}$ is n_3.

The fifth part includes several linear equations with short/binary solutions and can be reduced to an instance of the relation $\mathcal{R}^*$. Here, we use the fast mode to argue that $\bar{\boldsymbol{c}}$ is a short vector, and this will lead to an argument for an instance of $\mathcal{R}^*$, where the length of the witness is

$$n_1n_2 + k'n_1n_2 + kl_3 + 2kl_1 + kl_1 + \mathscr{b}_5 \cdot \lambda \cdot (\lfloor \log(k'n_1n_2 \cdot (2^\iota - 1)/\mathscr{b}_5) \rfloor + 1)$$

and the size of $\mathcal{M}$ is

$$kl_3 + 2kl_1 + kl_1 + \mathscr{b}_5 \cdot \lambda \cdot (\lfloor \log(k'n_1n_2 \cdot (2^\iota - 1)/\mathscr{b}_5) \rfloor + 1)$$

The last part argues knowledge of an accumulated value. As analyzed in Sec. 4.4, this part can be reduced to an instance of the relation $\mathcal{R}^*$, where the length of its witness is $2L + 4l_1L + 2l_2L$ and the size of $\mathcal{M}$ is $L + 2l_1L + 2l_2L$.

So, after combing all these statements, the length of the witness will be

$$\begin{aligned}\mathfrak{n}_2 =& 4n_2 + 3n_1n_2 + 4n_1 + 2(\lfloor \log(\frac{q}{p} - 1) \rfloor + 1) \cdot n_1 + \\ & 3k'n_2 + kl_4 + kl_5 + \mathscr{b}_2 \cdot \lambda \cdot (\lfloor \log(2k'n_2 \cdot (2^\iota - 1)/\mathscr{b}_2) \rfloor + 1) \\ & + \mathscr{b}_3 \cdot \lambda \cdot (\lfloor \log(k'n_2 \cdot (2^\iota - 1)/\mathscr{b}_3) \rfloor + 1) + \\ & m_3 + 2m_1m_3 + 4m_2 + km_1 + n_3 + \mathscr{b}_4 \cdot \lambda \cdot (\lfloor \log(8m_2 \cdot \beta/\mathscr{b}_4) \rfloor + 1) + \\ & k'n_1n_2 + kl_3 + 2kl_1 + \mathscr{b}_5 \cdot \lambda \cdot (\lfloor \log(k'n_1n_2 \cdot (2^\iota - 1)/\mathscr{b}_5) \rfloor + 1) + \\ & 2L + 4l_1L + 2l_2L\end{aligned}$$

and the size of $\mathcal{M}$ will be

$$\begin{aligned}\mathfrak{l}_2 =& 2(\lfloor \log(\frac{q}{p} - 1) \rfloor + 1) \cdot n_1 + 2n_1n_2 + \\ & kl_4 + kl_5 + \mathscr{b}_2 \cdot \lambda \cdot (\lfloor \log(2k'n_2 \cdot (2^\iota - 1)/\mathscr{b}_2) \rfloor + 1) \\ & + \mathscr{b}_3 \cdot \lambda \cdot (\lfloor \log(k'n_2 \cdot (2^\iota - 1)/\mathscr{b}_3) \rfloor + 1) + \\ & m_3 + m_1m_3 + km_1 + n_3 + \mathscr{b}_4 \cdot \lambda \cdot (\lfloor \log(8m_2 \cdot \beta/\mathscr{b}_4) \rfloor + 1) + \\ & kl_3 + 2kl_1 + \mathscr{b}_5 \cdot \lambda \cdot (\lfloor \log(k'n_1n_2 \cdot (2^\iota - 1)/\mathscr{b}_5) \rfloor + 1) + \\ & L + 2l_1L + 2l_2L\end{aligned}$$

Then the proof contains

$$\|\pi\| = (\log(2\hat{p} + 1) + \hat{\kappa} + (3\hat{l}_1 + 2\hat{l}_2 + 2\mathfrak{n}_2 + 2\mathfrak{l}_2) \cdot \log q) \cdot \hat{N} + (\hat{l}_1 + \mathfrak{n}_2) \cdot \log q$$

bits.

In summary, in each executation of the spend protocol, the communication cost is

$$\mathfrak{C}_S = (\log(2\hat{p} + 1) + \hat{\kappa} + (3\hat{l}_1 + 2\hat{l}_2 + 2\mathfrak{n}_2 + 2\mathfrak{l}_2) \cdot \log q) \cdot \hat{N} + (\hat{l}_1 + \mathfrak{n}_2) \cdot \log q + n_1^2\lceil \log p \rceil + 2n_1\lceil \log p \rceil \quad (8)$$

Choosing the Parameters. Now, we are ready to estimate the concrete communication cost of our electronic cash system via setting concrete values of parameters involved in Equation (7) and these involved in Equation (8). Note that security of the E-cash system relies on the following 11 assumptions:

$$\begin{cases} SIS_{\hat{l}_1,q,\beta_1};SIS_{\hat{l}_1,q,\beta_2} \\ LWE_{\hat{l}_2,q,\alpha} \\ \\ SIS_{m_1,q,\beta_3} \\ \\ SIS_{n_1,p,\beta_4} \\ \\ LWR_{n_2,q,p} \\ \\ SIS_{l_1,q,\beta_5} \\ \\ SIS_{2l_1,q,\beta_6};SIS_{l_3,q,\beta_7} \\ SIS_{l_4,q,\beta_8};SIS_{l_5,q,\beta_9} \end{cases}$$

where $\beta_1 = 16\hat{p}\cdot\sqrt{\hat{l}_1+\hat{l}_2+\mathfrak{n}_2}\cdot(\hat{\sigma}_2+\hat{p}\cdot\hat{\sigma}_1)$, $\beta_2 = 16\hat{p}\cdot\sqrt{\hat{l}_1+\hat{l}_2+\mathfrak{l}_2}\cdot(\hat{\sigma}_2 + \hat{p}\cdot\hat{\sigma}_1)$, $\beta_3 \approx \sigma^2\cdot m_2\cdot\sqrt{m_2}\cdot m_3\cdot(4m_2/\mathscr{b}_4)$, $\beta_4 = \sqrt{n_3}$, $\beta_5 = \sqrt{l_2}$, $\beta_6 = \sqrt{kl_3}$, $\beta_7 = \sqrt{k'n_1n_2}\cdot k'n_1n_2\cdot(2^\iota-1)/\mathscr{b}_5$, $\beta_8 = \sqrt{2k'n_2}\cdot 2k'n_2\cdot(2^\iota-1)/\mathscr{b}_2$, $\beta_9 = \sqrt{k'n_2}\cdot k'n_2\cdot(2^\iota-1)/\mathscr{b}_3$, $\alpha = \frac{\sqrt{2\pi}\cdot\hat{\sigma}_1}{q}$. Thus, we should guaratee that the above assumptions are valid while picking parameters. As discussed in Sec. B.1, this can be achieved by setting the parameters in a way that a small root-Hermite factor is required to break the assumptions.

More concretely, we set these parameters as per Table 6. Our parameters are set to achieve 80-bit security, 100-bit security and 128-bit security respectively in a setting that we fix $L = 10$ (i.e., each new wallet in our E-cash system contains 2^{10} coins) and $\hat{\kappa} = 128$.

Communication Cost of [55]. Here, we give a rough estimation of the concrete communication cost for the spend protocol of the E-cash system constructed in [55]. We omit several less significant parts in the estimation and thus the real communication cost will be larger than we have estimated here.

Let n_1, n_2, p, q, q_s be positive integers. Let $k_q = \lceil\log q\rceil$, $k_s = \lceil\log q_s\rceil$, $\sigma = 1.6k_s\cdot\sqrt{n_1}$ and $k_\sigma = \lceil\log(12\sigma)\rceil$. Then the communication cost of their spend protocol is

$$\mathfrak{C}_S \approx (960\cdot n_1\cdot k_s^2\cdot k_\sigma + 128\cdot n_2\cdot k_q^3)\cdot 137$$

bits and security of their system relies on the following two assumptions:

$$\begin{cases} SIS_{n_1,q_s,\beta}; \\ LWE_{n_2,q,\alpha} \end{cases}$$

Table 6: Concrete Parameters for Our Electronic Cash.

Soundness Error	2^{-80}	2^{-100}	2^{-128}
δ_0	1.0048	1.0042	1.0035
$\hat{p}$	2^{80}	2^{100}	2^{128}
$\hat{l}_1$	6700	9300	13700
$\hat{l}_2$	7200	9900	14900
$\hat{N}$	1	1	1
$\lceil \log p \rceil$	105	125	155
$\lceil \log q \rceil$	210	250	310
m_1	1050	1050	1050
n_1	9	9	9
n_2	2000	2700	4000
l_1	6	6	6
l_3	240	240	250
l_4	200	200	210
l_5	180	180	200
ι	10	10	10
$(\mathscr{b}_1, \mathscr{b}_2, \mathscr{b}_3, \mathscr{b}_4, \mathscr{b}_5,)$	$(4, 2, 2, 16, 2)$	$(4, 2, 2, 16, 2)$	$(4, 2, 2, 16, 2)$
$\mathfrak{C}_W$	53MB	78MB	130MB
$\mathfrak{C}_S$	262MB	394MB	671MB

where $\beta = \sigma^2 \cdot (2 \cdot k_s \cdot n_1)^{1.5} \cdot 80$ and $\alpha = 1/(2^{80} \cdot p \cdot (n_2 \cdot k_q)^{10})$.

To achieve a 80-bits security level, we can set $n_1 = 2000$, $n_2 = 11000$, $p = 2^8$, $q = 2^{310}$, $q_s = 2^{60}$, and the communication cost $\mathfrak{C}_S \approx 720$ TB.

Even removing the PRF parts in their arguments [11], the communication cost for their spend protocol can still reach 1.85 TB, which is more than 1000 times larger than the communication cost for the spend protocol of our E-cash system. Here, the efficiency improvement comes from two parts: 1) our `ZKAoK` is one-shot; and 2) our reduction from high-level lattice-based relations to the basic relation $\mathcal{R}^*$ is tighter.

G Range Proof

The Goal. In a range proof, the prover aims to prove to the verifier that 1) he knows the committed value of a given commitment; and 2) the committed

[11] Recall that in our construction, we use a weak PRF and an accumulator to replace the PRF.

value is in a given range. More precisely, let Com be the commitment scheme, the prover aruges the following relation:

$$\mathcal{R} = \{(A, B, c), (W, r) : A \le W \le B \ \wedge c = \mathsf{Com}(W; r)\}$$

Overview. For a public range $[A, B]$, we prove a number $W \in [A, B]$ by showing that there exists non-negative numbers U and V satisfying $W - U = A$ and $W + V = B$. This is exactly the strategy employed in the range proof proposed in [56], however, we design the ZKAoK for large integer additions in a different manner. In particular, for our setting that q is large, we use a more simple and efficient approach to argue the correctness of carriers generated in the addition.

The Underlying ZKAoK for Large Integer Additions. Let L be a large positive integer. Our goal is to prove knowledge of two L-bits integers W and U satisfying

$$W + U = A \tag{9}$$

for a public integer $A \in [0, 2^L - 1]$. Here, the integers W, U and A are represented by L-dimension binary vectors $\boldsymbol{w}$, $\boldsymbol{u}$ and $\boldsymbol{a}$ respectively, i.e.,

$$A = \sum_{i=1}^{L} 2^{i-1}\boldsymbol{a}[i], \quad W = \sum_{i=1}^{L} 2^{i-1} \cdot \boldsymbol{w}[i], \quad U = \sum_{i=1}^{L} 2^{i-1}\boldsymbol{u}[i]$$

We prove this via reducing the relation to

$$\mathcal{R}^* = \{(\boldsymbol{A}, \boldsymbol{y}, \mathcal{M}), (\boldsymbol{x}) : \boldsymbol{A}\cdot\boldsymbol{x} = \boldsymbol{y} \mod q \ \wedge \forall (h, i, j) \in \mathcal{M}, \boldsymbol{x}[h] = \boldsymbol{x}[i]\cdot\boldsymbol{x}[j]\}$$

First, let k' be a postive integer that $2^{k'+1} \le q$ and let $l = \lceil L/k' \rceil$. Then we define $\bar{a}_i = \sum_{j=1}^{k'} 2^{j-1}\boldsymbol{a}[(i-1)\cdot k' + j]$ for $i \in [1, l]$ ($\boldsymbol{a}[h]$ is regarded as 0 if $h > L$) and $\bar{\boldsymbol{a}} = (\bar{a}_1 \| \bar{a}_2 \| \ldots \| \bar{a}_l)^\intercal$. We also define $\bar{\boldsymbol{w}}_i = (\boldsymbol{w}[(i-1)\cdot k' + 1] \| \boldsymbol{w}[(i-1)\cdot k' + 2] \| \ldots \| \boldsymbol{w}[(i-1)\cdot k' + k'])^\intercal$ and $\bar{\boldsymbol{u}}_i = (\boldsymbol{u}[(i-1)\cdot k' + 1] \| \boldsymbol{u}[(i-1)\cdot k' + 2] \| \ldots \| \boldsymbol{u}[(i-1)\cdot k' + k'])^\intercal$ for $i \in [1, l]$.

Next, we transform Equation (9) into the following equations:

$$\begin{cases} \boldsymbol{g}\cdot\bar{\boldsymbol{w}}_1 + \boldsymbol{g}\cdot\bar{\boldsymbol{u}}_1 - 2^{k'}\cdot c_1 = \bar{a}_1 \\ \forall i \in [2, l-1], \boldsymbol{g}\cdot\bar{\boldsymbol{w}}_i + \boldsymbol{g}\cdot\bar{\boldsymbol{u}}_i + c_{i-1} - 2^{k'}\cdot c_i = \bar{a}_i \\ \boldsymbol{g}\cdot\bar{\boldsymbol{w}}_l + \boldsymbol{g}\cdot\bar{\boldsymbol{u}}_l + c_{l-1} = \bar{a}_l \end{cases} \tag{10}$$

where $\boldsymbol{g} = (1 \quad 2 \quad 4 \ldots 2^{k'-1})$ is a row vector and for $i \in [1, l]$, c_i is the carrier for the i-th addition.

Equation (10) is just a set of linear equations with binary solutions and we can reduce it to $\mathcal{R}^*$ via setting:

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{G} & \boldsymbol{G} & \boldsymbol{M} \end{pmatrix}, \quad \boldsymbol{x} = \begin{pmatrix} \boldsymbol{w} \\ \boldsymbol{u} \\ \boldsymbol{c} \end{pmatrix}, \quad \boldsymbol{y} = \bar{\boldsymbol{a}}$$

where $\boldsymbol{G} = \boldsymbol{I}_l \otimes \boldsymbol{g}$, $\boldsymbol{c} = (c_1 \| c_2 \| \dots \| c_{l-1})^\intercal$, and

$$\boldsymbol{M} = \begin{pmatrix} -2^{k'} & & & & \\ 1 & -2^{k'} & & & \\ & \ddots & \ddots & & \\ & & 1 & -2^{k'} \\ & & & 1 \end{pmatrix}$$

Finally, we define $\mathcal{M} = \{(i, i, i)\}_{i \in [2L+l]}$. In the new relation, both the length of the witness and the size of $\mathcal{M}$ are $2L + l$.

Note that the argument of knowledge of integers W, U that $W - U = A$ for a public A is identical to the above argument except that we set $\boldsymbol{A} = (\boldsymbol{G} \| -\boldsymbol{G} \| \boldsymbol{M}')$ where

$$\boldsymbol{M} = \begin{pmatrix} 2^{k'} & & & \\ -1 & 2^{k'} & & \\ & \ddots & \ddots & \\ & & -1 & 2^{k'} \\ & & & -1 \end{pmatrix}$$

The Construction of Range Proof. Let λ be the security parameter. Let n, L be postive integers that are polynomial in λ. Let q be a large enough prime number, and let $k = \lceil \log q \rceil$, $m = n(k+3)$, $k' = \lfloor \log q \rfloor - 1$, $l = \lceil L/k' \rceil$. Let $\mathsf{Com} = (\mathsf{Com}.\mathtt{Commit}, \mathsf{Com}.\mathtt{Open})$ be the commitment scheme recalled in Sec. A.1. Let $\boldsymbol{B}_1 \in \mathbb{Z}_q^{n \times m}$ and $\boldsymbol{B}_2 \in \mathbb{Z}_q^{n \times L}$ be the public parameter for Com.

The range proof works as follows:

- **Prove.** The prover's inputs include three L-bits integers W, A, B, a vector $\boldsymbol{r} \in [0,1]^n$ and a commitment $\boldsymbol{c} = \boldsymbol{B}_1 \cdot \boldsymbol{r} + \boldsymbol{B}_2 \cdot \boldsymbol{w}$, where $\boldsymbol{w} = (w_1 \| \ w_2 \| \dots \| w_{L-1})^\intercal \in \{0,1\}^L$ is the bit decomposition of W (i.e., $W = \sum_{i=1}^{L} 2^{i-1} \cdot w_i$). First, the prover computes $U = W - A$ and $V = B - W$. Then he generates the proof:

$$\begin{aligned} \pi = SPK\{&(\boldsymbol{B}_1, \boldsymbol{B}_2, \boldsymbol{c}, A, B), (U, V, W, \boldsymbol{w}, \boldsymbol{r}) \\ & W - U = A \ \wedge \ W + V = B \ \wedge \ \boldsymbol{c} = \boldsymbol{B}_1 \cdot \boldsymbol{r} + \boldsymbol{B}_2 \cdot \boldsymbol{w}\} \end{aligned}$$

  Note that here all L-bits strings are represented as binary vectors, thus the two witnesses W and $\boldsymbol{w}$ are the same.
  The output of the Prove algorithm is π.
- **Verify.** On input a commitment $\boldsymbol{c}$, two L-bits integers A and B and a proof π, the verifier outputs 1 iff the proof π is valid.

It is easy to verify that the range proof above is a secure one if the underlying arguments are secure.

The Efficiency of Range Proof. We focus on the communication cost, namely, the proof size, of the range proof.

The proved statement includes three parts. The first two parts argues the integers addition relation. As analyzed above, each part can be reduced to an instance of the relation $\mathcal{R}^*$, where the length of its witness and the size of $\mathcal{M}$ are both $2L + l$. The last part argues knowledge of the committed value. As analyzed in Sec. 4.1, this part can be reduced to an instance of the relation $\mathcal{R}^*$, where the length of its witness and the size of $\mathcal{M}$ are both $m + L$. So, after combing these three statements, the length of the witness will be

$$\mathfrak{n} = 3L + 2l + m$$

and the size of $\mathcal{M}$ will also be

$$\mathfrak{l} = 3L + 2l + m$$

(the overlapped part, namely, $\boldsymbol{w}$, should be counted only once).

Now, let $\hat{l}_1, \hat{l}_2$, $\hat{\sigma}_1$, $\hat{\sigma}_2$, $\hat{p}$, $\hat{\kappa}$, $\hat{N}$ be parameters used in the main protocol. Then the proof contains

$$\|\pi\| = (\log(2\hat{p}+1) + \hat{\kappa} + (3\hat{l}_1 + 2\hat{l}_2 + 2\mathfrak{n} + 2\mathfrak{l}) \cdot k) \cdot \hat{N} + (\hat{l}_1 + \mathfrak{n}) \cdot k \quad (11)$$

bits.

Next, we estimate the concrete size of the proof via setting concrete values of parameters involved in Equation (11). Note that security of the range proof relies on the following 4 assumptions:

$$\begin{cases} SIS_{\hat{l}_1,q,\beta_1} \\ SIS_{\hat{l}_1,q,\beta_2} \\ LWE_{\hat{l}_2,q,\alpha} \\ SIS_{n,q,\beta_3} \end{cases}$$

where $\beta_1 = 16\hat{p} \cdot \sqrt{\hat{l}_1 + \hat{l}_2 + \mathfrak{n}} \cdot (\hat{\sigma}_2 + \hat{p} \cdot \hat{\sigma}_1)$, $\beta_2 = 16\hat{p} \cdot \sqrt{\hat{l}_1 + \hat{l}_2 + \mathfrak{l}} \cdot (\hat{\sigma}_2 + \hat{p} \cdot \hat{\sigma}_1)$, $\beta_3 = \sqrt{m + L}$, $\alpha = \frac{\sqrt{2\pi} \cdot \hat{\sigma}_1}{q}$. Thus, we should guaratee that the above assumptions are valid while picking parameters. As discussed in Sec. B.1, this can be achieved by setting the parameters in a way that a small root-Hermite factor is required to break the assumptions.

More concretely, we set these parameters as per Table 7. Our parameters are set to achieve 80-bit security, 100-bit security and 128-bit security respectively in a setting that we fix $L = 1000$ (i.e., we will deal with 1000 bits integers in the argument) and $\hat{\kappa} = 128$.

Table 7: Concrete Parameters for Our Range Proof.

Soundness Error	2^{-80}	2^{-100}	2^{-128}
δ_0	1.0048	1.0042	1.0035
$\hat{p}$	2^{20}	2^{20}	2^{32}
$\hat{l}_1$	2200	2500	4200
$\hat{l}_2$	2600	3000	4800
$\hat{N}$	4	5	4
$\lceil \log q \rceil$	80	80	105
n	14	16	16
Proof Size	1.21MB	1.61MB	2.28MB