A Lattice-based Designated Verifier zkSNARK from Standard Assumptions

Mohammad Sadegh Ahmadi[1], Taraneh Eghlidos[1], Behzad Abdolmaleki[2] and Ngoc Khanh Nguyen[3]

[1] Sharif University of Technology
[2] University of Sheffield
[3] King's College London

Abstract. Designated Verifier zero-knowledge Succinct Non-Interactive Arguments of Knowledge (DV-zkSNARKs) are cryptographic argument systems in which the ability to verify proofs is restricted to a designated verifier. Unlike publicly verifiable zkSNARKs, these constructions ensure that only an authorized party can validate the correctness of the proof. Existing lattice-based DV-zkSNARK constructions typically rely either on linear-only encryption or on the Linear Targeted Malleability (LTM) assumption. The former has been cryptanalytically broken and therefore no longer provides a credible basis for post-quantum security. The latter is a non-standard and comparatively less established assumption and, moreover, restricts knowledge soundness to the non-adaptive setting. To overcome these limitations, we propose an inner-product argument system whose security relies solely on the well-established hardness of the Module Short Integer Solution (MSIS) problem and that achieves adaptive knowledge soundness in the random oracle model. This construction enables a designated verifier, holding a secret key, to succinctly verify inner product of a committed witness with an arbitrary vector. By combining our argument system with a linear probabilistic checkable proof (LPCP) compiler, to the best of our knowledge, we obtain the first DV-zkSNARK construction based on standard assumptions. Our implementation achieves prover and verification times comparable to the state of the art, while reducing public parameter size by a factor of 10, at the cost of a 2.5x increase in proof size.

Keywords: DV-zkSNARK · Linear targeted malleability · Adaptive soundness · Lattice-based cryptography

1 Introduction

Succinct Non-Interactive Arguments of knowledge (SNARKs) are fundamental cryptographic primitives that enable provers to efficiently convince verifiers of the validity of computational statements using proofs significantly smaller than the underlying computations [Kil92]. Over the past decade, SNARKs have seen rapid theoretical advancements and widespread practical adoption, particularly in blockchain technologies and verifiable computation frameworks [DMS24, Lip24, Gro16, Set20, CHM+20]. While most SNARK constructions are publicly verifiable, there exist settings in which the prover intends to convince only an authorized verifier. Such schemes, referred to as Designated Verifier zkSNARKs (DV-zkSNARKs), have been studied in the context of verifiable computation [ABVPS24], indistinguishability obfuscation [BISW17], and anonymous credentials [ADI25].

E-mail: mohammadsadegh.ahmadi@ee.sharif.edu (Mohammad Sadegh Ahmadi), teghlidos@sharif.edu (Taraneh Eghlidos), behzad.abdolmaleki@sheffield.ac.uk (Behzad Abdolmaleki), ngoc_khanh.nguyen@kcl.ac.uk (Ngoc Khanh Nguyen)

The anticipated threat of quantum computing has spurred the development of post-quantum secure SNARKs, with lattice-based approaches being particularly attractive due to their strong theoretical foundations and presumed resistance to quantum attacks. The first publicly verifiable lattice-based SNARK was proposed by Albrecht *et al.* [ACL+22]. However, as this scheme relied on non-standard assumptions, Wee and Wu [WW23] demonstrated attacks exploiting structural weaknesses in existing lattice-based SNARKs. LaBRADOR [BS23] is another lattice-based argument system that is publicly verifiable but suffers from linear-time verification. Several works have explored constructing SNARKs by combining polynomial commitment schemes with PIOP compilers [AFLN24, HSS24, NS24], but the resulting large parameter sizes lead to inefficiency. In contrast, DV-SNARKs often achieve better efficiency. Bitansky *et al.* [BCI+13] introduced a generic compiler for constructing SNARGs in the preprocessing model, based on the linear-only encryption (LOE) assumption. Their framework builds on a linear probabilistically checkable proof (LPCP) [IKO07] compiler, which provides succinct verification for Rank-1 Constraint System (R1CS) circuits, and employs homomorphic encryption over the encrypted challenges to compute the proof, allowing any verifier holding the secret key to verify it. The LOE assumption guarantees that an adversary with access to encrypted challenges can only form valid linear combinations, ensuring soundness. The first lattice-based SNARG in this framework was introduced by Boneh *et al.* [BISW17], using LWE-based homomorphic encryption to instantiate the LOE property. Subsequent works extended this approach to support stronger guarantees such as zero-knowledge, knowledge soundness, and improved efficiency [GMNO18, ISW21, Nit19, CKKK23, NYI+20]. However, recent results by Debris *et al.* [DAFS24] demonstrated that the LOE assumption fails against quantum adversaries, undermining the post-quantum security of these constructions. To address this, Steinfeld *et al.* [SSE+24] proposed replacing LOE with a weaker assumption called *linear targeted malleability* (LTM), which underlies the current state-of-the-art lattice-based DV-SNARK. Nevertheless, this assumption is not yet well established. Moreover, Bitansky *et al.* [BCI+13] showed that constructions based on the LTM assumption cannot achieve adaptive soundness, requiring the prover to commit to the witness before observing the CRS which is a strictly weaker security notion. Given this context, the open challenge is to devise a designated verifier lattice-based SNARK that simultaneously offer post-quantum security, relying on established lattice assumptions, while maintaining efficient verification processes.

1.1 Our Contribution

Our contributions are summarized as follows.

Trapdoor-based Ajtai Commitment. We introduce a variant of the Ajtai commitment scheme equipped with a trapdoor that enables efficient verification of linear operations on the committed witness.

Zero-Knowledge Lattice-based Inner-Product Argument. Building on the proposed commitment scheme, we construct a designated-verifier inner-product argument with zero-knowledge guarantees. The resulting protocol is of independent interest and can serve as a building block for applications such as secure two-party computation and privacy-preserving AI.

A Lattice-based DV-zkSNARK. Using the proposed inner-product argument, we obtain a DV-zkSNARK from the LPCP compiler under the Module-SIS assumption. Unlike previous lattice-based DV-zkSNARKs, our construction does not rely on the LTM assumption and achieves knowledge soundness in the random oracle model, thereby providing adaptive soundness. To the best of our knowledge, this is the first lattice-based DV-zkSNARK based solely on standard assumptions. We also provide a proof-of-concept implementation and an experimental evaluation demonstrating the practical feasibility of

Table 1: Asymptotic comparison of recent lattice-based argument systems. N is length of the witness.

Scheme	Assumption	Verifier Complexity	Proof Size	Commitment Size	Publicly Verifiable
LaBRADOR	MSIS	$\mathcal{O}(N)$	$\mathcal{O}(\log\log N)$	$\mathcal{O}(1)$	✓
LUNA	LTM	$\mathcal{O}(1)$	$\mathcal{O}(1)$	–	✗
Our Scheme	MSIS	$\mathcal{O}(\sqrt{N})$	$\mathcal{O}(1)$	$\mathcal{O}(\sqrt{N})$	✗

the construction. The source code is publicly available on GitHub[1].

An overall asymptotic comparison between our construction and recent lattice-based schemes is given in Table 1. The comparison highlights that our construction occupies an intermediate point between publicly verifiable schemes and designated-verifier schemes. From a security perspective, our scheme relies on the standard and well-studied MSIS assumption, which makes its assumption profile closer to publicly verifiable constructions and more conservative than recent designated-verifier approaches based on less established assumptions. In terms of efficiency, our construction is slightly less efficient than designated-verifier schemes with constant-size proofs and constant-time verification, but it still achieves sublinear verifier complexity and compact proofs.

1.2 Technical Overview

We provide a high-level overview of our construction and the techniques underlying our approach.

A Lattice-based Inner-Product Argument: We introduce a type of inner-product argument in which the prover first commits to a witness vector $\vec{\mathbf{b}}$ along with a noise vector $\vec{\boldsymbol{\mu}}$,to provide hiding, using a revised version of the Ajtai commitment scheme. The verifier then samples and sends a uniformly random challenge vector $\vec{\mathbf{u}}_{\mathsf{chall}}$. The prover computes the inner product of the witness with this challenge and provides a corresponding proof. In other words, our goal is to construct an argument system for the following relation:

$$\mathsf{R}_{\mathsf{IPA}} = \left\{ \begin{array}{l} \vec{\mathbf{c}} = \begin{bmatrix} \mathbf{A}_{\mathsf{com}} & \mathbf{A}_{\mathsf{hide}} \end{bmatrix} \vec{\mathbf{w}} \\ \wedge\ \mathbf{h} = \langle \vec{\mathbf{w}}, \vec{\mathbf{u}}_{\mathsf{chall}} \rangle \\ \wedge\ \|\vec{\mathbf{w}}\|_\infty < \beta_{\mathsf{open}} \end{array} \middle| \begin{array}{l} \mathsf{pp} : (\mathbf{A}_{\mathsf{com}}, \mathbf{A}_{\mathsf{hide}}) \in \mathcal{R}_q^{n\times m} \times \mathcal{R}_q^{n\times m} \\ \mathsf{vk} : (\vec{\mathbf{p}}, \mathbf{z}_{\mathsf{sec}}) \in \mathcal{R}_q^{n} \times \mathcal{R}_q \\ \varkappa : (\vec{\mathbf{c}}, \vec{\mathbf{u}}_{\mathsf{chall}}, \mathbf{h}) \in \mathcal{R}_q^{n} \times \mathcal{R}_q^{2m} \times \mathcal{R}_q \\ \wp : \vec{\mathbf{w}} := \begin{bmatrix} \vec{\mathbf{b}} \\ \vec{\boldsymbol{\mu}} \end{bmatrix} \end{array} \right\}. \tag{1}$$

We carefully modify the commitment matrices $\mathbf{A}_{\mathsf{com}}$ and $\mathbf{A}_{\mathrm{hide}}$, constructing them using submatrices that possess trapdoors (see Section 3). This modification enables a verifier holding these trapdoors, along with a low-norm secret key $\vec{\mathbf{p}}$, to compute a trapdoor-dependent vector $\vec{\mathbf{z}}_{\mathsf{G}}$ for any uniformly sampled challenge $\vec{\mathbf{u}}_{\mathsf{chall}}$. This procedure is described in the Algorithm 5 known as ChallTrap. As shown in Claim 2, the inner product of the witness with $\vec{\mathbf{u}}_{\mathsf{chall}}$ reduces to verifying the following relation:

$$\mathbf{h} = \langle \vec{\mathbf{w}}, \vec{\mathbf{u}}_{\mathsf{chall}} \rangle = \langle \vec{\mathbf{p}}, \vec{\mathbf{c}} \rangle + \langle \vec{\mathbf{z}}_{\mathsf{G}}, \vec{\mathbf{w}} \rangle. \tag{2}$$

The verifier can compute $\langle \vec{\mathbf{p}}, \vec{\mathbf{c}} \rangle$, as they know both $\vec{\mathbf{p}}$ and $\vec{\mathbf{c}}$. However, they still need $\langle \vec{\mathbf{z}}_{\mathsf{G}}, \vec{\mathbf{w}} \rangle$. A naive approach would be for the verifier to send $\vec{\mathbf{z}}_{\mathsf{G}}$ to the prover, who returns $\mathbf{h}' = \langle \vec{\mathbf{z}}_{\mathsf{G}}, \vec{\mathbf{w}} \rangle$. The verifier could then check if $\mathbf{h} = \langle \vec{\mathbf{p}}, \vec{\mathbf{c}} \rangle + \mathbf{h}'$. However, a malicious prover could add a fixed offset $\mathbf{h}''$ to their claimed responses for both challenges and still pass verification, thereby breaking soundness. To overcome this, we adopt a masking technique: embedding $\vec{\mathbf{z}}_{\mathsf{G}}$ into a statistically uniform-looking vector using the

[1] https://github.com/crypolover1998/LatticeBased-Dv-zkSNARK

Gaussian2Uniform algorithm (Algorithm 6). This algorithm takes $\vec{\mathbf{u}}_{\mathsf{chall}}$, $\vec{\mathbf{z}}_{\mathsf{G}}$, and a Gaussian-sampled secret $\mathbf{z}_{\mathsf{sec}}$ (matching the distribution of $\vec{\mathbf{z}}_{\mathsf{G}}$) to produce $\vec{\mathbf{u}}_{\mathsf{aux}} = \vec{\mathbf{z}}_{\mathsf{G}} + \mathbf{z}_{\mathsf{sec}}\vec{\mathbf{u}}_{\mathsf{chall}}$. If the conditions of Lemma 4 hold, this vector is statistically close to uniform. Then, the prover's response to this challenge is $\mathbf{h}_z = \langle \vec{\mathbf{w}}, \vec{\mathbf{z}}_{\mathsf{G}} \rangle + \mathbf{z}_{\mathsf{sec}}\mathbf{h}$. Adding $\langle \vec{\mathbf{p}}, \vec{\mathbf{c}} \rangle$ to both sides yields:

$$\mathbf{h}_z + \langle \vec{\mathbf{p}}, \vec{\mathbf{c}} \rangle = (\mathbf{z}_{\mathsf{sec}} + 1)\mathbf{h}. \tag{3}$$

The verifier can now easily check whether Equation (3) holds. Importantly, the prover cannot cheat in this process, as they do not know the value of $\mathbf{z}_{\mathsf{sec}}$.

Extending to Arbitrary Challenges: A verifier can also verify the inner product with an arbitrary vector $\vec{\mathbf{u}}_\chi$. Using ChallTrap and Gaussian2Uniform with a new secret $\mathbf{z}_{\mathsf{sec},\chi}$, the verifier constructs a uniform-looking challenge:

$$\vec{\mathbf{u}}_{\mathsf{chall},\chi} = \vec{\mathbf{z}}_{\mathsf{G},\chi} + \mathbf{z}_{\mathsf{sec},\chi}\vec{\mathbf{u}}_{\mathsf{chall}}.$$

The procedure for computing such a challenge is illustrated in Figure 1.

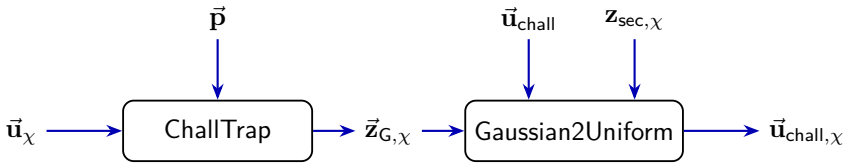


Figure 1: Converting a desired challenge vector to a uniform one.

The prover's response to this challenge is:

$$\mathbf{h}_\chi = \langle \vec{\mathbf{w}}, \vec{\mathbf{u}}_{\mathsf{chall},\chi} \rangle = \mathbf{z}_{\mathsf{sec},\chi}\mathbf{h} + \langle \vec{\mathbf{w}}, \vec{\mathbf{z}}_{\mathsf{G},\chi} \rangle.$$

Subtracting $\mathbf{z}_{\mathsf{sec},\chi}\mathbf{h}$ from $\mathbf{h}_\chi$ and adding $\langle \vec{\mathbf{p}}, \vec{\mathbf{c}} \rangle$ to it recovers $\langle \vec{\mathbf{p}}, \vec{\mathbf{c}} \rangle + \langle \vec{\mathbf{w}}, \vec{\mathbf{z}}_{\mathsf{G},\chi} \rangle$, which is equal to $\langle \vec{\mathbf{w}}, \vec{\mathbf{u}}_\chi \rangle$ according to Equation (2).

Hence, the verifier could ask about inner product of any desired vector. Note that the verifer also needs to verify that the inner product of witness with $\vec{\mathbf{u}}_{\mathsf{chall},\chi}$ is computed correctly. They can adopt the same approach which they have used for verifying the inner product of witness with $\vec{\mathbf{u}}_{\mathsf{chall}}$ since the $\vec{\mathbf{u}}_{\mathsf{chall},\chi}$ is also a uniform challenge.

2 Preliminaries

2.1 Notation

We denote the security parameter by λ, and use $\mathsf{negl}(\cdot)$ to denote negligible function. For integers $a, b \in \mathbb{N}$ with $a < b$, we define

$$[a, b] := \{a, a+1, \ldots, b\}, \qquad [a] := [1, a].$$

Let d be a power of two. We define the cyclotomic ring as $\mathcal{R} = \mathbb{Z}[X]/(X^d + 1)$, and the corresponding residue ring modulo q as $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^d + 1)$. Elements of $\mathcal{R}_q$ are denoted by bold lowercase letters, e.g., $\mathbf{f}$. Vectors over $\mathcal{R}_q$ are denoted by bold lowercase letters with arrows, such as $\vec{\mathbf{f}}$, while matrices over these rings are denoted by bold uppercase letters. Vectors over fields such as $\mathbb{Z}_q$, where q is prime, are denoted by lowercase letters with arrows, e.g., $\vec{f}$.

For a vector $\vec{\mathbf{f}} \in \mathcal{R}_q^n$, we denote the subvector consisting of entries indexed from i to j by $[\vec{\mathbf{f}}]_i^j$. For a matrix $\mathbf{A}$, the notation $\mathbf{A}[i, j]$ denotes the entry in the i-th row and j-th column. Similarly, $\mathbf{A}[i, :]$ denotes the i-th row of $\mathbf{A}$, and $\mathbf{A}[:, j]$ denotes the j-th column.

For an element $\mathbf{w} = w_0 + w_1 X + \cdots + w_{d-1} X^{d-1} \in \mathcal{R}_q$, we define its coefficient vector representation as

$$\mathsf{cf}(\mathbf{w}) = [w_0, w_1, \ldots, w_{d-1}]^\top \in \mathbb{Z}_q^d.$$

The ℓ_p-norm of $\mathbf{w}$, for $p \in \{1, 2, \infty\}$, is defined as the ℓ_p-norm of its coefficient vector $\mathsf{cf}(\mathbf{w})$. We further define the rotation matrix associated with $\mathbf{w}$ as

$$\mathsf{Rot}(\mathbf{w}) = \left[\mathsf{cf}(\mathbf{w}) \mid \mathsf{cf}(X\mathbf{w}) \mid \cdots \mid \mathsf{cf}(X^{d-1}\mathbf{w})\right] \in \mathbb{Z}_q^{d \times d}.$$

For any $\mathbf{w}_1, \mathbf{w}_2 \in \mathcal{R}_q$, it is straightforward to verify that polynomial multiplication can be expressed as

$$\mathbf{w}_1 \cdot \mathbf{w}_2 = \mathsf{Rot}(\mathbf{w}_1)\mathsf{cf}(\mathbf{w}_2).$$

Lemma 1. [BTT22] *Let $q \equiv 5 \pmod 8$ be a prime integer, and let $m \geq n \geq 1$. Then, for a uniformly random matrix $\mathbf{A} \in \mathcal{R}_q^{n \times m}$, the probability that $\mathbf{A}$ is not full-rank is negligible.*

Lemma 2 (**Generalized Schwartz–Zippel** [BCPS18])**.** *Let $f \in \mathcal{R}_q^{\leq n}[X_1, X_2, \ldots, X_\mu]$ be a nonzero μ-variate polynomial over the ring $\mathcal{R}_q$, where the degree of f in each variable is at most n. Let $\mathcal{C} \subseteq \mathcal{R}_q$ be a sampling set. Then,*

$$\Pr\left[f(\vec{\mathbf{r}}) = 0 \;\middle|\; \vec{\mathbf{r}} \xleftarrow{\$} \mathcal{C}^\mu\right] \leq \frac{n\mu}{|\mathcal{C}|}.$$

Lemma 3. [BL17] *Let $\mathbf{M} \in \mathbb{Z}_p^{d \times k}$ be an arbitrary matrix, and let $\mathbf{C} \xleftarrow{\$} \{0,1\}^{k \times \lambda}$. Define $\|\mathbf{M}\|_\infty = \max_{i,j} |m_{i,j}|$. Then,*

$$\Pr\left[\|\mathbf{MC} \mod p\|_\infty < \frac{1}{2}\|\mathbf{M}\|_\infty\right] \leq 2^{-\lambda}.$$

Lemma 4 (**Regularity Lemma** [LPR13])**.** *Let $\mathcal{R}_q$ be a cyclotomic ring of degree d, and let $q \geq 2$ be an integer. For positive integers $k \leq l \leq \mathsf{poly}(d)$, define*

$$\mathbf{A} = \left[\mathbf{I}_k \mid \bar{\mathbf{A}}\right] \in \mathcal{R}_q^{k \times l},$$

where $\mathbf{I}_k$ denotes the identity matrix and $\bar{\mathbf{A}} \in \mathcal{R}_q^{k \times (l-k)}$ is sampled uniformly at random. Let $\vec{\mathbf{x}} \in \mathcal{R}_q^l$ be a vector whose coordinates are independently sampled from a discrete Gaussian distribution with parameter $\sigma > 2d \cdot q^{\frac{k}{l} + \frac{2}{dl}}$. Then, the distribution of $\mathbf{A}\vec{\mathbf{x}}$ is statistically close to the uniform distribution over $\mathcal{R}_q^k$.

Lemma 5 (**Modified Regularity Lemma** [FMN24])**.** *Let $q \equiv 5 \pmod 8$ be a prime integer, and let $m \geq n$. Assume $\sigma > 2d \cdot q^{\frac{n}{m} + \frac{2}{dm}}$. Then the following two distributions are statistically close:*

$$\left\{(\mathbf{A}, \mathbf{A}\vec{\mathbf{x}}) \;\middle|\; \mathbf{A} \xleftarrow{\$} \mathcal{R}_q^{n \times m},\ \vec{\mathbf{x}} \leftarrow \mathcal{D}_\sigma^m\right\},$$
$$\left\{(\mathbf{A}, \vec{\mathbf{u}}) \;\middle|\; \mathbf{A} \xleftarrow{\$} \mathcal{R}_q^{n \times m},\ \vec{\mathbf{u}} \xleftarrow{\$} \mathcal{R}_q^n\right\}.$$

2.2 Lattices

A subset $\Lambda \subseteq \mathsf{R}^m$ is called a **lattice** if: (i) $\mathbf{0} \in \Lambda$, and for any $\mathbf{x}, \mathbf{y} \in \Lambda$, we have $\mathbf{x} + \mathbf{y} \in \Lambda$; (ii) for each $\mathbf{x} \in \Lambda$, there exists an $\epsilon > 0$ such that the set $\{\mathbf{y} \in \mathsf{R}^m : \|\mathbf{x} - \mathbf{y}\| < \epsilon\} \cap \Lambda = \{\mathbf{x}\}$. We say that $\mathbf{B} \in \mathsf{R}^{n \times m}$ is a basis for Λ if its columns are linearly independent, and define the lattice as $\Lambda = \mathcal{L}(\mathbf{B}) := \{\mathbf{B} \cdot \mathbf{z} : \mathbf{z} \in \mathbb{Z}^m\}$. The span (as a vector space) of the basis of a lattice is the span of the lattice, denoted as $\mathsf{Span}(\Lambda)$.

Let $\sigma > 0$ be a parameter and Λ be an m-dimensional lattice. The discrete Gaussian distribution $\mathcal{D}_{\Lambda,\sigma,\mathbf{c}}$ over a lattice coset $\mathbf{c} + \Lambda$ is defined as follows:

$$\rho_{\sigma,\mathbf{c}}(\mathbf{z}) := \exp\left(-\pi \frac{\|\mathbf{z}-\mathbf{c}\|^2}{\sigma^2}\right), \quad \mathcal{D}_{\Lambda,\sigma,\mathbf{c}}(\mathbf{z}) := \frac{\rho_{\sigma,\mathbf{c}}(\mathbf{z})}{\sum_{\mathbf{x}\in\Lambda}\rho_{\sigma,\mathbf{c}}(\mathbf{x})}.$$

When $\mathbf{c} = \mathbf{0}$ or $\Lambda = \mathbb{Z}^m$, we omit either from the notation.

Definition 1 (**Module-SIS**)**.** Let $q = q(\lambda)$, $n = n(\lambda)$, $m = m(\lambda)$ and $\beta = \beta(\lambda)$. The $\mathrm{MSIS}_{n,m,q,\beta}$ assumption holds if for any PPT adversary $\mathcal{A}$, the following probability is negligible:

$$\Pr\left[\mathbf{A}\vec{\mathbf{z}} = 0 \wedge 0 < \|\vec{\mathbf{z}}\| \leq \beta \;\middle|\; \mathbf{A} \xleftarrow{\$} (\mathcal{R}_q^{n\times m}), \vec{\mathbf{z}} \leftarrow \mathcal{A}(\mathbf{A})\right].$$

Definition 2 (**Gadget Matrix**)**.** Let $b \geq 2$. We define $\delta := \lfloor \log_b q \rfloor + 1$, and set $\vec{\mathbf{g}}^\top = [1, b, \ldots, b^{\delta-1}] \in \mathcal{R}_q^{1\times\delta}$, and the gadget matrix $\mathbf{G}_n := \mathbf{I}_n \otimes \vec{\mathbf{g}}^\top \in \mathcal{R}_q^{n\times n\delta}$.

Lemma 6 (**Trapdoor Generation** [MP12])**.** *Let $\mathcal{D}$ be a discrete Gaussian distribution over $\mathcal{R}_q^{n\times n\delta}$ with parameter s_R, and $\mathbf{G}_n$ be the gadget matrix Then, there exists a PPT algorithm* $\mathsf{TrapGen}(n, \mathcal{D})$ *that, with overwhelming probability, returns two matrices $(\mathbf{A}, \mathbf{R}) \in \mathcal{R}_q^{n\times n(\delta+1)} \times \mathcal{R}_q^{n\times n\delta}$ such that $\mathbf{A}\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right] = \mathbf{G}_n \mod q$. Moreover, $\mathbf{A}$ is statistically close to a uniformly random matrix.*

Algorithm 1 $\mathsf{TrapGen}$ [MP12]

1: **Input**: Trapdoor number of rows n and Gaussian parameter s_R.
2: $\delta := \lfloor \log_b q \rfloor + 1$.
3: $\mathbf{B} \xleftarrow{\$} \mathcal{R}_q^{n\times n}$ and $\mathbf{R} \leftarrow \mathcal{D}_{s_R}^{n\times n\delta}$.
4: **Output** $\mathbf{A} := \begin{bmatrix}\mathbf{B} & | & \mathbf{G} - \mathbf{B}\mathbf{R}\end{bmatrix}$ and $\mathbf{R}$.

Lemma 7 (**Preimage Sampling** [MP12])**.** *There exists a PPT algorithm* $\mathsf{SamplePre}$ *that takes as input a matrix $\mathbf{A}$, its corresponding trapdoor $\mathbf{R}$, a target vector $\vec{\mathbf{u}}$, and a Gaussian parameter s_v, and outputs a low-norm vector $\vec{\mathbf{v}}$ such that $\mathbf{A}\cdot\vec{\mathbf{v}} \equiv \vec{\mathbf{u}} \mod q$.*

Algorithm 2 $\mathsf{SamplePre}$ [MP12]

1: **Input**: Matrix $\mathbf{A}$ and its corresponding trapdoor $\mathbf{R}$. Syndrome vector $\vec{\mathbf{u}}$ and Gaussian parameter of preimage s_v.
2: Sample $\vec{\mathbf{p}}$ from $\mathcal{D}^m$ with covariance matrix $s_v\mathbf{I} - s_R^2 \begin{bmatrix}\mathbf{R}\\ \mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{R}^\top & \mathbf{I}^\top\end{bmatrix}$
3: $\delta := \lfloor \log_b q \rfloor + 1$.
4: Sample spherical $\vec{\mathbf{z}} \leftarrow \mathcal{D}_{s_R}^m$ such that $\mathbf{G}\vec{\mathbf{z}} = \vec{\mathbf{u}} - \mathbf{A}\vec{\mathbf{p}}$.
5: **Output** $\vec{\mathbf{v}} = \vec{\mathbf{p}} + \begin{bmatrix}\mathbf{R}\\ \mathbf{I}\end{bmatrix}\vec{\mathbf{z}}$.

2.3 Commitment Scheme

We recall the notion of a commitment scheme. A commitment scheme consists of the following PPT algorithms:

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$: On input the security parameter λ, outputs the public parameters pp.

- $\mathsf{Commit}(\mathsf{pp}, m) \to (\mathsf{cm}, \mathsf{st})$: On input the public parameters pp and a message m, outputs a commitment cm together with a decommitment state st.
- $\mathsf{Open}(\mathsf{pp}, \mathsf{cm}, m, \mathsf{st}) \to b$: On input the public parameters pp, a commitment cm, a message m, and a decommitment state st, outputs a bit $b \in \{0,1\}$, indicating whether the opening is valid.

Definition 3 (**Binding**)**.** A commitment scheme satisfies binding if for all PPT adversary $\mathcal{A}$,

$$\Pr\left[\begin{array}{l} m \neq m' \\ \wedge \mathsf{Open}(\mathsf{pp}, \mathsf{cm}, m, \mathsf{st}) = 1 \\ \mathsf{Open}(\mathsf{pp}, \mathsf{cm}, m', \mathsf{st}') = 1 \end{array} \middle| \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathsf{cm}, m, m', \mathsf{st}, \mathsf{st}') \leftarrow \mathcal{A}(\mathsf{pp}) \end{array}\right] = \mathsf{negl}(\lambda).$$

Definition 4 (**Hiding**)**.** A commitment scheme satisfies hiding if for all PPT adversary $\mathcal{A}$,

$$\Pr\left[b = b' \middle| \begin{array}{r} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ (m_0, m_1) \leftarrow \mathcal{A}(\mathsf{pp}) \\ b \xleftarrow{\$} \{0,1\} \\ (\mathsf{cm}, \mathsf{st}) \leftarrow \mathsf{Commit}(\mathsf{pp}, m_b) \\ b' \leftarrow \mathcal{A}(\mathsf{cm}) \end{array}\right] = \frac{1}{2} + \mathsf{negl}(\lambda).$$

2.4 DV-zkSNARKs

Designated Verifier Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge (DV-zkSNARKs) are argument systems in which only a designated verifier, possessing a secret verification key, can verify the validity of a proof. A DV-zkSNARK is defined with respect to a relation R between a public statement $\mathbb{x}$ and a private witness $\mathbb{w}$. If the relation holds, then $(\mathbb{x}, \mathbb{w}) \in \mathsf{R}$.

Formally, a DV-zkSNARK scheme Π for a relation R consists of the following PPT algorithms:

- $\mathsf{Setup}(1^\lambda, \mathsf{R}) \to (\mathsf{vk}, \mathsf{crs})$: On input the security parameter λ and a relation R, outputs a verifier key vk, and a common reference string crs.
- $\mathsf{Prover}(\mathbb{x}, \mathbb{w}, \mathsf{crs}) \to \boldsymbol{\pi}$: On input a public statement $\mathbb{x}$, a witness $\mathbb{w}$, and the common reference string crs, outputs a proof $\boldsymbol{\pi}$ attesting that $(\mathbb{x}, \mathbb{w}) \in \mathsf{R}$.
- $\mathsf{Verifier}(\mathsf{vk}, \mathbb{x}, \boldsymbol{\pi}) \to b$: On input the verifier key vk, a public statement $\mathbb{x}$, and a proof $\boldsymbol{\pi}$, outputs a bit $b \in \{0,1\}$, indicating whether the proof is accepted.

A DV-zkSNARK scheme satisfies the properties defined in the following definitions.

Definition 5 (**Completeness**)**.** A DV-zkSNARK scheme for a relation R satisfies completeness with completeness error $\varepsilon(\lambda)$ if for all adversaries $\mathcal{A}$,

$$\Pr\left[\mathsf{Verifier}(\mathsf{vk}, \mathbb{x}, \boldsymbol{\pi}) = 0 \wedge (\mathbb{x}, \mathbb{w}) \in \mathsf{R} \middle| \begin{array}{r} (\mathsf{vk}, \mathsf{crs}) \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{R}) \\ \boldsymbol{\pi} \leftarrow \mathsf{Prover}(\mathbb{x}, \mathbb{w}, \mathsf{crs}) \end{array}\right] \leq \varepsilon(\lambda) + \mathsf{negl}(\lambda).$$

Definition 6 (**Adaptive Knowledge Soundness**)**.** A DV-zkSNARK scheme for a relation R satisfies knowledge soundness with knowledge error $\varepsilon(\lambda)$ if there exists an expected PPT extractor $\mathcal{E}$ such that, for every stateful PPT adversarial Prover^*,

$$\Pr\left[b = 1 \wedge (\mathbb{x}, \mathbb{w}) \notin \mathsf{R} \middle| \begin{array}{r} (\mathsf{vk}, \mathsf{crs}) \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{R}) \\ (\mathbb{x}, \boldsymbol{\pi}) \leftarrow \mathsf{Prover}^*(\mathsf{R}, \mathsf{crs}) \\ b \leftarrow \mathsf{Verifier}(\mathsf{vk}, \mathbb{x}, \boldsymbol{\pi}) \\ \mathbb{w} \leftarrow \mathcal{E}(\mathsf{R}, \mathsf{crs}, \boldsymbol{\pi}) \end{array}\right] \leq \varepsilon(\lambda) + \mathsf{negl}(\lambda).$$

The extractor is assumed to have black-box access to the (malicious) prover and may rewind it at any point during the interaction.

The *adaptive* variant of this property means that the adversary $\mathcal{A}$ is allowed to observe the crs before choosing the statement and producing a proof. In contrast, *non-adaptive* knowledge soundness restricts the adversary to commit to a statement before observing the crs. It was shown in [BCI+13] (Lemma 6.1) that zkSNARKs constructed using linear targeted malleability assumption cannot achieve adaptive knowledge soundness.

Definition 7 (Honest Verifier Zero-Knowledge). A $\mathsf{DV\text{-}zkSNARK}$ scheme for a relation R satisfies the honest-verifier zero-knowledge (HVZK) property if there exists a PPT simulator $\mathsf{Sim}(\mathsf{vk}, \mathsf{crs}, \mathbb{x})$ such that for every PPT adversary $\mathcal{A}$ and every $(\mathbb{x}, \mathbb{w}) \in \mathsf{R}$,

$$\left| \Pr\left[\mathcal{A}(\mathsf{vk}, \mathbb{x}, \boldsymbol{\pi}) = 1 \;\middle|\; \begin{array}{c}(\mathsf{vk}, \mathsf{crs}) \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{R}) \\ \boldsymbol{\pi} \leftarrow \mathsf{Prover}(\mathbb{x}, \mathbb{w}, \mathsf{crs})\end{array}\right] - \Pr\left[\mathcal{A}(\mathsf{vk}, \mathbb{x}, \boldsymbol{\pi}) = 1 \;\middle|\; \begin{array}{c}(\mathsf{vk}, \mathsf{crs}) \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{R}) \\ \boldsymbol{\pi} \leftarrow \mathsf{Sim}(\mathsf{vk}, \mathsf{crs}, \mathbb{x})\end{array}\right] \right| = \mathsf{negl}(\lambda).$$

2.5 Coordinate-Wise Special Soundness (CWSS)

To prove knowledge soundness, we rely on coordinate-wise special soundness (CWSS) as introduced in [FMN24]. Let $\mathcal{K}$ be a finite set and $\ell \in \mathbb{N}$. For vectors $\vec{\mathbf{x}} = (\mathbf{x}_1, \ldots, \mathbf{x}_\ell)$ and $\vec{\mathbf{y}} = (\mathbf{y}_1, \ldots, \mathbf{y}_\ell)$ in $\mathcal{K}^\ell$, define the relation $\vec{\mathbf{x}} \equiv_i \vec{\mathbf{y}}$ for $i \in [\ell]$ as:

$$\vec{\mathbf{x}} \equiv_i \vec{\mathbf{y}} \iff \mathbf{x}_i \neq \mathbf{y}_i \text{ and } \forall j \in [\ell] \setminus \{i\},\ \mathbf{x}_j = \mathbf{y}_j.$$

This means $\vec{\mathbf{x}}$ and $\vec{\mathbf{y}}$ differ only at the i-th position. Now define $\mathsf{SS}(\mathcal{K}, \ell)$ as follows:

$$\left\{ (\vec{x}_1, \ldots, \vec{x}_{\ell+1}) \in \mathcal{K}^{\ell+1} : \begin{array}{l}\exists k \in [\ell+1],\ \forall i \in [\ell], \\ \exists j \in [\ell+1] \setminus \{k\},\ \vec{x}_k \equiv_i \vec{x}_j\end{array} \right\}.$$

Definition 8. Let $\Pi = (\mathsf{Setup}, \mathsf{Prover}, \mathsf{Verifier})$ be an interactive proof system for a relation R, and suppose challenge space of $\mathsf{Verifier}$ is $\mathcal{K}^\ell$. We say that Π satisfies ℓ-coordinate-wise special soundness if there exists a polynomial-time extractor that, given public parameters, a statement $\mathbb{x}$, and $\ell + 1$ accepting transcripts $(\vec{\mathbf{k}}_i, \boldsymbol{\pi}_i)_{i \in [\ell+1]}$ with distinct $\vec{\mathbf{k}}_i$ such that $(\vec{\mathbf{k}}_1, \ldots, \vec{\mathbf{k}}_{\ell+1}) \in \mathsf{SS}(\mathcal{K}, \ell)$, outputs a witness $\mathbb{w}$ such that $(\mathbb{x}, \mathbb{w}) \in \mathsf{R}$.

Lemma 8. [FMN24] *Let Π be a public-coin three-round protocol for relation R with the challenge space of $\mathcal{K}$. If Π is ℓ-coordinate-wise special sound, then it is knowledge sound with knowledge error $\frac{\ell}{|\mathcal{K}|}$.*

2.6 Ring–Field Isomorphism

We use the ring–field isomorphism of [HSS24], which allows vectors over a finite field to be represented by low-norm elements over a polynomial ring. This algebraic correspondence is summarized in the following lemma.

Lemma 9 ([HSS24])**.** *Let $b, r, d \in \mathbb{Z}_{>0}$ with $r \mid d$, and let $p = b^r + 1$ be prime. Set $m = d/r$. Then*

$$\mathcal{R}_q/(X^m - b) \cong \mathbb{Z}_p^m.$$

A key property of this isomorphism is that the resulting ring representatives have coefficients bounded in absolute value by $(b + 2)/2$. Hence, encoded witnesses have small coefficient norm, which is particularly useful in lattice-based constructions where shortness of committed or opened vectors is required.

We denote the corresponding encoding and decoding maps by

$$\mathsf{Encode} : \mathbb{Z}_p^m \to \mathcal{R}_q/(X^m - b), \qquad \mathsf{Decode} : \mathcal{R}_q/(X^m - b) \to \mathbb{Z}_p^m.$$

In particular, a field element $a \in \mathbb{Z}_p$ is encoded by first identifying it with the vector

$$a\mathbf{e}_1 = \begin{bmatrix} a & 0 & \cdots & 0 \end{bmatrix}^{\mathsf{T}} \in \mathbb{Z}_p^m,$$

and then applying Encode. Moreover, decoding is invariant under the choice of representative: if $f, g \in \mathcal{R}_q$ satisfy

$$f \equiv g \pmod{X^m - b},$$

then

$$\mathsf{Decode}(f) = \mathsf{Decode}(g).$$

Finally, we emphasize that the moduli p and q are chosen for different purposes. The prime $p = b^r + 1$ determines the field over which the encoded computation is defined, whereas q is the modulus of the ambient lattice ring. In particular, p need not be smaller than q. Our parameter selection follows the regime

$$b < q < p,$$

with q chosen sufficiently larger than b so that the encoded elements remain genuinely low-norm when viewed in $\mathcal{R}_q$.

3 Revisiting the Ajtai Commitment Scheme

Algorithm 3 TACS(Trapdoor-based Ajtai Commitment Scheme)

$\underline{\mathsf{TACS.Setup}}$

Input: security parameter λ.
Set $n = n(\lambda)$ and $q = q(\lambda)$.
Set $\delta := \lfloor \log q \rfloor + 1$
for $i = 1, \cdots, 2k$ **do**
  $\{\mathbf{A}_i, \mathbf{R}_i\} \leftarrow \mathsf{TrapGen}(n, n(\delta+1))$.
  Split each $\mathbf{A}_i$ into $\mathbf{A}_i^{\mathsf{up}}, \mathbf{A}_i^{\mathsf{down}} \in \mathcal{R}_q^{\frac{n}{2} \times n(\delta+1)}$.
end for
Set $\mathbf{A}_{\mathsf{com}} := \begin{bmatrix} (\mathbf{A}_1^{\mathsf{up}})^\top & \cdots & (\mathbf{A}_{2k}^{\mathsf{up}})^\top \end{bmatrix} \in \mathcal{R}_q^{n(\delta+1) \times nk}$.
Set $\mathbf{A}_{\mathsf{hide}} := \begin{bmatrix} (\mathbf{A}_1^{\mathsf{down}})^\top & \cdots & (\mathbf{A}_{2k}^{\mathsf{down}})^\top \end{bmatrix} \in \mathcal{R}_q^{n(\delta+1) \times nk}$.
Output public parameters $\mathsf{pp} := (\mathbf{A}_{\mathsf{com}}, \mathbf{A}_{\mathsf{hide}})$ and the trapdoors $\mathsf{td} = \{\mathbf{R}_i\}_{i=1}^{2k}$.

$\underline{\mathsf{TACS.Commit}}$

Input: public parameters pp, witness vector $\vec{\mathbf{b}} \in \mathcal{R}_q^{nk}$.
Select $s_\mu > 2d.q^{\frac{(\delta+1)}{k} + \frac{2}{dnk}}$.
$\vec{\boldsymbol{\mu}} \leftarrow \mathcal{D}_{s_\mu}^{nk}$.
$\vec{\mathbf{c}} := \mathbf{A}_{\mathsf{com}}\vec{\mathbf{b}} + \mathbf{A}_{\mathsf{hide}}\vec{\boldsymbol{\mu}}$.
Output the commitment $\mathsf{cm} := (\vec{\mathbf{c}})$ and decommitment state $\mathsf{st} := (\vec{\boldsymbol{\mu}})$.

$\underline{\mathsf{TACS.Open}}$

Input: public parameters pp, commitment cm, witness vector $\vec{\mathbf{b}}$ and decommitment state st.
Parse $\mathsf{pp} := (\mathbf{A}_{\mathsf{com}}, \mathbf{A}_{\mathsf{hide}})$, $\mathsf{cm} := (\vec{\mathbf{c}})$ $\mathsf{st} := (\vec{\boldsymbol{\mu}})$.
Output 1 if $\| \begin{bmatrix} \vec{\mathbf{b}} \\ \vec{\boldsymbol{\mu}} \end{bmatrix} \|_\infty \leq \beta_{\mathsf{open}}$ and $\vec{\mathbf{c}} = \mathbf{A}_{\mathsf{com}}\vec{\mathbf{b}} + \mathbf{A}_{\mathsf{hide}}\vec{\boldsymbol{\mu}}$

The Ajtai commitment scheme is a fundamental building block in the construction of lattice-based zero-knowledge proof systems. Given a uniformly sampled matrix $\mathbf{A} \in \mathcal{R}_q^{n \times m}$, a commitment to a short witness vector $\vec{\mathbf{w}} \in \mathcal{R}_q^m$ is computed as $\vec{\mathbf{c}} = \mathbf{A}\vec{\mathbf{w}}$. The binding

property follows from the hardness of the Module Short Integer Solution (MSIS) problem. Indeed, suppose an adversary is able to open the same commitment to another short vector $\vec{\mathbf{w}}'$. Then,

$$\mathbf{A}\vec{\mathbf{w}} = \mathbf{A}\vec{\mathbf{w}}' \quad \Longrightarrow \quad \mathbf{A}(\vec{\mathbf{w}} - \vec{\mathbf{w}}') = \mathbf{0}.$$

Since $\vec{\mathbf{w}} - \vec{\mathbf{w}}'$ remains short, the adversary obtains a non-trivial short vector in the kernel of $\mathbf{A}$, thereby solving an instance of the MSIS problem, which is assumed to be computationally infeasible. To achieve the hiding property, it is sufficient to sample an additional matrix $\mathbf{A}'$ and include in the commitment a Gaussian vector $\vec{\mathbf{w}}'$ sampled according to the conditions specified in Lemma 5. The resulting commitment $\mathbf{A}\vec{\mathbf{w}} + \mathbf{A}'\vec{\mathbf{w}}'$ is statistically hiding.

In this section, we introduce several modifications to the Ajtai commitment scheme. The main modification concerns the setup procedure, where the commitment matrix is generated from multiple uniformly random submatrices, each associated with a corresponding trapdoor. The resulting construction is presented in Algorithm 3. We refer to the proposed scheme as the Trapdoor-based Ajtai Commitment Scheme, denoted by TACS.

Claim 1. *The matrices* $\mathbf{A}_{\mathsf{com}}$ *and* $\mathbf{A}_{\mathsf{hide}}$ *generated by the* $\mathsf{TACS.Setup}$ *algorithm are statistically close to uniformly random matrices.*

Proof. According to Step 5 of the $\mathsf{TACS.Setup}$ algorithm and Lemma 6, each matrix $\mathbf{A}_i$ is statistically close to uniform. Consequently, both the upper and lower submatrices derived from each $\mathbf{A}_i$ are also statistically close to uniform.

Since the matrices are sampled independently, the concatenation of these submatrices preserves statistical uniformity. Therefore, the resulting matrices $\mathbf{A}_{\mathsf{com}}$ and $\mathbf{A}_{\mathsf{hide}}$ are statistically close to uniformly random matrices. □

Remark. Since the matrices $\mathbf{A}_{\mathsf{com}}$ and $\mathbf{A}_{\mathsf{hide}}$ are statistically close to uniformly random matrices, the standard assumptions underlying the Ajtai commitment scheme remain satisfied. Consequently, the proposed construction preserves both the binding and hiding properties.

4 An Argument System for Verifying Inner Product Relation

In this section, we present an argument system for the relation $\mathsf{R}_{\mathrm{IPA}}$ defined in Equation (1), built upon TACS commitment with a trusted setup. To construct this protocol, we first introduce several modifications to Algorithm 2. We then present the argument system for $\mathsf{R}_{\mathrm{IPA}}$. This relation requires proving two statements. First, the prover must demonstrate that the inner product has been computed correctly. Second, the prover must prove that the witness has a bounded norm.

4.1 Modified Preimage Sampling Algorithm

We modify the $\mathsf{SamplePre}$ algorithm to suit our protocol. In this algorithm, a trapdoor holder can sample a short preimage $\vec{\mathbf{u}} = \vec{\mathbf{p}} + \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]\vec{\mathbf{z}}$ for a given syndrome vector $\vec{\mathbf{v}}$. In our setting, however, the prover does not directly observe $\vec{\mathbf{u}}$. As we see later, the vector $\vec{\mathbf{p}}$ becomes part of versifier's secret key, while a randomized version of $\vec{\mathbf{z}}$ is embedded in the CRS. We note that this randomized vector is computationally independent of the trapdoor $\mathbf{R}$ and reveals no information about $\vec{\mathbf{z}}$ itself. We introduce two key modifications to the standard preimage sampling algorithm.

1. Shared Perturbation Vector: For matrices $(\mathbf{A}_i, \mathbf{R}_i)$, $i \in [k]$, and corresponding syndromes $\vec{\mathbf{u}}_i$, preimages $\vec{\mathbf{v}}_i$ satisfying $\mathbf{A}_i\vec{\mathbf{u}}_i = \vec{\mathbf{v}}_i$ share the same perturbation vector $\vec{\mathbf{p}}$.
2. Independent Sampling of $\vec{\mathbf{p}}$ and $\vec{\mathbf{z}}$: Vectors $\vec{\mathbf{p}}$ and $\vec{\mathbf{z}}$ are independently drawn from Gaussian distributions with distinct covariance matrices, ensuring $\vec{\mathbf{v}}$ remains within the required norm bound.

The modified SamplePre algorithm is shown in Algorithm 4.

Algorithm 4 ModSamplePre

Input: Ajtai matrix $\mathbf{A} \in \mathcal{R}_q^{n\times n(\delta+1)}$, corresponding trapdoor $\mathbf{R}$, syndrome vector $\vec{\mathbf{u}}$, perturbation vector $\vec{\mathbf{p}}$ and gaussian parameter s_z.
Sample $\vec{\mathbf{z}} \leftarrow \mathcal{D}_{s_z}^m$ such that

$$\mathbf{G}\vec{\mathbf{z}} = \vec{\mathbf{u}} - \mathbf{A}\vec{\mathbf{p}}.$$

Output $\vec{\mathbf{z}}$.

4.2 Inner product verification

In this section, we show that a verifier with trapdoors $\mathbf{R}_i$ for the TACS Ajtai matrices can verify the correctness of a claimed inner product between a committed witness and a uniformly random challenge $\vec{\mathbf{u}}_{\mathsf{chall}} \xleftarrow{\$} (\mathcal{R}_q^{1\times 2nk})$. We first generate a trapdoor vector corresponding to this challenge using the ChallTrap in Algorithm 5. We show that the verifier, equipped with the trapdoor vector and the secret perturbation vector $\vec{\mathbf{p}}$, can efficiently verify the correctness of the inner product between the committed witness and the challenge vector $\vec{\mathbf{u}}_{\mathsf{chall}}$. It is easy to check that the vector $\vec{\mathbf{z}}_{\mathsf{G}}$ has Gaussian parameter equal to $s_z\sqrt{\sum_{i=0}^{\log q-1} 4^i}$ where, s_z is the same as the input of Algorithm 4.

Algorithm 5 ChallTrap

Input: public parameters pp, trapdoor td, perturbation vector $\vec{\mathbf{p}}$ and the challenge vector $\vec{\mathbf{u}}_{\mathsf{chall}}$.
Split $\vec{\mathbf{u}}_{\mathsf{chall}}$ into $4k$ subvectors $\vec{\mathbf{u}}_i \in \mathcal{R}_q^{1\times n/2}$, for all $i \in [4k]$.
for $i = 1, \cdots, 2k$ **do**
  $\vec{\mathbf{z}}_i \leftarrow \mathsf{ModSamplePre}(\mathbf{A}_i, \mathbf{R}_i, \begin{bmatrix}\vec{\mathbf{u}}_i & \vec{\mathbf{u}}_{i+2k}\end{bmatrix}^\top, \vec{\mathbf{p}}, s_z)$.
  Set $\vec{\mathbf{y}}_i := \vec{\mathbf{p}} + \begin{bmatrix}\mathbf{R}_i \\ \mathbf{I}\end{bmatrix}\vec{\mathbf{z}}_i$. ▷ Note that $\mathbf{A}_i\vec{\mathbf{y}}_i = \begin{bmatrix}\vec{\mathbf{u}}_i & \vec{\mathbf{u}}_{i+2k}\end{bmatrix}^\top$.
  Set $\vec{\mathbf{z}}_{\mathsf{G},i}^{\mathsf{up}} := \mathbf{G}^{\mathsf{up}}\vec{\mathbf{z}}_i \in \mathcal{R}_q^{n/2}$.
  Set $\vec{\mathbf{z}}_{\mathsf{G},i}^{\mathsf{down}} := \mathbf{G}^{\mathsf{down}}\vec{\mathbf{z}}_i \in \mathcal{R}_q^{n/2}$.
end for
Set $\vec{\mathbf{z}}_{\mathsf{G}}^{\mathsf{up}} = \begin{bmatrix}(\vec{\mathbf{z}}_{\mathsf{G},1}^{\mathsf{up}})^\top & \cdots & (\vec{\mathbf{z}}_{\mathsf{G},2k}^{\mathsf{up}})^\top\end{bmatrix} \in \mathcal{R}_q^{1\times nk}$.
Set $\vec{\mathbf{z}}_{\mathsf{G}}^{\mathsf{down}} = \begin{bmatrix}(\vec{\mathbf{z}}_{\mathsf{G},1}^{\mathsf{down}})^\top & \cdots & (\vec{\mathbf{z}}_{\mathsf{G},2k}^{\mathsf{down}})^\top\end{bmatrix} \in \mathcal{R}_q^{1\times nk}$.
Output $\vec{\mathbf{z}}_{\mathsf{G}} = \begin{bmatrix}\vec{\mathbf{z}}_{\mathsf{G}}^{\mathsf{up}} & \vec{\mathbf{z}}_{\mathsf{G}}^{\mathsf{down}}\end{bmatrix} \in \mathcal{R}_q^{1\times 2nk}$.

The following claim shows that a verifier who possesses $\vec{\mathbf{p}}$ and having access to the product of $\vec{\mathbf{z}}_{\mathsf{G}}$ with the witness can verify the product of $\vec{\mathbf{u}}_{\mathsf{chall}}$ with the same witness.

Claim 2. *The following equality holds, if $\vec{\mathbf{z}}_{\mathsf{G}}$ is computed from $\vec{\mathbf{u}}_{\mathsf{chall}}$ using the algorithm* ChallTrap.

$$\vec{\mathbf{u}}_{\mathsf{chall}}.\begin{bmatrix}\vec{\mathbf{b}} \\ \vec{\boldsymbol{\mu}}\end{bmatrix} = \langle\vec{\mathbf{p}}, \vec{\mathbf{c}}\rangle + \vec{\mathbf{z}}_{\mathsf{G}}.\begin{bmatrix}\vec{\mathbf{b}} \\ \vec{\boldsymbol{\mu}}\end{bmatrix}. \tag{4}$$

Proof. . According to step 2 of Algorithm 5, we split $\vec{\mathbf{u}}_{\mathsf{chall}}$ into $\vec{\mathbf{u}}_i$ for $i \in [4k]$. We recall that $\vec{\mathbf{b}}, \vec{\boldsymbol{\mu}} \in \mathcal{R}_q^{nk}$.Then, the inner product of witness by $\vec{\mathbf{u}}_{\mathsf{chall}}$ can be splitted as follows,

$$\vec{\mathbf{u}}_{\mathsf{chall}} . \begin{bmatrix} \vec{\mathbf{b}} \\ \vec{\boldsymbol{\mu}} \end{bmatrix} = \sum_{i=1}^{2k} (\vec{\mathbf{u}}_i . \left[\vec{\mathbf{b}}\right]_{(i-1)\frac{n}{2}+1}^{i\frac{n}{2}} + \vec{\mathbf{u}}_{i+2k} . [\vec{\boldsymbol{\mu}}]_{(i-1)\frac{n}{2}+1}^{i\frac{n}{2}}).$$

Set $l_i = (i-1)\frac{n}{2} + 1$ and $h_i = i\frac{n}{2}$ then we have:

$$\begin{aligned} \vec{\mathbf{u}}_{\mathsf{chall}} \begin{bmatrix} \vec{\mathbf{b}} \\ \vec{\boldsymbol{\mu}} \end{bmatrix} &= \sum_{i=1}^{2k} \begin{bmatrix} \vec{\mathbf{u}}_i & \vec{\mathbf{u}}_{i+2k} \end{bmatrix} \begin{bmatrix} [\vec{\mathbf{b}}]_{l_i}^{h_i} \\ [\vec{\boldsymbol{\mu}}]_{l_i}^{h_i} \end{bmatrix} = \sum_{i=1}^{2k} (\mathbf{A}_i \vec{\mathbf{y}}_i)^\top \begin{bmatrix} [\vec{\mathbf{b}}]_{l_i}^{h_i} \\ [\vec{\boldsymbol{\mu}}]_{l_i}^{h_i} \end{bmatrix} \\ &= \sum_{i=1}^{2k} \vec{\mathbf{y}}_i^\top \begin{bmatrix} (\mathbf{A}_i^{\mathsf{up}})^\top & (\mathbf{A}_i^{\mathsf{down}})^\top \end{bmatrix} \begin{bmatrix} \left[\vec{\mathbf{b}}\right]_{l_i}^{h_i} \\ [\vec{\boldsymbol{\mu}}]_{l_i}^{h_i} \end{bmatrix}. \end{aligned}$$

We replace $\vec{\mathbf{y}}_i^\top$ vectors with $\vec{\mathbf{p}}^\top + \vec{\mathbf{z}}_i^\top \begin{bmatrix} \mathbf{R}_i^\top & \mathbf{I} \end{bmatrix}$ (see step 5 of Algorithm 5) and get:

$$\begin{aligned} &\sum_{i=1}^{2k} \left(\vec{\mathbf{p}}^\top + \vec{\mathbf{z}}_i^\top \begin{bmatrix} \mathbf{R}_i^\top & \mathbf{I} \end{bmatrix}\right) \begin{bmatrix} (\mathbf{A}_i^{\mathsf{up}})^\top & (\mathbf{A}_i^{\mathsf{down}})^\top \end{bmatrix} \begin{bmatrix} \left[\vec{\mathbf{b}}\right]_{l_i}^{h_i} \\ [\vec{\boldsymbol{\mu}}]_{l_i}^{h_i} \end{bmatrix} \\ &= \vec{\mathbf{p}}^\top \sum_{i=1}^{2k} \begin{bmatrix} (\mathbf{A}_i^{\mathsf{up}})^\top & (\mathbf{A}_i^{\mathsf{down}})^\top \end{bmatrix} \begin{bmatrix} \left[\vec{\mathbf{b}}\right]_{l_i}^{h_i} \\ [\vec{\boldsymbol{\mu}}]_{l_i}^{h_i} \end{bmatrix} + \sum_{i=1}^{2k} \left(\vec{\mathbf{z}}_i^\top \begin{bmatrix} \mathbf{R}_i^\top & \mathbf{I} \end{bmatrix}\right) \begin{bmatrix} (\mathbf{A}_i^{\mathsf{up}})^\top & (\mathbf{A}_i^{\mathsf{down}})^\top \end{bmatrix} \begin{bmatrix} \left[\vec{\mathbf{b}}\right]_{l_i}^{h_i} \\ [\vec{\boldsymbol{\mu}}]_{l_i}^{h_i} \end{bmatrix} \\ &= \vec{\mathbf{p}}^\top . \vec{\mathbf{c}} + \sum_{i=1}^{2k} \vec{\mathbf{z}}_i^\top \begin{bmatrix} \mathbf{R}_i^\top & \mathbf{I} \end{bmatrix} \left((\mathbf{A}_i^{\mathsf{up}})^\top \left[\vec{\mathbf{b}}\right]_{l_i}^{h_i} + (\mathbf{A}_i^{\mathsf{down}})^\top [\vec{\boldsymbol{\mu}}]_{l_i}^{h_i} \right) \end{aligned} \tag{5}$$

Note that based on definition of gadget trapdoors that $\begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} \mathbf{A} = \mathbf{G}$(see Lemma 6) we can write:

$$\begin{bmatrix} \mathbf{R}_i^\top & \mathbf{I} \end{bmatrix} (\mathbf{A}_i^{\mathsf{up}})^\top = (\mathbf{G}^{\mathsf{up}})^\top \ , \ \begin{bmatrix} \mathbf{R}_i^\top & \mathbf{I} \end{bmatrix} (\mathbf{A}_i^{\mathsf{down}})^\top = (\mathbf{G}^{\mathsf{down}})^\top.$$

By replacing in Equation (5) we get

$$\begin{aligned} \vec{\mathbf{u}}_{\mathsf{chall}} \begin{bmatrix} \vec{\mathbf{b}} \\ \vec{\boldsymbol{\mu}} \end{bmatrix} &= \vec{\mathbf{p}}^\top . \vec{\mathbf{c}} + \sum_{i=1}^{2k} \left(\vec{\mathbf{z}}_i^\top (\mathbf{G}^{\mathsf{up}})^\top \left[\vec{\mathbf{b}}\right]_{l_i}^{h_i} + \vec{\mathbf{z}}_i^\top (\mathbf{G}^{\mathsf{down}})^\top [\vec{\boldsymbol{\mu}}]_{l_i}^{h_i} \right) \xrightarrow{\text{steps 6,7 of ChallTrap}} \\ &= \vec{\mathbf{p}}^\top . \vec{\mathbf{c}} + \sum_{i=1}^{2k} \left((\vec{\mathbf{z}}_{\mathsf{G},i}^{\mathsf{up}})^\top \left[\vec{\mathbf{b}}\right]_{l_i}^{h_i} + (\vec{\mathbf{z}}_{\mathsf{G},i}^{\mathsf{down}})^\top [\vec{\boldsymbol{\mu}}]_{l_i}^{h_i} \right) = \langle \vec{\mathbf{p}}, \vec{\mathbf{c}} \rangle + \vec{\mathbf{z}}_{\mathsf{G}} . \begin{bmatrix} \vec{\mathbf{b}} \\ \vec{\boldsymbol{\mu}} \end{bmatrix} \end{aligned}$$

□

To verify Equation (4), the verifier must evaluate $\vec{\mathbf{z}}_{\mathsf{G}} \begin{bmatrix} \vec{\mathbf{b}} \\ \vec{\boldsymbol{\mu}} \end{bmatrix}$.Such a requirement introduces a recursive issue, since the verifier would need to verify an additional inner product in order to validate the original inner-product relation. To resolve this issue, the verifier computes a random linear combination of $\vec{\mathbf{u}}_{\mathsf{chall}}$ and $\vec{\mathbf{z}}_{\mathsf{G}}$. The combination is generated using the Gaussian2Uniform algorithm, as described in Algorithm 6. The following claim shows that, when the inputs to the algorithm are chosen appropriately, the resulting combination is statistically close to uniform distribution.

Claim 3. *Assume that the value* $\mathbf{z}_{\mathsf{sec}}$ *is sampled from a discrete Gaussian distribution with parameter* $s_{\mathsf{sec}} = s_z \sqrt{\sum_{i=0}^{\log q - 1} 4^i}$, *and that the modulus* q *satisfies* $q \equiv 5 \pmod 8$. *Then, the output distribution of Algorithm 6 is statistically close to the uniform distribution.*

Algorithm 6 Gaussian2Uniform

1: **Input**: Uniform vector $\vec{\mathbf{u}}_{\mathsf{chall}}$, Gaussian secret value $\mathbf{z}_{\mathsf{sec}}$, trapdoor based challenge vector $\vec{\mathbf{z}}_{\mathsf{G}}$.
2: Set $\vec{\mathbf{z}}_{\mathsf{G,ext}} := \begin{bmatrix} \vec{\mathbf{z}}_{\mathsf{G}}^{\top} \\ \mathbf{z}_{\mathsf{sec}} \end{bmatrix} \in \mathcal{R}_q^{2nk+1}$.
3: Set matrix $\mathbf{A} := \begin{bmatrix} \mathbf{I}_{2nk} & \vec{\mathbf{u}}_{\mathsf{chall}}^{\top} \end{bmatrix} \in \mathcal{R}_q^{2nk \times 2nk+1}$.
4: **Output** $\vec{\mathbf{u}}_{\mathsf{aux}} = \left(\mathbf{A}\vec{\mathbf{z}}_{\mathsf{G,ext}}\right)^{\top}$.

Proof. According to Steps 6–7 of Algorithm 5, each entry of $\vec{\mathbf{z}}_{\mathsf{G}}$ is obtained as a weighted sum of independent discrete Gaussian samples with parameter s_z. Since the gadget weights are $1, 2, 4, \ldots, 2^{\log q-1}$, the Gaussian parameter of each resulting entry is $s_{\mathsf{sec}} = s_z\sqrt{\sum_{i=0}^{\log q-1} 4^i}$, which matches the distribution parameter of $\mathbf{z}_{\mathsf{sec}}$. Moreover, the block-diagonal structure of $\mathbf{G}$, together with the independence of the underlying vectors $\{\vec{\mathbf{z}}_i\}_i$, ensures that the entries of $\vec{\mathbf{z}}_{\mathsf{G}}$ are mutually independent, provided that no two entries are computed from overlapping Gaussian coordinates. Consequently, the extended vector $\vec{\mathbf{z}}_{\mathsf{G,ext}}$ also consists of independent discrete Gaussian samples with the same parameter s_{sec}. Moreover, it holds that $s_{\mathsf{sec}} > 2d \cdot q^{\frac{2nk}{2nk+1}+\frac{2}{d(2nk+1)}}$. On the other hand, the vector $\vec{\mathbf{u}}_{\mathsf{chall}}$ is uniformly distributed. Therefore, all conditions of Lemma 4 are satisfied. It follows that the output distribution of Algorithm 6 is statistically close to uniform. □

Figure 2 illustrates how the verifier can validate the inner product between the witness and $\vec{\mathbf{u}}_{\mathsf{chall}}$ by using the auxiliary challenge vector $\vec{\mathbf{u}}_{\mathsf{aux}}$.

We briefly explain why verification succeeds when the prover follows the protocol honestly. A complete analysis of completeness and knowledge soundness for the full protocol is provided in Section 4.5.

Note that $\vec{\mathbf{u}}_{\mathsf{aux}}$ is equal to $\vec{\mathbf{z}}_{\mathsf{G}} + \mathbf{z}_{\mathsf{sec}}\vec{\mathbf{u}}_{\mathsf{chall}}$. Therefore, $\boldsymbol{\pi}_{\mathsf{aux}}$ is computed as:

$$\begin{aligned} \boldsymbol{\pi}_{\mathsf{aux}} = \vec{\mathbf{u}}_{\mathsf{aux}}.\begin{bmatrix}\vec{\mathbf{b}}\\ \vec{\boldsymbol{\mu}}\end{bmatrix} &= \vec{\mathbf{z}}_{\mathsf{G}}.\begin{bmatrix}\vec{\mathbf{b}}\\ \vec{\boldsymbol{\mu}}\end{bmatrix} + \mathbf{z}_{\mathsf{sec}}\left(\vec{\mathbf{u}}_{\mathsf{chall}}.\begin{bmatrix}\vec{\mathbf{b}}\\ \vec{\boldsymbol{\mu}}\end{bmatrix}\right) \\ &= \vec{\mathbf{z}}_{\mathsf{G}}.\begin{bmatrix}\vec{\mathbf{b}}\\ \vec{\boldsymbol{\mu}}\end{bmatrix} + \mathbf{z}_{\mathsf{sec}}\boldsymbol{\pi}_{\mathsf{chall}}. \end{aligned} \tag{6}$$

By adding $\langle \vec{\mathbf{p}}, \vec{\mathbf{c}} \rangle$ to both sides of Equation (6), and applying Claim 2 we get:

$$\boldsymbol{\pi}_{\mathsf{aux}} + \langle \vec{\mathbf{p}}, \vec{\mathbf{c}} \rangle = (\mathbf{z}_{\mathsf{sec}} + 1)\boldsymbol{\pi}_{\mathsf{chall}}.$$

4.3 Range Proof Verification

Beyond verifying the inner product, the verifier must be assured that the witness satisfies a norm constraint, otherwise, binding property of the commitment scheme may fail. For this purpose, we use the approximate range proof technique from [BL17, LNS21], formalized in Lemma 3. As this method relies on inner product, it is naturally compatible with our protocol. Since the original technique is defined over $\mathbb{Z}_q$, we adapt it to operate over polynomial rings. In what follows, we describe how this adaptation is achieved.

The witness vectors $\vec{\mathbf{b}}$ and $\vec{\boldsymbol{\mu}}$ could be formed as matrices

$$\begin{aligned} \mathbf{B} &= \begin{bmatrix} \mathsf{Rot}(\vec{\mathbf{b}}[1]) & \mathsf{Rot}(\vec{\mathbf{b}}[2]) & \cdots & \mathsf{Rot}(\vec{\mathbf{b}}[nk]) \end{bmatrix} \in \mathbb{Z}_q^{d\times dnk} \\ \mathbf{MU} &= \begin{bmatrix} \mathsf{Rot}(\vec{\boldsymbol{\mu}}[1]) & \mathsf{Rot}(\vec{\boldsymbol{\mu}}[2]) & \cdots & \mathsf{Rot}(\vec{\boldsymbol{\mu}}[nk]) \end{bmatrix} \in \mathbb{Z}_q^{d\times dnk} \\ \mathbf{M} &= \begin{bmatrix} \mathbf{B} | \mathbf{MU} \end{bmatrix} \in \mathbb{Z}_q^{d\times 2dnk} \quad \|\mathbf{M}\|_\infty \leq \beta_{\mathsf{open}} \end{aligned} \tag{7}$$

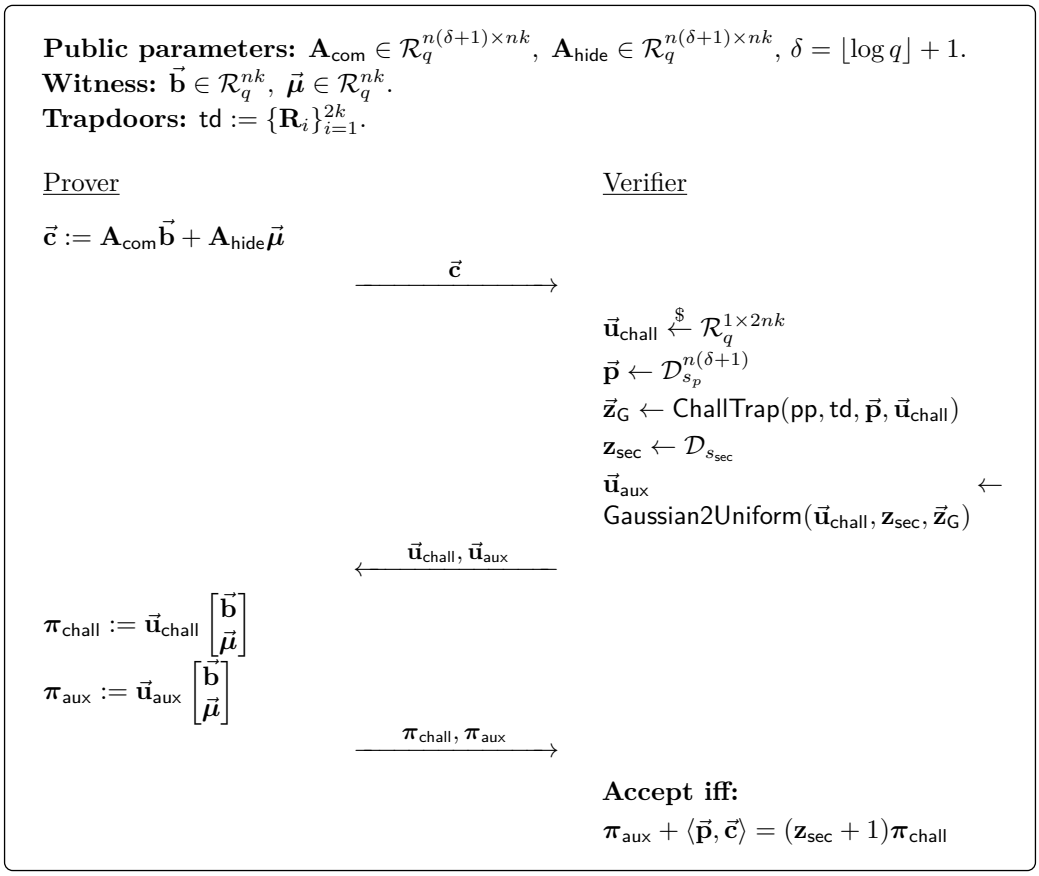


Figure 2: A Sigma protocol for verifying the inner product between a committed witness and the challenge vector $\vec{\mathbf{u}}_{\mathsf{chall}}$ using the auxiliary challenge $\vec{\mathbf{u}}_{\mathsf{aux}}$.

For a uniformly sampled matrix $\mathbf{K}$ from $\{0,1\}^{2dnk\times\lambda}$, the probability that a malicious prover can find a matrix $\mathbf{M}$ such that $||\mathbf{MK}||_\infty > \frac{1}{2}\beta_{\mathsf{open}}$ is bounded by $2^{-\lambda}$. It is necessary to sample these vectors using a hash function, as they will later be employed for extraction, which is only feasible within the random oracle model [FMN24]. Each column of $\mathbf{K}$ can be interpreted as a vector $\vec{\mathbf{k}}_i \in \mathcal{R}_q^{2nk}$, obtained by applying the inverse of coefficient embedding function, $\mathsf{cf}(.)^{-1}$, on every d consecutive entries in that column. The matrix product $\mathbf{MK}$ is equivalent to $\begin{bmatrix} \vec{\mathbf{b}} \\ \vec{\boldsymbol{\mu}} \end{bmatrix}^\top \cdot \begin{bmatrix} \vec{\mathbf{k}}_1 & \vec{\mathbf{k}}_2 & \dots & \vec{\mathbf{k}}_\lambda \end{bmatrix}$. Therefore, these vectors $\vec{\mathbf{k}}_i$ can naturally serve as norm-checking challenge vectors.

Figure 3 illustrates the follow-up protocol to Figure 2, augmented with an additional range-checking procedure.

The verifier first applies the $\mathsf{ChallTrap}$ algorithm to the vectors $\vec{\mathbf{k}}_i$ in order to generate the corresponding trapdoor challenges, denoted by $\vec{\mathbf{z}}_{\mathsf{G},i}$. By Claim 2, the following relation holds:

$$\vec{\mathbf{k}}_i \begin{bmatrix} \vec{\mathbf{b}} \\ \vec{\boldsymbol{\mu}} \end{bmatrix} = \langle \vec{\mathbf{p}}, \vec{\mathbf{c}} \rangle + \vec{\mathbf{z}}_{\mathsf{G},i} \begin{bmatrix} \vec{\mathbf{b}} \\ \vec{\boldsymbol{\mu}} \end{bmatrix}. \tag{8}$$

Next, the verifier applies the $\mathsf{Gaussian2Uniform}$ algorithm to the vectors $\vec{\mathbf{z}}_{\mathsf{G},i}$ in order to derive uniformly distributed vectors $\vec{\mathbf{u}}_{\mathsf{chall},i}$, defined by

$$\vec{\mathbf{u}}_{\mathsf{chall},i} = \vec{\mathbf{z}}_{\mathsf{G},i} + \mathbf{z}_{\mathsf{ksec},i}\vec{\mathbf{u}}_{\mathsf{chall}}.$$

Public parameters: $\mathsf{pp} := (\mathbf{A}_{\mathsf{com}} \in \mathcal{R}_q^{n(\delta+1)\times nk},\ \mathbf{A}_{\mathsf{hide}} \in \mathcal{R}_q^{n(\delta+1)\times nk})$
Statement: The commitment $\vec{\mathbf{c}}$ to the witness $\vec{\mathbf{w}}$, previously generated $\vec{\mathbf{u}}_{\mathsf{chall}}$ and the corresponding verified proof $\boldsymbol{\pi}_{\mathsf{chall}}$.
Witness: $\vec{\mathbf{w}} := \begin{bmatrix}\vec{\mathbf{b}}\\ \vec{\boldsymbol{\mu}}\end{bmatrix} \in \mathcal{R}_q^{2nk}$
Trapdoors: $\mathsf{td} := (\{\mathbf{R}_i\}_{i=1}^{2k}, \vec{\mathbf{p}})$

Prover		Verifier
		$\vec{\mathbf{k}}_i \xleftarrow{\$} \mathsf{cf}^{-1}(\{0,1\}^{2dnk}) \quad \forall i \in [\lambda]$
		$\vec{\mathbf{z}}_{\mathsf{G},i} \leftarrow \mathsf{ChallTrap}(\mathsf{pp}, \mathsf{td}, \vec{\mathbf{p}}, \vec{\mathbf{k}}_i)$
		$(\mathbf{z}_{\mathsf{ksec},i}, \mathbf{z}_{\mathsf{sec},i}) \leftarrow \mathcal{D}^2_{s_{\mathsf{sec}}}$
		$\vec{\mathbf{u}}_{\mathsf{chall},i} \leftarrow$
		$\mathsf{Gaussian2Uniform}(\vec{\mathbf{u}}_{\mathsf{chall}}, \mathbf{z}_{\mathsf{ksec},i}, \vec{\mathbf{z}}_{\mathsf{G},i})$
		$\vec{\mathbf{z}}_{\mathsf{k},i} \leftarrow \mathsf{ChallTrap}(\mathsf{pp}, \mathsf{td}, \vec{\mathbf{p}}, \vec{\mathbf{u}}_{\mathsf{chall},i})$
		$\vec{\mathbf{u}}_{\mathsf{aux},i} \leftarrow$
		$\mathsf{Gaussian2Uniform}(\vec{\mathbf{u}}_{\mathsf{chall},i}, \mathbf{z}_{\mathsf{sec},i}, \vec{\mathbf{z}}_{\mathsf{k},i})$
	$\xleftarrow{\vec{\mathbf{u}}_{\mathsf{chall},i},\ \vec{\mathbf{u}}_{\mathsf{aux},i}}$	
$\boldsymbol{\pi}_{\mathsf{chall},i} := \vec{\mathbf{u}}_{\mathsf{chall},i} \begin{bmatrix}\vec{\mathbf{b}}\\ \vec{\boldsymbol{\mu}}\end{bmatrix}$		
$\boldsymbol{\pi}_{\mathsf{aux},i} := \vec{\mathbf{u}}_{\mathsf{aux},i} \begin{bmatrix}\vec{\mathbf{b}}\\ \vec{\boldsymbol{\mu}}\end{bmatrix}$		
$\forall i \in [\lambda]$		
	$\xrightarrow{\boldsymbol{\pi}_{\mathsf{chall},i},\ \boldsymbol{\pi}_{\mathsf{aux},i}}$	
		Accept iff:
		$\boldsymbol{\pi}_{\mathsf{aux},i} + \langle \vec{\mathbf{p}}, \vec{\mathbf{c}} \rangle = (\mathbf{z}_{\mathsf{sec},i} + 1)\boldsymbol{\pi}_{\mathsf{chall},i}$
		$\lVert \boldsymbol{\pi}_{\mathsf{chall},i} + \langle \vec{\mathbf{p}}, \vec{\mathbf{c}} \rangle - \mathbf{z}_{\mathsf{ksec},i}\boldsymbol{\pi}_{\mathsf{chall},i} \rVert_\infty < \frac{1}{2}\beta_{\mathsf{open}}$
		$\forall i \in [\lambda]$

Figure 3: A follow-up Sigma protocol for norm checking.

The prover's response to these challenges can then be expressed as

$$\begin{aligned}
\boldsymbol{\pi}_{\mathsf{chall},i} &= \vec{\mathbf{u}}_{\mathsf{chall},i} \begin{bmatrix}\vec{\mathbf{b}}\\ \vec{\boldsymbol{\mu}}\end{bmatrix} \\
&= (\vec{\mathbf{z}}_{\mathsf{G},i} + \mathbf{z}_{\mathsf{ksec},i}\vec{\mathbf{u}}_{\mathsf{chall}}) \begin{bmatrix}\vec{\mathbf{b}}\\ \vec{\boldsymbol{\mu}}\end{bmatrix} \\
&= \vec{\mathbf{z}}_{\mathsf{G},i} \begin{bmatrix}\vec{\mathbf{b}}\\ \vec{\boldsymbol{\mu}}\end{bmatrix} + \mathbf{z}_{\mathsf{ksec},i}\boldsymbol{\pi}_{\mathsf{chall}} \xrightarrow{+\langle \vec{\mathbf{p}}, \vec{\mathbf{c}} \rangle} \\
\boldsymbol{\pi}_{\mathsf{chall},i} + \langle \vec{\mathbf{p}}, \vec{\mathbf{c}} \rangle &= \langle \vec{\mathbf{p}}, \vec{\mathbf{c}} \rangle + \vec{\mathbf{z}}_{\mathsf{G},i} \begin{bmatrix}\vec{\mathbf{b}}\\ \vec{\boldsymbol{\mu}}\end{bmatrix} + \mathbf{z}_{\mathsf{ksec},i}\boldsymbol{\pi}_{\mathsf{chall}} \xrightarrow{-\mathbf{z}_{\mathsf{ksec},i}\boldsymbol{\pi}_{\mathsf{chall}}} \\
\boldsymbol{\pi}_{\mathsf{chall},i} + \langle \vec{\mathbf{p}}, \vec{\mathbf{c}} \rangle - \mathbf{z}_{\mathsf{ksec},i}\boldsymbol{\pi}_{\mathsf{chall}} &= \langle \vec{\mathbf{p}}, \vec{\mathbf{c}} \rangle + \vec{\mathbf{z}}_{\mathsf{G},i} \begin{bmatrix}\vec{\mathbf{b}}\\ \vec{\boldsymbol{\mu}}\end{bmatrix} \xrightarrow{\text{Equation (8)}} = \vec{\mathbf{k}}_i \begin{bmatrix}\vec{\mathbf{b}}\\ \vec{\boldsymbol{\mu}}\end{bmatrix}. \qquad (9)
\end{aligned}$$

Equation (9) shows that the verifier can derive the inner products between the norm-checking vectors and the witness from the prover's responses, and subsequently verify that the resulting values are sufficiently short.

Observe that the inner products involving the vectors $\vec{\mathbf{u}}_{\mathsf{chall},i}$ must themselves be verified for correctness. For this reason, the auxiliary challenge vectors $\vec{\mathbf{u}}_{\mathsf{aux},i}$ are also included in the protocol. In particular, the first verification step in Figure 3 is dedicated to validating the correctness of these inner-product responses.

4.4 Putting Everything Together

We now combine the protocols presented in Figures 2–3 to construct an inner-product argument system for the relation defined in Equation (1).

The protocols in Figures 2–3 are interactive and rely on private-coin. To eliminate interaction, we instantiate the construction in the common reference string (CRS) model. As a result, the protocol becomes a non-interactive designated verifier argument system. The resulting construction is described in Algorithms 7–9, and is denoted by DV-LatticeIPA.

In the setup phase, the protocol first generates the public parameters of the TACS commitment scheme together with the corresponding trapdoors. Next, the uniformly random challenge vectors, for which the prover must compute the corresponding inner products with the witness, are generated along with their associated auxiliary challenges. This phase corresponds to the verifier's role in the interactive protocol shown in Figure 2. Subsequently, Steps 10–17 generate the challenges required for the range-checking procedure. This part corresponds to the verifier's operations in the interactive protocol presented in Figure 3.

Using the crs, the prover can compute the commitment to the witness vector $\vec{\mathbf{b}}$ and generate the corresponding responses to all challenges in a single round. The prover then sends the resulting commitment and proof components to the verifier.

Given the verification key, the verifier checks the validity of the proof by verifying Equations (10)–(11).

4.5 Security Analysis

This section analyzes the security of the proposed argument system, focusing on completeness, knowledge soundness, and honest verifier zero-knowledge.

Lemma 10 (**Completeness**). *The* DV-LatticeIPA *protocol satisfies the completeness property with probability* $1 - 2^{-\lambda}$.

Proof. . Let $\vec{\mathbf{w}} := \begin{bmatrix} \vec{\mathbf{b}} \\ \vec{\boldsymbol{\mu}} \end{bmatrix}$ denotes the witness vector. To prove completeness, we begin with Equation (10). According to Step 16 of Algorithm 7 together with the Gaussian2Uniform procedure, the value $\vec{\boldsymbol{\pi}}_{\mathsf{aux}}[i]$ can be expressed as

$$\begin{aligned}
\vec{\boldsymbol{\pi}}_{\mathsf{aux}}[i] = \vec{\mathbf{u}}_{\mathsf{aux},i}\vec{\mathbf{w}} = \vec{\mathbf{u}}_{\mathsf{aux},i}. \begin{bmatrix} \vec{\mathbf{b}} \\ \vec{\boldsymbol{\mu}} \end{bmatrix} &= \left(\begin{bmatrix} \mathbf{I}_{2nk} | \vec{\mathbf{u}}_{\mathsf{chall},i}^{\top} \end{bmatrix} \begin{bmatrix} \vec{\mathbf{z}}_{\mathsf{aux},i}^{\top} \\ \mathbf{z}_{\mathsf{vsec},i} \end{bmatrix} \right)^{\top} . \begin{bmatrix} \vec{\mathbf{b}} \\ \vec{\boldsymbol{\mu}} \end{bmatrix} \\
&= \left(\vec{\mathbf{z}}_{\mathsf{aux},i}^{\top} + \mathbf{z}_{\mathsf{vsec},i} \vec{\mathbf{u}}_{\mathsf{chall},i}^{\top} \right)^{\top} . \begin{bmatrix} \vec{\mathbf{b}} \\ \vec{\boldsymbol{\mu}} \end{bmatrix} = \vec{\mathbf{z}}_{\mathsf{aux},i}. \begin{bmatrix} \vec{\mathbf{b}} \\ \vec{\boldsymbol{\mu}} \end{bmatrix} + \mathbf{z}_{\mathsf{vsec},i} \vec{\mathbf{u}}_{\mathsf{chall},i}. \begin{bmatrix} \vec{\mathbf{b}} \\ \vec{\boldsymbol{\mu}} \end{bmatrix} \\
&= \vec{\mathbf{z}}_{\mathsf{aux},i}. \begin{bmatrix} \vec{\mathbf{b}} \\ \vec{\boldsymbol{\mu}} \end{bmatrix} + \mathbf{z}_{\mathsf{vsec},i} \vec{\boldsymbol{\pi}}_{\mathsf{chall}}[i] \xrightarrow{+\langle \vec{\mathbf{p}}, \vec{\mathbf{c}} \rangle} \\
\langle \vec{\mathbf{p}}, \vec{\mathbf{c}} \rangle + \vec{\boldsymbol{\pi}}_{\mathsf{aux}}[i] = \langle \vec{\mathbf{p}}, \vec{\mathbf{c}} \rangle + \vec{\mathbf{z}}_{\mathsf{aux},i}. \begin{bmatrix} \vec{\mathbf{b}} \\ \vec{\boldsymbol{\mu}} \end{bmatrix} &+ \mathbf{z}_{\mathsf{vsec},i} \vec{\boldsymbol{\pi}}_{\mathsf{chall}}[i].
\end{aligned}$$

Algorithm 7 DV-LatticeIPA.Setup

1: **Input**: security parameter λ, Gaussian parameters s_z and s_p.
2: $(\mathsf{pp}, \mathsf{td}) \leftarrow \mathsf{TACS.Setup}(1^\lambda)$.
3: Set range proof challenge space as $\mathcal{K} := \left\{ \mathsf{cf}^{-1}(k) \in \mathcal{R}_q^\lambda \;\middle|\; k \in \{0,1\}^{\lambda d} \right\}$.
4: Parse $\mathsf{pp} := (\mathbf{A}_{\mathsf{com}}, \mathbf{A}_{\mathrm{hide}})$ and $\mathsf{td} := \{\mathbf{R}_i\}_{i=1}^{2k}$.
5: Set $s_{\mathsf{sec}} = s_z \sqrt{\sum_{i=0}^{\log q - 1} 4^i}$.
6: $\vec{\mathbf{u}}_{\mathsf{chall},1} \xleftarrow{\$} \mathcal{R}_q^{1\times 2nk}$, $\vec{\mathbf{p}} \leftarrow \mathcal{D}_{s_p}^{n(\delta+1)}$, $\mathbf{z}_{\mathsf{vsec},1} \leftarrow \mathcal{D}_{s_{\mathsf{sec}}}$.
7: $\vec{\mathbf{z}}_{\mathsf{aux},1} \leftarrow \mathsf{ChallTrap}(\mathsf{pp}, \mathsf{td}, \vec{\mathbf{p}}, \vec{\mathbf{u}}_{\mathsf{chall},1})$.
8: $\vec{\mathbf{u}}_{\mathsf{aux},1} \leftarrow \mathsf{Gaussian2Uniform}(\vec{\mathbf{u}}_{\mathsf{chall},1}, \mathbf{z}_{\mathsf{vsec},1}, \vec{\mathbf{z}}_{\mathsf{aux},1})$
9: Sample $\mathbf{K} := \begin{bmatrix} \vec{\mathbf{k}}_2 & \vec{\mathbf{k}}_3 & \cdots & \vec{\mathbf{k}}_{\lambda+1} \end{bmatrix} \in \mathcal{R}_q^{2nk\times\lambda} \xleftarrow{\$} \mathcal{K}^{2nk}$.
10: **for** $j = 2, \cdots, \lambda+1$ **do**
11: $\quad \mathbf{z}_{\mathsf{ksec},j-1} \leftarrow \mathcal{D}_{s_{\mathsf{sec}}}$.
12: $\quad \vec{\mathbf{z}}_{\mathsf{chall},j} \leftarrow \mathsf{ChallTrap}(\mathsf{pp}, \mathsf{td}, \vec{\mathbf{p}}, \vec{\mathbf{k}}_j^\top)$.
13: $\quad \vec{\mathbf{u}}_{\mathsf{chall},j} \leftarrow \mathsf{Gaussian2Uniform}(\vec{\mathbf{u}}_{\mathsf{chall},1}, \mathbf{z}_{\mathsf{ksec},j-1}, \vec{\mathbf{z}}_{\mathsf{chall},j})$.
14: $\quad \mathbf{z}_{\mathsf{vsec},j} \leftarrow \mathcal{D}_{s_{\mathsf{sec}}}$.
15: $\quad \vec{\mathbf{z}}_{\mathsf{aux},j} \leftarrow \mathsf{ChallTrap}(\mathsf{pp}, \mathsf{td}, \vec{\mathbf{p}}, \vec{\mathbf{u}}_{\mathsf{chall},j})$.
16: $\quad \vec{\mathbf{u}}_{\mathsf{aux},j} \leftarrow \mathsf{Gaussian2Uniform}(\vec{\mathbf{u}}_{\mathsf{chall},j}, \mathbf{z}_{\mathsf{vsec},j}, \vec{\mathbf{z}}_{\mathsf{aux},j})$.
17: **end for**
18: Set

$$\mathbf{U}_{\mathsf{chall}} := \begin{bmatrix} \vec{\mathbf{u}}_{\mathsf{chall},1} \\ \vdots \\ \vec{\mathbf{u}}_{\mathsf{chall},\lambda+1} \end{bmatrix} \mathbf{U}_{\mathsf{aux}} := \begin{bmatrix} \vec{\mathbf{u}}_{\mathsf{aux},1} \\ \vdots \\ \vec{\mathbf{u}}_{\mathsf{aux},\lambda+1} \end{bmatrix} \vec{\mathbf{z}}_{\mathsf{vsec}} := \begin{bmatrix} \mathbf{z}_{\mathsf{vsec},1} \\ \vdots \\ \mathbf{z}_{\mathsf{vsec},\lambda+1} \end{bmatrix} \vec{\mathbf{z}}_{\mathsf{ksec}} := \begin{bmatrix} \mathbf{z}_{\mathsf{ksec},1} \\ \vdots \\ \mathbf{z}_{\mathsf{ksec},\lambda} \end{bmatrix}$$

19: **Output** $\mathsf{vk} = (\vec{\mathbf{p}}, \vec{\mathbf{z}}_{\mathsf{vsec}}, \vec{\mathbf{z}}_{\mathsf{ksec}})$, $\mathsf{crs} = (\mathbf{A}_{\mathsf{com}}, \mathbf{A}_{\mathsf{hide}}, \mathbf{U}_{\mathsf{chall}}, \mathbf{U}_{\mathsf{aux}})$ and td.

Since $\vec{\mathbf{z}}_{\mathsf{aux},i}$ is generated by the $\mathsf{ChallTrap}$ algorithm using $\vec{\mathbf{u}}_{\mathsf{chall},i}$ as input (see Step 15 of Algorithm 7), Claim 2 implies that

$$\langle \vec{\mathbf{p}}, \vec{\mathbf{c}} \rangle + \vec{\mathbf{z}}_{\mathsf{aux},i} \begin{bmatrix} \vec{\mathbf{b}} \\ \vec{\boldsymbol{\mu}} \end{bmatrix} = \vec{\boldsymbol{\pi}}_{\mathsf{chall}}[i].$$

Therefore,

$$\langle \vec{\mathbf{p}}, \vec{\mathbf{c}} \rangle + \vec{\boldsymbol{\pi}}_{\mathsf{aux}}[i] = (1 + \mathbf{z}_{\mathsf{vsec},i}) \vec{\boldsymbol{\pi}}_{\mathsf{chall}}[i] = (1 + \vec{\mathbf{z}}_{\mathsf{vsec}}[i]) \vec{\boldsymbol{\pi}}_{\mathsf{chall}}[i] \quad \forall i \in [\lambda+1].$$

Next, we show that Equation (11) holds with overwhelming probability for an honest prover. Recall that $\vec{\boldsymbol{\pi}}_{\mathsf{chall}}[j]$ is computed as

$$\vec{\boldsymbol{\pi}}_{\mathsf{chall}}[j] = \vec{\mathbf{u}}_{\mathsf{chall},j} \cdot \begin{bmatrix} \vec{\mathbf{b}} \\ \vec{\boldsymbol{\mu}} \end{bmatrix}$$

According to Step 13 of Algorithm 7 together with the $\mathsf{Gaussian2Uniform}$ procedure, the vector $\vec{\mathbf{u}}_{\mathsf{chall},j}$ can be written as

$$\begin{aligned} \vec{\boldsymbol{\pi}}_{\mathsf{chall}}[j] &= (\vec{\mathbf{z}}_{\mathsf{chall},j} + \mathbf{z}_{\mathsf{ksec}}[j-1] \vec{\mathbf{u}}_{\mathsf{chall},1}) \begin{bmatrix} \vec{\mathbf{b}} \\ \vec{\boldsymbol{\mu}} \end{bmatrix} \\ &= \vec{\mathbf{z}}_{\mathsf{chall},j} \begin{bmatrix} \vec{\mathbf{b}} \\ \vec{\boldsymbol{\mu}} \end{bmatrix} + \mathbf{z}_{\mathsf{ksec}}[j-1] \vec{\boldsymbol{\pi}}_{\mathsf{chall}}[1] \quad j \in [2, \lambda+1]. \end{aligned}$$

Algorithm 8 DV-LatticeIPA.Prover

1: **Input**: common reference string crs, and the witness vector $\vec{\mathbf{b}}$.
2: Parse $\mathsf{crs} := (\mathbf{A}_{\mathsf{com}}, \mathbf{A}_{\mathsf{hide}}, \mathbf{U}_{\mathsf{chall}}, \mathbf{U}_{\mathsf{aux}})$.
3: $(\vec{\mathbf{c}}, \vec{\boldsymbol{\mu}}) \leftarrow \mathsf{TACS.Commit}((\mathbf{A}_{\mathsf{com}}, \mathbf{A}_{\mathsf{hide}}), \vec{\mathbf{b}})$.
4: Set $\vec{\boldsymbol{\pi}}_{\mathsf{chall}} = \mathbf{U}_{\mathsf{chall}} \cdot \begin{bmatrix} \vec{\mathbf{b}} \\ \vec{\boldsymbol{\mu}} \end{bmatrix} \in \mathcal{R}_q^{\lambda+1}$ and $\vec{\boldsymbol{\pi}}_{\mathsf{aux}} = \mathbf{U}_{\mathsf{aux}} \cdot \begin{bmatrix} \vec{\mathbf{b}} \\ \vec{\boldsymbol{\mu}} \end{bmatrix} \in \mathcal{R}_q^{\lambda+1}$.
5: **Output** $(\vec{\mathbf{c}}, \vec{\boldsymbol{\pi}}_{\mathsf{chall}}, \vec{\boldsymbol{\pi}}_{\mathsf{aux}})$.

Algorithm 9 DV-LatticeIPA.Verifier

1: **Input**: common reference string crs, verification key vk, the commitment $\vec{\mathbf{c}}$, and the proof vectors $\vec{\boldsymbol{\pi}}_{\mathsf{chall}}$, $\vec{\boldsymbol{\pi}}_{\mathsf{aux}}$.
2: Parse $\mathsf{vk} := (\vec{\mathbf{p}}, \vec{\mathbf{z}}_{\mathsf{vsec}}, \vec{\mathbf{z}}_{\mathsf{ksec}})$.
3: **Output** 1 if and only if:

$$\vec{\boldsymbol{\pi}}_{\mathsf{aux}} + \begin{bmatrix} \langle \vec{\mathbf{p}}, \vec{\mathbf{c}} \rangle \\ \vdots \\ \langle \vec{\mathbf{p}}, \vec{\mathbf{c}} \rangle \end{bmatrix} = \vec{\boldsymbol{\pi}}_{\mathsf{chall}} \odot \begin{bmatrix} \vec{\mathbf{z}}_{\mathsf{vsec}}[1] + 1 \\ \vdots \\ \vec{\mathbf{z}}_{\mathsf{vsec}}[\lambda+1] + 1 \end{bmatrix} \tag{10}$$

$$\|\vec{\boldsymbol{\pi}}_{\mathsf{chall}}[j] + \langle \vec{\mathbf{p}}, \vec{\mathbf{c}} \rangle - \vec{\mathbf{z}}_{\mathsf{ksec}}[j-1]\vec{\boldsymbol{\pi}}_{\mathsf{chall}}[1]\|_\infty < \frac{1}{2}\beta_{\mathsf{open}} \quad \forall j \in [2, \lambda+1] \tag{11}$$

$\odot$ means element-wise product.

Subtracting $\vec{\mathbf{z}}_{\mathsf{ksec}}[j-1]\vec{\boldsymbol{\pi}}_{\mathsf{chall}}[1]$ from both side yields:

$$\vec{\boldsymbol{\pi}}_{\mathsf{chall}}[j] - \mathbf{z}_{\mathsf{ksec}}[j-1]\vec{\boldsymbol{\pi}}_{\mathsf{chall}}[1] = \vec{\mathbf{z}}_{\mathsf{chall},j} \cdot \begin{bmatrix} \vec{\mathbf{b}} \\ \vec{\boldsymbol{\mu}} \end{bmatrix} \quad j \in [2, \lambda+1]. \tag{12}$$

The vector $\vec{\mathbf{z}}_{\mathsf{chall},j}$ is generated by applying the $\mathsf{ChallTrap}$ algorithm to the vector $\vec{\mathbf{k}}_j^\top$ (see Step 12 of Algorithm 7). Therefore, Claim 2 implies that

$$\vec{\mathbf{k}}_j^\top \cdot \begin{bmatrix} \vec{\mathbf{b}} \\ \vec{\boldsymbol{\mu}} \end{bmatrix} = \langle \vec{\mathbf{p}}, \vec{\mathbf{c}} \rangle + \vec{\mathbf{z}}_{\mathsf{chall},j} \cdot \begin{bmatrix} \vec{\mathbf{b}} \\ \vec{\boldsymbol{\mu}} \end{bmatrix}.$$

Consequently, by adding $\langle \vec{\mathbf{p}}, \vec{\mathbf{c}} \rangle$ to both sides of Equation (12), we obtain

$$\vec{\boldsymbol{\pi}}_{\mathsf{chall}}[j] - \mathbf{z}_{\mathsf{ksec}}[j-1]\vec{\boldsymbol{\pi}}_{\mathsf{chall}}[1] + \langle \vec{\mathbf{p}}, \vec{\mathbf{c}} \rangle = \vec{\mathbf{k}}_j^\top \cdot \begin{bmatrix} \vec{\mathbf{b}} \\ \vec{\boldsymbol{\mu}} \end{bmatrix} = \begin{bmatrix} \vec{\mathbf{b}} \\ \vec{\boldsymbol{\mu}} \end{bmatrix}^\top \vec{\mathbf{k}}_j \quad \forall j \in [2, \lambda+1].$$

By concatenating these λ values, we obtain

$$\begin{aligned} &\left[\vec{\boldsymbol{\pi}}_{\mathsf{chall}}[2] - \mathbf{z}_{\mathsf{ksec}}[1]\vec{\boldsymbol{\pi}}_{\mathsf{chall}}[1] + \langle \vec{\mathbf{p}}, \vec{\mathbf{c}} \rangle \quad \cdots \quad \vec{\boldsymbol{\pi}}_{\mathsf{chall}}[\lambda+1] - \mathbf{z}_{\mathsf{ksec}}[\lambda]\vec{\boldsymbol{\pi}}_{\mathsf{chall}}[1] + \langle \vec{\mathbf{p}}, \vec{\mathbf{c}} \rangle\right] \\ &= \begin{bmatrix} \vec{\mathbf{b}} \\ \vec{\boldsymbol{\mu}} \end{bmatrix}^\top \left[\vec{\mathbf{k}}_2 \quad \cdots \quad \vec{\mathbf{k}}_{\lambda+1}\right] \end{aligned} \tag{13}$$

As discussed in Section 4.3, applying the mapping $\mathsf{cf}(\cdot)$ entry-wise to Equation (13) yields the matrix product $\mathbf{MK}$, where $\mathbf{M}$ is defined in Equation (4.3). If the prover uses sufficiently short vectors, then by Lemma 3, the resulting values satisfy

$$\|\vec{\boldsymbol{\pi}}_{\mathsf{chall}}[j] + \langle \vec{\mathbf{p}}, \vec{\mathbf{c}} \rangle - \vec{\mathbf{z}}_{\mathsf{ksec}}[j-1]\vec{\boldsymbol{\pi}}_{\mathsf{chall}}[1]\|_\infty < \frac{1}{2}\beta_{\mathsf{open}}.$$

Consequently, the verifier accepts the norm check except with negligible probability. More precisely, the probability that this condition fails for any of the λ challenge vectors $\vec{\mathbf{k}}_j$ is at most $2^{-\lambda}$. □

We now prove the knowledge soundness property of DV-LatticeIPA. The proof proceeds by first establishing the coordinate-wise special soundness (CWSS) property. Knowledge soundness then follows directly from Lemma 8. The main idea underlying the proof of CWSS is to rewind the prover while fixing the commitment to the witness and providing distinct challenges that satisfy the special structure SS introduced in Section 2.5. Such a strategy requires black-box access to the prover. Given a sufficient number of accepting transcripts, or challenges and corresponding responses, the witness can then be extracted. As we need to form structural challenges, we need a programable random oracle which by programming it correctly we achieve the SS. Therefore the knowledge soundness in guaranteed in random oracle model.

We note that the extractor is granted to have access to the trapdoors generated by DV-LatticeIPA.Setup algorithm in step 2. This is because of that the extractor needs to rewind the prover over $\mathbf{U}_{\mathsf{chall}}$ and $\mathbf{U}_{\mathsf{aux}}$ challenges, therefore it needs these trapdoors.

The extraction procedure is shown in Algorithm 10. The extractor first execute the DV-LatticeIPA.Setup algorithm using a uniformly sampled challenge $\mathbf{K}^{(1)}$. Then the extractor forms the SS by sampling $\{\mathbf{K}^{(j)}\}_{j=2}^{2nk+1}$ by resampling the $j-1$ row of $\mathbf{K}^{(1)}$. This is done in a manner that

$$\mathbf{K}[j-1,:]^{(1)} - \mathbf{K}[j-1,:]^{(j)} = \begin{bmatrix} \pm X^{r_{j,1}} & \cdots & \pm X^{r_{j,\lambda}} \end{bmatrix}, 0 \leq r_{j,i} \leq d-1 \; \forall i \in [\lambda]. \quad (14)$$

For each j, the extractor computes the corresponding challenge components

$$\mathbf{U}_{\mathsf{chall}}^{(j)} \quad \text{and} \quad \mathbf{U}_{\mathsf{aux}}^{(j)},$$

and rewinds the prover using these challenges. The extraction step is then carried out according to Lemma 11, which specifies how the witness is recovered from the resulting accepting transcripts.

Lemma 11 (CWSS). *The* DV-LatticeIPA *protocol satisfies the adaptive knowledge soundness property using the extractor presented in Algorithm 10 with knowledge error* $\frac{2nk+1}{2^{\lambda d}}$.

Proof. To recover the witness, the extractor constructs the vectors $\vec{\mathbf{v}}^{(j)}$ in Step 15. It follows directly that these vectors correspond to Equation (13). Therefore,

$$\vec{\mathbf{v}}^{(j)} = \begin{bmatrix} \vec{\mathbf{b}} \\ \vec{\boldsymbol{\mu}} \end{bmatrix}^{\top} \begin{bmatrix} \vec{\mathbf{k}}_2^{(j)} & \cdots & \vec{\mathbf{k}}_{\lambda+1}^{(j)} \end{bmatrix} = \begin{bmatrix} \vec{\mathbf{b}} \\ \vec{\boldsymbol{\mu}} \end{bmatrix}^{\top} \mathbf{K}^{(j)}, \qquad \forall j \in [2nk+1].$$

The vectors $\vec{\mathbf{v}}_{\mathsf{diff}}^{(j)}$ are then defined as follows:

$$\vec{\mathbf{v}}_{\mathsf{diff}}^{(j)} = \vec{\mathbf{v}}^{(j)} - \vec{\mathbf{v}}^{(1)} = \begin{bmatrix} \vec{\mathbf{b}} \\ \vec{\boldsymbol{\mu}} \end{bmatrix}^{\top} (\mathbf{K}^{(j)} - \mathbf{K}^{(1)}) \qquad \forall j \in [2nk+1]. \quad (15)$$

By the structure of the SS set, and according to Equation (14) all entries of $\mathbf{K}^{(1)} - \mathbf{K}^{(j)}$ are equal to zero except for the $(j-1)$-th row. Consequently, Equation (15) simplifies to

$$\vec{\mathbf{v}}_{\mathsf{diff}}^{(j)} = \left(\begin{bmatrix} \vec{\mathbf{b}} \\ \vec{\boldsymbol{\mu}} \end{bmatrix} [j-1] \right) \begin{bmatrix} \pm X^{r_{j,1}} & \cdots & \pm X^{r_{j,\lambda}} \end{bmatrix} \xrightarrow{\text{step 17}}$$

$$\vec{\mathbf{b}}_{\mathsf{Ext}}^{(j)} = \left(\begin{bmatrix} \vec{\mathbf{b}} \\ \vec{\boldsymbol{\mu}} \end{bmatrix} [j-1] \right) \begin{bmatrix} 1 & \cdots & 1 \end{bmatrix} \rightarrow \vec{\mathbf{b}}_{\mathsf{Ext}}[1] = \left(\begin{bmatrix} \vec{\mathbf{b}} \\ \vec{\boldsymbol{\mu}} \end{bmatrix} [j-1] \right) \quad j \in [2, 2nk+1].$$

Therefore, each coordinate of the witness vector can be uniquely computed. Consequently, the extractor successfully reconstructs the entire witness. By Lemma 8, the knowledge error is $\kappa = \frac{2nk+1}{2^{\lambda d}}$.

Algorithm 10 DV-LatticeIPA.Extractor

Input: black-box access to adversary prover $\mathsf{DV\text{-}LatticeIPA.Prover}^*$ and ability to rewind it at any point of interest.
$\mathbf{K}^{(1)} \xleftarrow{\$} \mathcal{K}^{2nk}$.
Run $\mathsf{DV\text{-}LatticeIPA.Setup}(1^\lambda, s_z, s_p)$ using $\mathbf{K}^{(1)}$ in step 9, obtain $(\mathsf{vk}^{(1)}, \mathsf{crs}^{(1)}, \mathsf{td}^{(1)})$.
$\left(\vec{\mathbf{c}}, \vec{\boldsymbol{\pi}}^{(1)}_{\mathsf{chall}}, \vec{\boldsymbol{\pi}}^{(1)}_{\mathsf{aux}}\right) \leftarrow \mathsf{DV\text{-}LatticeIPA.Prover}^*(\mathsf{crs}^{(1)})$.
if $\mathsf{DV\text{-}LatticeIPA.Verifier}(\mathsf{vk}^{(1)}, \vec{\mathbf{c}}, \vec{\boldsymbol{\pi}}^{(1)}_{\mathsf{chall}}, \vec{\boldsymbol{\pi}}^{(1)}_{\mathsf{aux}}) = 0$ **then**
  Abort.
end if
for $j = 2, \ldots, 2nk+1$ **do**
  Sample $\mathbf{K}^{(j)}$ by resampling the $j-1$ row of $\mathbf{K}^{(1)}$ s.t Equation (14).
  Compute $\mathbf{U}^{(j)}_{\mathsf{chall}}$, $\mathbf{U}^{(j)}_{\mathsf{aux}}$ and $\mathsf{vk}^{(j)}$ according to steps 7–17 of Algorithm 7 using $\mathbf{K}^{(j)}$.
  Rewind the prover in step 4 of Algorithm 8 and obtain $\vec{\boldsymbol{\pi}}^{(j)}_{\mathsf{chall}}$ and $\vec{\boldsymbol{\pi}}^{(j)}_{\mathsf{aux}}$.
  if $\mathsf{DV\text{-}LatticeIPA.Verifier}(\mathsf{vk}^{(j)}, \vec{\mathbf{c}}, \vec{\boldsymbol{\pi}}^{(j)}_{\mathsf{chall}}, \vec{\boldsymbol{\pi}}^{(j)}_{\mathsf{aux}}) = 0$ **then**
    Abort.
  end if
  Define $\vec{\mathbf{v}}^{(j)} := \left(\vec{\boldsymbol{\pi}}^{(j)}_{\mathsf{chall}}[\ell+1] + \langle \vec{\mathbf{p}}, \vec{\mathbf{c}} \rangle - \vec{\mathbf{z}}^{(j)}_{\mathsf{ksec}}[\ell]\vec{\boldsymbol{\pi}}^{(j)}_{\mathsf{chall}}[1]\right)_{\ell=1}^{\lambda}$.
  $\vec{\mathbf{v}}^{(j)}_{\mathsf{diff}} = \vec{\mathbf{v}}^{(j)} - \vec{\mathbf{v}}^{(1)}$
  $\vec{\mathbf{b}}^{(j)}_{\mathsf{Ext}} := \begin{bmatrix} \pm X^{r_{j,1}} & \cdots & \pm X^{r_{j,\lambda}} \end{bmatrix} \odot \vec{\mathbf{v}}^{(j)}_{\mathsf{diff}}$.
end for
Set $\mathsf{SS}(\mathcal{K}, 2nk+1) = (\mathbf{K}^{(1)}, \ldots, \mathbf{K}^{(2nk+1)})$.
Output $\{\vec{\mathbf{b}}^{(j)}_{\mathsf{Ext}}[1]\}_{j=2}^{2nk+1}$

We next show that the extraction procedure remains valid in the adaptive setting, where the prover may choose the witness after observing the crs. It suffices to show that the distribution of the crs is statistically indistinguishable from a uniformly random one. The crs contains two commitment matrices, both sampled uniformly at random. It also contains the challenge matrix $\mathbf{U}_{\mathsf{chall}}$, whose first row is sampled uniformly, while the remaining rows are generated using the Gaussian2Uniform algorithm. By Claim 3, these rows are statistically close to uniform. Similarly, every row of $\mathbf{U}_{\mathsf{aux}}$ is generated using Gaussian2Uniform and is therefore statistically close to uniform. Consequently, all components of the crs are statistically indistinguishable from uniformly random matrices and vectors. Therefore, observing the crs does not provide the prover with any useful information beyond uniformly random public parameters. Hence, the extraction procedure remains valid even when the witness is chosen adaptively after the generation of the crs. □

Next, we prove the zero-knowledge property. To this end, we first establish two supporting claims.

Claim 4. *The inner product $\langle \vec{\mathbf{p}}, \vec{\mathbf{c}} \rangle$ is statistically close to uniform over $\mathcal{R}_q$, assuming that the commitment $\vec{\mathbf{c}}$ is generated honestly and Gaussian parameter of $\vec{\mathbf{p}}$, $s_{\mathbf{p}}$, satisfies $s_{\mathbf{p}} > 2d \cdot q^{\frac{1}{n(\delta+1)} + \frac{2}{dn(\delta+1)}}$.*

Proof. It was shown in [BBC+18] that honestly generated Ajtai commitment $\vec{\mathbf{c}}$ is statistically indistinguishable from a uniformly random vector over $\mathcal{R}_q^m$, where $m = n(\delta+1)$. We can view $\vec{\mathbf{c}}^\top$ as a $1 \times m$ matrix. If $\vec{\mathbf{p}}$ is sampled from a discrete Gaussian distribution with parameter $s_{\mathbf{p}}$ satisfying the given bound, then the conditions of Lemma 5 are met. As a result, the inner product $\vec{\mathbf{c}}^\top \cdot \vec{\mathbf{p}}$ is statistically close to uniform over $\mathcal{R}_q$.

□

Claim 5. *The vectors $\vec{\boldsymbol{\pi}}_{\mathsf{chall}}$ and $\vec{\boldsymbol{\pi}}_{\mathsf{aux}}$ is statistically indistinguishable from a uniformly random vector, assuming the following conditions hold: the vector $\vec{\boldsymbol{\mu}}$ is sampled from $\mathcal{D}_{\sigma}^{nk}$ where $\sigma > 2d \cdot q^{\frac{\lambda+1}{nk}+\frac{2}{dnk}}$, q is a prime such that $q \equiv 5 \pmod 8$, and $\lambda + 1 \leq nk$.*

Proof. . The matrix $\mathbf{U}_{\mathsf{chall}}$ is sampled uniformly at random. The response vector is computed as:

$$\vec{\boldsymbol{\pi}}_{\mathsf{chall}} = \mathbf{U}_{\mathsf{chall}} \cdot \begin{bmatrix} \vec{\mathbf{b}} \\ \vec{\boldsymbol{\mu}} \end{bmatrix} = \mathbf{U}_{\mathsf{chall}}^{\mathsf{up}} \cdot \vec{\mathbf{b}} + \mathbf{U}_{\mathsf{chall}}^{\mathsf{down}} \cdot \vec{\boldsymbol{\mu}}.$$

Since $\lambda + 1 \leq nk$, and $\vec{\boldsymbol{\mu}}$ is sampled from a sufficiently wide Gaussian, Lemma 5 implies that $\mathbf{U}_{\mathsf{chall}}^{\mathsf{down}} \cdot \vec{\boldsymbol{\mu}}$ is statistically close to uniform. As a result, it masks the deterministic term $\mathbf{U}_{\mathsf{chall}}^{\mathsf{up}} \cdot \vec{\mathbf{b}}$, making the entire response $\vec{\boldsymbol{\pi}}_{\mathsf{chall}}$ statistically indistinguishable from uniform. The same reasoning applies to $\vec{\boldsymbol{\pi}}_{\mathsf{aux}}$, which has an analogous structure.

□

Algorithm 11 Simulator for DV-LatticeIPA

$\underline{\mathsf{Sim}_{\mathsf{DV\text{-}LatticeIPA}}(\mathsf{crs}, \mathsf{vk})}$

1: Parse $\mathsf{vk} := (\vec{\mathbf{p}}, \vec{\mathbf{z}}_{\mathsf{vsec}}, \vec{\mathbf{z}}_{\mathsf{ksec}})$.
2: $\vec{\mathbf{c}} \xleftarrow{\$} \mathcal{R}_q^{n(\delta+1)}$.
3: $\vec{\boldsymbol{\pi}}_{\mathsf{chall}}[1] \xleftarrow{\$} \mathcal{R}_q$ and set $\vec{\boldsymbol{\pi}}_{\mathsf{aux}}[1] = (\vec{\mathbf{z}}_{\mathsf{vsec}}[1] + 1)\vec{\boldsymbol{\pi}}_{\mathsf{chall}}[1] - \langle \vec{\mathbf{p}}, \vec{\mathbf{c}} \rangle$.
4: **for** $j = 2, \cdots, \lambda + 1$ **do**
5: Choose $\boldsymbol{\pi}'_j$ s.t. $\|\boldsymbol{\pi}'_j\|_\infty \leq \frac{1}{2}\beta_{\mathsf{open}}$
6: Set $\vec{\boldsymbol{\pi}}_{\mathsf{chall}}[j] = \boldsymbol{\pi}'_j + \vec{\mathbf{z}}_{\mathsf{ksec}}[j-1]\vec{\boldsymbol{\pi}}_{\mathsf{chall}}[1] - \langle \vec{\mathbf{p}}, \vec{\mathbf{c}} \rangle$
7: Set $\vec{\boldsymbol{\pi}}_{\mathsf{aux}}[j] = (\vec{\mathbf{z}}_{\mathsf{vsec}}[j] + 1)\vec{\boldsymbol{\pi}}_{\mathsf{chall}}[j] - \langle \vec{\mathbf{p}}, \vec{\mathbf{c}} \rangle$
8: **end for**
9: **Output** $(\vec{\mathbf{c}}, \vec{\boldsymbol{\pi}}_{\mathsf{chall}}, \vec{\boldsymbol{\pi}}_{\mathsf{aux}})$

Lemma 12 (**Zero-Knowledge**). *The* DV-LatticeIPA *satisfies the zero-knowledge property, assuming that: the simulator* $\mathsf{Sim}_{\mathsf{DV\text{-}LatticeIPA}}$*, as defined in Algorithm 11, is used, q is a prime such that $q \equiv 5 \pmod 8$ and the noise vector $\vec{\boldsymbol{\mu}}$ is sampled from $\mathcal{D}_\sigma$ where $\sigma > 2d \cdot q^{\frac{1}{nk}+\frac{2}{dnk}}$.*

Proof. . According to Claim 5 and the statistical hiding property of the commitment, the three components of the transcript, $(\vec{\mathbf{c}}, \vec{\boldsymbol{\pi}}_{\mathsf{chall}}, \vec{\boldsymbol{\pi}}_{\mathsf{aux}})$, produced by the honest prover are statistically close to uniform. It suffices to show that the corresponding components generated by the simulator in Algorithm 11, are also statistically close to uniform. In that case, no distinguisher, as defined in Definition 7, can distinguish between the output of the simulator and that of an honest prover.
Commitment $\vec{\mathbf{c}}$: According to step 2 of simulator algorithm, the commitment $\vec{\mathbf{c}}$ is sampled directly from a uniform distribution over $\mathcal{R}_q^{n(\delta+1)}$.
Challenge responses $\vec{\boldsymbol{\pi}}_{\mathsf{chall}}$: In step 3, the first entry $\vec{\boldsymbol{\pi}}_{\mathsf{chall}}[1]$ is sampled uniformly at random over $\mathcal{R}_q$. The remaining entries are computed as:

$$\vec{\boldsymbol{\pi}}_{\mathsf{chall}}[j] = \boldsymbol{\pi}'_j + \vec{\mathbf{z}}_{\mathrm{ksec}}[j-1] \cdot \vec{\boldsymbol{\pi}}_{\mathsf{chall}}[1] - \langle \vec{\mathbf{p}}, \vec{\mathbf{c}} \rangle \quad \forall j \in [2, \lambda + 1],$$

where $\boldsymbol{\pi}'_j$ is a fixed (deterministic) value and does not increase entropy. The inner product $\langle \vec{\mathbf{p}}, \vec{\mathbf{c}} \rangle$ is statistically close to uniform by Claim 4. It remains to show that $\vec{\mathbf{z}}_{\mathsf{ksec}}[j-1] \cdot \vec{\boldsymbol{\pi}}_{\mathsf{chall}}[1]$ is statistically close to uniform. Since $\vec{\boldsymbol{\pi}}_{\mathsf{chall}}[1]$ is uniformly distributed and can be viewed as a 1×1 matrix, and $\vec{\mathbf{z}}_{\mathsf{ksec}}[j-1]$ is sampled from a wide enough Gaussian distribution, Lemma 5 applies to it. Therefore, this term is statistically close to uniform, and so is each $\vec{\boldsymbol{\pi}}_{\mathsf{chall}}[j]$.

Auxiliary responses $\vec{\boldsymbol{\pi}}_{\mathsf{aux}}$: An identical argument applies to the entries of $\vec{\boldsymbol{\pi}}_{\mathsf{aux}}$, which are constructed in the same form under the same conditions.

Therefore, all components generated by the simulator are statistically close to uniform, and thus indistinguishable from those in a real protocol execution. □

5 A DV-zkSNARK Based on the DV-LatticeIPA

In this section, we present a new lattice-based DV-zkSNARK construction derived from the DV-LatticeIPA protocol. Previous lattice-based DV-zkSNARK constructions rely on the LPCP compiler of [ISW21]. The compiler involves four challenge vectors, represented by the matrix

$$Q_{\mathsf{LPCP}} \in \mathbb{Z}_q^{4\times nk},$$

for which the prover must compute the corresponding inner products by the witness. To achieve the zero-knowledge property, prior works [SSE+24, ISW21] encrypt these challenges using homomorphic encryption and embed the resulting ciphertexts into the CRS. Our construction avoids the use of homomorphic encryption altogether. Instead, we exploit the fact that the LPCP compiler is naturally compatible with inner-product arguments and incorporate it directly into the DV-LatticeIPA framework. More precisely, the vectors $\vec{\mathbf{k}}_i$ used in Figure 3 can be replaced by the rows of Q_{LPCP}. As a result, the LPCP challenges are hidden by the challenge randomization mechanism of DV-LatticeIPA, and the zero-knowledge property follows directly from the honest-verifier zero-knowledge property of the underlying DV-LatticeIPA protocol.

The resulting construction is presented in Algorithms 12–14. At a high level, the protocol defines an argument system in which the prover demonstrates that the witness satisfies the LPCP constraints while simultaneously proving that the witness has sufficiently small norm.

During the setup phase, shown in Algorithm 12, the protocol first executes the setup procedure of DV-LatticeIPA. Next, in Step 4, the matrix Q_{LPCP} is encoded using the ring–field isomorphism introduced in Section 2.6. This transformation converts the LPCP challenges from field elements into ring elements, thereby making them compatible with the DV-LatticeIPA protocol, which operates over cyclotomic rings. Subsequently, in step 5, a zero matrix of dimension $4 \times nk$ is appended to the encoded matrix $\mathbf{Q}_{\mathsf{LPCP}}$. The resulting matrix is constructed so that it can be multiplied by $\begin{bmatrix}\vec{\mathbf{b}}\\ \vec{\boldsymbol{\mu}}\end{bmatrix}$, where $\vec{\mathbf{b}}$ denotes the actual witness and $\vec{\boldsymbol{\mu}}$ is a masking vector used solely to provide statistical hiding. Since the LPCP constraints depend only on $\vec{\mathbf{b}}$, the columns corresponding to $\vec{\boldsymbol{\mu}}$ are set to zero. Finally, Steps 6–13 of Algorithm 12 process the rows of $\mathbf{Q}_{\mathsf{modified}}$ using the challenge-randomization mechanism of DV-LatticeIPA, which transforms them into vectors that are statistically close to uniform. These vectors constitute the hidden LPCP challenges embedded in the CRS.

The prover first encodes each entry of the witness vector using the same encoding applied to $\mathbf{Q}_{\mathsf{LPCP}}$ in step 3 of Algorithm 13. This encoding ensures compatibility with ring arithmetic. Moreover, since the encoding defines an isomorphism, the resulting inner products between the encoded witness and the encoded challenge vectors can be decoded back into the field while preserving their algebraic structure. The prover then executes the DV-LatticeIPA.Prover algorithm to generate the proof components corresponding to the norm-checking procedure, and subsequently computes the proof components associated with the LPCP challenges.

Upon receiving the proof vectors and the verification key, the verifier first checks the validity of the proof components corresponding to the DV-LatticeIPA protocol. If these

Algorithm 12 DV-zkSNARK.Setup

Input: security parameter λ, Gaussian parameters s_z, and s_p.
$(\mathsf{vk}_{\mathsf{IPA}}, \mathsf{crs}_{\mathsf{IPA}}, \mathsf{td}_{\mathsf{IPA}}) \leftarrow \mathsf{DV\text{-}LatticeIPA.Setup}(1^\lambda, s_z, s_p)$.
Set $s_{\mathsf{sec}} = s_z \sum_{i=0}^{(\log q)-1} 2^i$.
Encode each entry of $Q_{\mathsf{LPCP}}^\top$ using $\mathsf{Encode}(.)$ (Section 2.6) and output $\mathbf{Q}_{\mathsf{LPCP}}^\top$. ▷ $Q_{\mathsf{LPCP}}^\top \in \mathbb{Z}_q^{4\times nk}$ and $\mathbf{Q}_{\mathsf{LPCP}}^\top \in \mathcal{R}_q^{4\times nk}$.
Set $\mathbf{Q}_{\mathsf{modified}}^\top := \begin{bmatrix}\mathbf{Q}_{\mathsf{LPCP}}^\top & \mathbf{0}^{4\times nk}\end{bmatrix}$.
for $r = 1, \cdots, 4$ **do**
  $\mathbf{z}_{\mathsf{qsec},r} \leftarrow \mathcal{D}_{s_{\mathsf{sec}}}$.
  $\vec{\mathbf{z}}_{\mathsf{chall},r} \leftarrow \mathsf{ChallTrap}(\mathsf{pp}, \mathsf{td}, \vec{\mathbf{p}}, \vec{\mathbf{q}}_r)$. ▷ $\vec{\mathbf{q}}_r$ is the rth row of $\mathbf{Q}_{\mathsf{modified}}^\top$.
  $\vec{\mathbf{q}}_{\mathsf{chall},r} \leftarrow \mathsf{Gaussian2Uniform}(\vec{\mathbf{u}}_{\mathsf{chall},1}, \mathbf{z}_{\mathsf{qsec},r}, \vec{\mathbf{z}}_{\mathsf{chall},r})$.
  $\mathbf{z}_{\mathsf{vqsec},r} \leftarrow \mathcal{D}_{s_{\mathsf{sec}}}$.
  $\vec{\mathbf{z}}_{\mathsf{aux},r} \leftarrow \mathsf{ChallTrap}(\mathsf{pp}, \mathsf{td}, \vec{\mathbf{p}}, \vec{\mathbf{q}}_{\mathsf{chall},r})$.
  $\vec{\mathbf{q}}_{\mathsf{aux},r} \leftarrow \mathsf{Gaussian2Uniform}(\vec{\mathbf{q}}_{\mathsf{chall},r}, \mathbf{z}_{\mathsf{vqsec},r}, \vec{\mathbf{z}}_{\mathsf{aux},r})$.
end for
Set

$$\mathbf{Q}_{\mathsf{chall}} := \begin{bmatrix}\vec{\mathbf{q}}_{\mathsf{chall},1}\\ \vdots \\ \vec{\mathbf{q}}_{\mathsf{chall},4}\end{bmatrix} \quad \mathbf{Q}_{\mathsf{aux}} := \begin{bmatrix}\vec{\mathbf{q}}_{\mathsf{aux},1}\\ \vdots \\ \vec{\mathbf{q}}_{\mathsf{aux},4}\end{bmatrix} \quad \vec{\mathbf{z}}_{\mathsf{qsec}} := \begin{bmatrix}\mathbf{z}_{\mathsf{qsec},1}\\ \vdots \\ \mathbf{z}_{\mathsf{qsec},4}\end{bmatrix} \quad \vec{\mathbf{z}}_{\mathsf{vqsec}} := \begin{bmatrix}\mathbf{z}_{\mathsf{vqsec},1}\\ \vdots \\ \mathbf{z}_{\mathsf{vqsec},4}\end{bmatrix}$$

Output $\mathsf{crs} := (\mathsf{crs}_{\mathsf{IPA}}, \mathbf{Q}_{\mathsf{chall}}, \mathbf{Q}_{\mathsf{aux}})$ and $\mathsf{vk} := (\mathsf{vk}_{\mathsf{IPA}}, \vec{\mathbf{z}}_{\mathsf{qsec}}, \vec{\mathbf{z}}_{\mathsf{vqsec}})$

Algorithm 13 DV-zkSNARK.Prover

Input: common reference string, crs, witness vector correspond to LPCP proof system, $\vec{b} \in \mathbb{Z}_q^{nk}$.
Parse $\mathsf{crs} := (\mathsf{crs}_{\mathsf{IPA}}, \mathbf{Q}_{\mathsf{chall}}, \mathbf{Q}_{\mathsf{aux}})$.
For each entry in vector $\vec{b}$, run the Encode algorithm and compute $\vec{\mathbf{b}}$ from it.
$(\vec{\mathbf{c}}, \vec{\boldsymbol{\pi}}_{\mathsf{chall}}, \vec{\boldsymbol{\pi}}_{\mathsf{aux}}) \leftarrow \mathsf{Lattice\text{-}IPA.Prover}(\mathsf{pp}, \mathsf{crs}_{\mathsf{IPA}}, \vec{\mathbf{b}})$.
Set $\vec{\boldsymbol{\pi}}_{Q_{\mathsf{chall}}} := \mathbf{Q}_{\mathsf{chall}} \begin{bmatrix}\vec{\mathbf{b}}\\ \vec{\boldsymbol{\mu}}\end{bmatrix}$ and $\vec{\boldsymbol{\pi}}_{Q_{\mathsf{aux}}} = \mathbf{Q}_{\mathsf{aux}} \begin{bmatrix}\vec{\mathbf{b}}\\ \vec{\boldsymbol{\mu}}\end{bmatrix}$.
Output $(\vec{\mathbf{c}}, \vec{\boldsymbol{\pi}}_{\mathsf{chall}}, \vec{\boldsymbol{\pi}}_{\mathsf{aux}}, \vec{\boldsymbol{\pi}}_{Q_{\mathsf{chall}}}, \vec{\boldsymbol{\pi}}_{Q_{\mathsf{aux}}})$.

checks pass, the verifier then verifies that the following equality holds for $\vec{\boldsymbol{\pi}}_{Q_{\mathsf{aux}}}$:

$$\vec{\boldsymbol{\pi}}_{Q_{\mathsf{aux}}} = \mathbf{Q}_{\mathsf{aux}} \begin{bmatrix}\vec{\mathbf{b}}\\ \vec{\boldsymbol{\mu}}\end{bmatrix} = \begin{bmatrix}(\vec{\mathbf{z}}_{\mathsf{vqsec}}[1]+1)\vec{\boldsymbol{\pi}}_{Q_{\mathsf{chall}}}[1] - \langle \vec{\mathbf{p}}, \vec{\mathbf{c}}\rangle \\ \vdots \\ (\vec{\mathbf{z}}_{\mathsf{vqsec}}[4]+1)\vec{\boldsymbol{\pi}}_{Q_{\mathsf{chall}}}[4] - \langle \vec{\mathbf{p}}, \vec{\mathbf{c}}\rangle\end{bmatrix}. \tag{16}$$

The verifier then computes $\vec{\boldsymbol{\pi}}_{\mathsf{LPCP}}$ and applies the decoding algorithm Dcd to obtain the vector $\vec{\pi}_{\mathsf{LPCP}}$. Finally, the verifier executes the same verification procedure as in the LPCP compiler using $\vec{\pi}_{\mathsf{LPCP}}$.

Theorem 1. *The* DV-zkSNARK *construction described in Algorithms 12–14 satisfies completeness.*

Proof. The verifier accepts the proof if the following conditions hold:

1. The verification procedure of DV-LatticeIPA succeeds. This follows directly from Lemma 10.

Algorithm 14 DV-zkSNARK.Verifier

Input: common reference string crs, verification key vk, commitment $\vec{\mathbf{c}}$, and proof vectors $\vec{\boldsymbol{\pi}}_{\mathsf{chall}}$, $\vec{\boldsymbol{\pi}}_{\mathsf{aux}}$, $\vec{\boldsymbol{\pi}}_{Q_{\mathsf{chall}}}$, $\vec{\boldsymbol{\pi}}_{Q_{\mathsf{aux}}}$.
$\mathsf{bit} \leftarrow \mathsf{DV\text{-}LatticeIPA.Verifier}(\mathsf{crs}, \mathsf{vk}, \vec{\mathbf{c}}, \vec{\boldsymbol{\pi}}_{\mathsf{chall}}, \vec{\boldsymbol{\pi}}_{\mathsf{aux}})$
if $\mathsf{bit} = 1$ **then**
  Output 0 if Equation (16) does not hold.
  Compute $\vec{\boldsymbol{\pi}}_{\mathsf{LPCP}} = \vec{\boldsymbol{\pi}}_{Q_{\mathsf{chall}}} + \begin{bmatrix} \langle \vec{\mathbf{p}}, \vec{\mathbf{c}} \rangle - \vec{\mathbf{z}}_{\mathsf{qsec}}[1]\, \vec{\boldsymbol{\pi}}_{\mathsf{chall}}[1] \\ \vdots \\ \langle \vec{\mathbf{p}}, \vec{\mathbf{c}} \rangle - \vec{\mathbf{z}}_{\mathsf{qsec}}[4]\, \vec{\boldsymbol{\pi}}_{\mathsf{chall}}[1] \end{bmatrix}$.
  Set $\vec{\pi}_{\mathsf{LPCP}} = \mathsf{Decode}(\vec{\boldsymbol{\pi}})$.
  $\mathsf{bit}' \leftarrow \mathsf{LPCP.Verifier}(\vec{\pi}_{\mathsf{LPCP}})$.
  Output bit'.
else
  Output 0.
end if

2. Equation (16) is satisfied. We now show that this equation holds whenever the prover behaves honestly. In particular, consider each row of $\vec{\boldsymbol{\pi}}_{Q_{\mathsf{aux}}}$:

$$\vec{\boldsymbol{\pi}}_{Q_{\mathsf{aux}}}[i] = \mathbf{Q}_{\mathsf{aux}} \begin{bmatrix} \vec{\mathbf{b}} \\ \vec{\boldsymbol{\mu}} \end{bmatrix}[i] = \vec{\mathbf{q}}_{\mathsf{aux},i} \begin{bmatrix} \vec{\mathbf{b}} \\ \vec{\boldsymbol{\mu}} \end{bmatrix}, \quad \forall i \in [4]. \tag{17}$$

According to step 12 of Algorithm 12 and the $\mathsf{Gaussian2Uniform}$ procedure, we have

$$\vec{\mathbf{q}}_{\mathsf{aux},i} = \vec{\mathbf{z}}_{\mathsf{aux},i} + \mathbf{z}_{\mathsf{vqsec},i} \vec{\mathbf{q}}_{\mathsf{chall},i}. \tag{18}$$

Substituting Equation (18) into Equation (17) yields

$$\vec{\mathbf{q}}_{\mathsf{aux},i} \begin{bmatrix} \vec{\mathbf{b}} \\ \vec{\boldsymbol{\mu}} \end{bmatrix} = (\vec{\mathbf{z}}_{\mathsf{aux},i} + \mathbf{z}_{\mathsf{vqsec},i} \vec{\mathbf{q}}_{\mathsf{chall},i}) \begin{bmatrix} \vec{\mathbf{b}} \\ \vec{\boldsymbol{\mu}} \end{bmatrix} = \vec{\mathbf{z}}_{\mathsf{aux},i} \begin{bmatrix} \vec{\mathbf{b}} \\ \vec{\boldsymbol{\mu}} \end{bmatrix} + \mathbf{z}_{\mathsf{vqsec},i} \vec{\boldsymbol{\pi}}_{Q_{\mathsf{chall}}}[i]. \tag{19}$$

Furthermore, by step 11 of Algorithm 12 together with Claim 2, we obtain

$$\vec{\mathbf{z}}_{\mathsf{aux},i} \begin{bmatrix} \vec{\mathbf{b}} \\ \vec{\boldsymbol{\mu}} \end{bmatrix} + \langle \vec{\mathbf{p}}, \vec{\mathbf{c}} \rangle = \vec{\boldsymbol{\pi}}_{Q_{\mathsf{chall}}}[i].$$

Adding $\langle \vec{\mathbf{p}}, \vec{\mathbf{c}} \rangle$ to both sides of Equation (19) gives

$$\vec{\mathbf{q}}_{\mathsf{aux},i} \begin{bmatrix} \vec{\mathbf{b}} \\ \vec{\boldsymbol{\mu}} \end{bmatrix} + \langle \vec{\mathbf{p}}, \vec{\mathbf{c}} \rangle = (\mathbf{z}_{\mathsf{vqsec},i} + 1) \vec{\boldsymbol{\pi}}_{Q_{\mathsf{chall}}}[i] = (\mathbf{z}_{\mathsf{vqsec}}[i] + 1) \vec{\boldsymbol{\pi}}_{Q_{\mathsf{chall}}}[i] \quad \forall i \in [4],$$

which coincides exactly with the i-th row of Equation (16).

3. If Equation (16) holds, the verifier proceeds to check the validity of the LPCP proof. We now show that for an honest prover, the vector $\vec{\pi}_{\mathsf{LPCP}}$ would pass the LPCP verification. Each entry of the encoded vector $\vec{\boldsymbol{\pi}}_{\mathsf{LPCP}}$ from step 5 of Algorithm 14 is computed as

$$\vec{\boldsymbol{\pi}}_{\mathsf{LPCP}}[i] = \vec{\mathbf{q}}_{\mathsf{chall},i} \begin{bmatrix} \vec{\mathbf{b}} \\ \vec{\boldsymbol{\mu}} \end{bmatrix} + \langle \vec{\mathbf{p}}, \vec{\mathbf{c}} \rangle - \vec{\mathbf{z}}_{\mathsf{qsec}}[i]\, \vec{\boldsymbol{\pi}}_{\mathsf{chall}}[1]. \tag{20}$$

By step 9 of Algorithm 12 and the $\mathsf{Gaussian2Uniform}$ procedure, we have

$$\vec{\mathbf{q}}_{\mathsf{chall},i} = \vec{\mathbf{z}}_{\mathsf{chall},i} + \vec{\mathbf{z}}_{\mathsf{qsec}}[i]\, \vec{\mathbf{u}}_{\mathsf{chall},1}.$$

Substituting this expression into Equation (20) yields

$$\begin{aligned}\vec{\boldsymbol{\pi}}_{\mathsf{LPCP}}[i] &= (\vec{\mathbf{z}}_{\mathsf{chall},i} + \vec{\mathbf{z}}_{\mathsf{qsec}}[i]\,\vec{\mathbf{u}}_{\mathsf{chall},1}) \begin{bmatrix}\vec{\mathbf{b}}\\ \vec{\boldsymbol{\mu}}\end{bmatrix} + \langle \vec{\mathbf{p}}, \vec{\mathbf{c}}\rangle - \vec{\mathbf{z}}_{\mathsf{qsec}}[i]\,\vec{\boldsymbol{\pi}}_{\mathsf{chall}}[1] \\ &= \vec{\mathbf{z}}_{\mathsf{chall},i} \begin{bmatrix}\vec{\mathbf{b}}\\ \vec{\boldsymbol{\mu}}\end{bmatrix} + \vec{\mathbf{z}}_{\mathsf{qsec}}[i]\,\vec{\boldsymbol{\pi}}_{\mathsf{chall}}[1] + \langle \vec{\mathbf{p}}, \vec{\mathbf{c}}\rangle - \vec{\mathbf{z}}_{\mathsf{qsec}}[i]\,\vec{\boldsymbol{\pi}}_{\mathsf{chall}}[1] \\ &= \vec{\mathbf{z}}_{\mathsf{chall},i} \begin{bmatrix}\vec{\mathbf{b}}\\ \vec{\boldsymbol{\mu}}\end{bmatrix} + \langle \vec{\mathbf{p}}, \vec{\mathbf{c}}\rangle \quad \forall i \in [4] \end{aligned} \tag{21}$$

Since the vectors $\vec{\mathbf{z}}_{\mathsf{chall},i}$ are generated by applying the ChallTrap algorithm to the vectors $\vec{\mathbf{q}}_i$ (see Step 8 of Algorithm 12), Claim 2 can be applied to Equation (21). Consequently, we obtain

$$\vec{\boldsymbol{\pi}}_{\mathsf{LPCP}}[i] = \vec{\mathbf{q}}_i \begin{bmatrix}\vec{\mathbf{b}}\\ \vec{\boldsymbol{\mu}}\end{bmatrix} = \begin{bmatrix}\vec{\mathbf{q}}_{\mathsf{LPCP},i} & \vec{\mathbf{0}}^{nk}\end{bmatrix} \begin{bmatrix}\vec{\mathbf{b}}\\ \vec{\boldsymbol{\mu}}\end{bmatrix} = \vec{\mathbf{q}}_{\mathsf{LPCP},i}.\vec{\mathbf{b}}.$$

By decoding the values $\vec{\boldsymbol{\pi}}[i]$ and applying the isomorphism introduced in Section 2.6, we obtain

$$\mathsf{Decode}(\vec{\boldsymbol{\pi}}[i]) = \vec{q}_{\mathsf{LPCP},i} \cdot \vec{b}.$$

This corresponds exactly to a valid transcript for the LPCP compiler. Therefore, by the completeness property of the LPCP protocol, the verifier accepts the proof with overwhelming probability.

We next analyze the completeness error probability. The verifier rejects only if either the DV-LatticeIPA proof verification fails or the LPCP verification fails. Therefore, the completeness error is bounded by

$$\Pr[\mathsf{bit} = 0] + \Pr[\mathsf{bit}' = 0 \mid \mathsf{bit} = 1] \leq 2^{-\lambda} + \varepsilon_{\mathsf{LPCP}}(\lambda),$$

where $\varepsilon_{\mathsf{LPCP}}(\lambda)$ denotes the completeness error of the LPCP compiler. □

Theorem 2. *The* DV-zkSNARK *construction described in Algorithms 12–14 satisfies knowledge soundness.*

Proof. . The extractor is given black-box access to the adversarial prover $\mathcal{A}$ and is allowed to rewind it at arbitrary points during the interaction. In particular, the protocol retains the $\mathbf{U}_{\mathsf{aux}}$ challenges inherited from the DV-LatticeIPA protocol. Therefore, by applying Lemma 11, the extractor can rewind the prover with respect to these challenges and successfully extract the witness. Consequently, the proposed DV-zkSNARK satisfies the coordinate-wise special soundness property. Furthermore, by applying Lemma 8, it also satisfies the knowledge soundness property. □

Theorem 3. *Assume that both the* LPCP *compiler and the* DV-LatticeIPA *protocol satisfy the honest-verifier zero-knowledge (HVZK) property. Then, the proposed* DV-zkSNARK *also satisfies the HVZK property.*

Proof. . We construct a simulator for the DV-zkSNARK protocol using black-box access to the simulators of DV-LatticeIPA and LPCP. The simulator is described in Algorithm 15.

By Lemma 12, the simulated tuple $(\vec{\mathbf{c}}, \vec{\boldsymbol{\pi}}_{\mathsf{chall}}, \vec{\boldsymbol{\pi}}_{\mathsf{aux}})$ is computationally indistinguishable from the corresponding tuple generated during a real protocol execution. Moreover, since the LPCP compiler satisfies the HVZK property, the simulated vector $\vec{a}$ is computationally indistinguishable from a real execution, and consequently, so is its encoding.

Using the same argument as in Lemma 12, the vectors $\vec{\boldsymbol{\pi}}_{Q_{\mathsf{chall}}}$ and $\vec{\boldsymbol{\pi}}_{Q_{\mathsf{aux}}}$ are distributed uniformly over the corresponding space. Therefore, their simulated distributions are computationally indistinguishable from those produced in a real protocol execution. It follows that the output of $\mathsf{Sim}_{\mathsf{DV\text{-}zkSNARK}}$ is computationally indistinguishable from the view of an honest verifier interacting with the real prover. Hence, the DV-zkSNARK protocol satisfies the HVZK property. □

Algorithm 15 Simulator for DV-zkSNARK

$\mathsf{Sim}_{\mathsf{DV\text{-}zkSNARK}}(\mathsf{crs}, \mathsf{vk}, \mathsf{Sim}_{\mathsf{LPCP}}, \mathsf{Sim}_{\mathsf{DV\text{-}LatticeIPA}})$

1: Parse $\mathsf{vk} := (\vec{\mathbf{p}}, \vec{\mathbf{z}}_{\mathsf{vsec}}, \vec{\mathbf{z}}_{\mathsf{ksec}}, \vec{\mathbf{z}}_{\mathsf{qsec}}, \vec{\mathbf{z}}_{\mathsf{vqsec}})$.
2: $(\vec{\mathbf{c}}, \vec{\boldsymbol{\pi}}_{\mathsf{chall}}, \vec{\boldsymbol{\pi}}_{\mathsf{aux}}) \leftarrow \mathsf{Sim}_{\mathsf{DV\text{-}LatticeIPA}}(\mathsf{crs}, \mathsf{vk})$
3: $\vec{\pi}_{\mathsf{LPCP}} \leftarrow \mathsf{Sim}_{\mathsf{LPCP}}(\mathsf{crs})$
4: **for** $j = 1, \cdots, 4$ **do**
5: Set $\vec{\boldsymbol{\pi}}_{Q_{\mathsf{chall}}}[j] = \mathsf{Encode}(\vec{\pi}_{\mathsf{LPCP}}[j]) + \vec{\mathbf{z}}_{\mathsf{qsec}}[j]\vec{\boldsymbol{\pi}}_{\mathsf{chall}}[1] - \langle \vec{\mathbf{p}}, \vec{\mathbf{c}} \rangle$
6: Set $\vec{\boldsymbol{\pi}}_{Q_{\mathsf{aux}}}[j] = (\vec{\mathbf{z}}_{\mathsf{qvsec}}[j] + 1)\vec{\boldsymbol{\pi}}_{Q_{\mathsf{chall}}}[j] - \langle \vec{\mathbf{p}}, \vec{\mathbf{c}} \rangle$
7: **end for**
8: **Output** $(\vec{\mathbf{c}}, \vec{\boldsymbol{\pi}}_{\mathsf{chall}}, \vec{\boldsymbol{\pi}}_{\mathsf{aux}}, \vec{\boldsymbol{\pi}}_{Q_{\mathsf{chall}}}, \vec{\boldsymbol{\pi}}_{Q_{\mathsf{aux}}})$

6 Implementation and Evaluation

The proposed scheme is implemented in C language[2] and compiled with `clang` 14.0.0. It builds on the publicly available code from [BEP+21], which provides trapdoor sampling routines over ring arithmetic.

Note that our work can only be compared with [SSE+24], as other schemes have been broken [DAFS24]. The benchmark results in Figure 4 is obtained by running both implementations on the same machine, running Ubuntu 22.04.5 LTS with a 4-core Intel(R) Core(TM) i7-12700H processor, 8 GB of RAM, and a 60 GB hard disk.

Prior to implementation, we determine suitable parameter settings to achieve 128-bit security. This security level is chosen to facilitate comparison with previous works. Higher security levels, such as 256-bit security, can be attained by increasing the parameter sizes accordingly.

For this purpose, we use the estimation tool developed by Albrecht et al. [APS15], which allows selecting parameters such that solving the underlying MSIS instance requires at least 2^{128} operations.

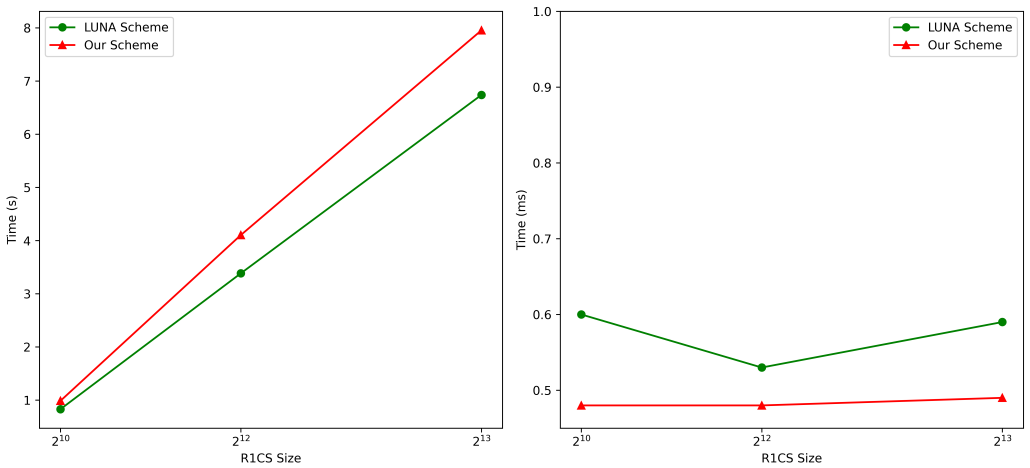
Table 2 shows the parameters we have used in our implementation, and their selected values.

Table 2: List of common parameters and their description.

Parameters	Param Value	Description	Remarks
q	$\approx 2^{32}$	ring modulus	
d	16	ring dimension	
n	2, 4, 4	dimension of trapdoors	
k	2^9, 2^{10}, 2^{11}	number of trapdoors	
N_w	2^{10}, 2^{12}, 2^{13}	the witness length	$N_w = nk$
β_{open}	2^{16}	witness l_∞ norm bound	

The parameter β_{open} is chosen to ensure that the MSIS problem remains hard for the commitment matrices. Based on our estimation, setting $\beta_{\mathsf{open}} = 2^{16}$ yields at least 128-bits security for witness lengths ranging from 2^{10} to 2^{13}. Due to the isomorphism between ring and field described in Section 2.6, the value $(2\beta_{\mathsf{open}} - 2)^r$, where $r|d$, corresponds to the field size. Consequently, the soundness of the underlying LPCP protocol is bounded by $\Omega(\frac{1}{\beta_{\mathsf{open}}^r})$. By choosing $r = d$ the soundness of the LPCP protocol becomes approximately 2^{-256}. We set $n = 2$ for the trapdoor dimension, which yields trapdoor matrices in $\mathcal{R}_q^{2\times 66}$. To ensure security, the variance of the vectors $\vec{\mathbf{p}}$ and $\vec{\mathbf{z}}$ must be chosen sufficiently small, typically less than 2^{10}. This ensures that finding short pre-images for matrices of size 2×66 remains computationally hard, and only an entity possessing the corresponding

[2] https://github.com/crypolover1998/LatticeBased-Dv-zkSNARK



(a) Performance of Prover for Different R1CS Size.

(b) Performance of Verifier for Different R1CS Sizes.

Figure 4: Performance comparison of our scheme, and LUNA [SSE+24].

trapdoor can efficiently compute such pre-images. The parameter k is then determined directly by fixing the witness length and the value of n.

Table 3: Proof and CRS size for $\lambda = 128$.

Metric	**This Work**	**LUNA** [SSE+24]
Proof Size (KB)	20.75	8.3
Full CRS Size (GB)	1.3	11

Table 3 presents a comparison of proof and CRS sizes. In our construction, the proof size includes both the commitment and the responses to the verifier's challenges size. Our CRS size is significantly smaller than that of LUNA. This improvement is primarily due to two factors: First, earlier constructions often rely on homomorphic encryption, whose outputs are inherently large. Second, to achieve the desired level of soundness, these protocols typically require multiple repetitions, further increasing the CRS size. In contrast, our construction avoids these overheads.

We also compare our construction with LUNA in terms of execution time, as shown in Figure 4. Since LUNA is not a complete SNARK, we provide an approximate comparison based on analogous components. On the prover side, we exclude LPCP prover time for both schemes. For LUNA, we measure homomorphic addition and CRS expansion times. As CRS expansion is not implemented in LUNA, we implemented it using AES as a PRF to simulate seed-to-public-key expansion and incorporated this cost into the additive homomorphism execution time. In our scheme, the prover's time includes only commitment generation and responses to verifier challenges. On the verifier side, we exclude LPCP verification time since both schemes uses LPCP compiler and compare LUNA's verification time, which consists of decryption of prover responses, with verification time of our proposed scheme. Due to our hardware limits, we could simulate LUNA only for circuits up to size 2^{13}, restricting our comparison to this range. We simulate the prover and verifier with fewer number of challenges, then extrapolate the total execution time, noting that core operations scale linearly with the number of challenges. The main bottleneck in our protocol is the range proof, which requires 2λ challenge vectors and the corresponding responses, which results in an increase in the overhead. However, the protocol remains succinct and practical.

7 Conclusions and Future Works

DV-zkSNARKs retain the broad applicability of traditional zkSNARKs while enabling verification restricted to a designated party. They have applications in areas such as verifiable computations and anonymous credentials. To the best of our knowledge, this work presents the first lattice-based DV-zkSNARK construction from the standard Module-SIS assumption which is more efficient than other variants of SIS problem. Prior lattice-based approaches typically rely on non-standard assumptions, such as linear targeted malleability, which are not well-established. Furthermore, these approaches ensure knowledge soundness only under non-adaptive conditions. In contrast, our scheme achieves adaptive knowledge soundness in the random oracle model by leveraging coordinate-wise special soundness. This approach to proving knowledge soundness is inherently confined to the random oracle model.

A key technical departure from prior works is that, unlike previous works that rely on homomorphic encryption, our construction employs a new inner-product argument combined with a modified Ajtai commitment scheme. The inner-product argument itself may be of independent interest.

Our implementation demonstrates that the prover and verifier achieve efficiency comparable to the best existing works, while the size of the CRS is reduced by a factor of 10. However, the commitment and proof sizes are approximately 2.5 times larger, primarily due to our reliance on standard assumptions. An important direction for future research is therefore to reduce these sizes, which would further improve the practicality of our scheme.

Acknowledgment

We have used ChatGPT solely for grammatical refinement and language editing. No technical content, including proofs or constructions, was generated with its assistance. We declare that the submitted manuscript received no financial or non-financial support from any public or private organization. We further declare that we have no relevant competing financial or non-financial interests to disclose.

References

[ABVPS24] Shahla Atapoor, Karim Baghery, Hilder VL Pereira, and Jannik Spiessens. Verifiable FHE via lattice-based SNARKs. *IACR Communications in Cryptology (CiC)*, 1(1), 2024.

[ACL+22] Martin R Albrecht, Valerio Cini, Russell WF Lai, Giulio Malavolta, and Sri AravindaKrishnan Thyagarajan. Lattice-based SNARKs: publicly verifiable, preprocessing, and recursively composable. In *Annual International Cryptology Conference*, pages 102–132. Springer, 2022.

[ADI25] Gal Arnon, Jesko Dujmovic, and Yuval Ishai. Designated-verifier SNARGs with one group element. In *Annual International Cryptology Conference*, pages 192–224. Springer, 2025.

[AFLN24] Martin R Albrecht, Giacomo Fenzi, Oleksandra Lapiha, and Ngoc Khanh Nguyen. SLAP: succinct lattice-based polynomial commitments from standard assumptions. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 90–119. Springer, 2024.

[APS15] 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.

[BBC+18] Carsten Baum, Jonathan Bootle, Andrea Cerulli, Rafaël Del Pino, Jens Groth, and Vadim Lyubashevsky. Sub-linear lattice-based zero-knowledge arguments for arithmetic circuits. In *Annual International Cryptology Conference*, pages 669–699. Springer, 2018.

[BCI+13] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Omer Paneth, and Rafail Ostrovsky. Succinct non-interactive arguments via linear interactive proofs. In *Proceedings of the 10th theory of cryptography conference on Theory of Cryptography*, pages 315–333, 2013.

[BCPS18] Anurag Bishnoi, Pete L Clark, Aditya Potukuchi, and John R Schmitt. On zeros of a polynomial in a finite grid. *Combinatorics, Probability and Computing*, 27(3):310–333, 2018.

[BEP+21] Pauline Bert, Gautier Eberhart, Lucas Prabel, Adeline Roux-Langlois, and Mohamed Sabt. Implementation of lattice trapdoors on modules and applications. In *Post-Quantum Cryptography: 12th International Workshop, PQCrypto 2021, Daejeon, South Korea, July 20–22, 2021, Proceedings 12*, pages 195–214. Springer, 2021.

[BISW17] Dan Boneh, Yuval Ishai, Amit Sahai, and David J Wu. Lattice-based SNARGs and their application to more efficient obfuscation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 247–277. Springer, 2017.

[BL17] Carsten Baum and Vadim Lyubashevsky. Simple amortized proofs of shortness for linear relations over polynomial rings. *Cryptology ePrint Archive*, 2017.

[BS23] Ward Beullens and Gregor Seiler. LaBRADOR: compact proofs for R1CS from module-SIS. In *Annual International Cryptology Conference*, pages 518–548. Springer, 2023.

[BTT22] Cecilia Boschini, Akira Takahashi, and Mehdi Tibouchi. MuSig-L: Lattice-based multi-signature with single-round online phase. In *Annual International Cryptology Conference*, pages 276–305. Springer, 2022.

[CHM+20] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In *Advances in Cryptology–EUROCRYPT 2020: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10–14, 2020, Proceedings, Part I 39*, pages 738–768. Springer, 2020.

[CKKK23] Heewon Chung, Dongwoo Kim, Jeong Han Kim, and Jiseung Kim. Amortized efficient zk-SNARK from linear-only RLWE encodings. *Journal of Communications and Networks*, 25(3):271–284, 2023.

[DAFS24] Thomas Debris-Alazard, Pouria Fallahpour, and Damien Stehlé. Quantum Oblivious LWE Sampling and Insecurity of Standard Model Lattice-Based SNARKs. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, pages 423–434, 2024.

[DMS24] Michel Dellepere, Pratyush Mishra, and Alireza Shirzad. Garuda and Pari: Faster and Smaller SNARKs via Equifficient Polynomial Commitments. *Cryptology ePrint Archive*, 2024.

[FMN24] Giacomo Fenzi, Hossein Moghaddas, and Ngoc Khanh Nguyen. Lattice-based polynomial commitments: Towards asymptotic and concrete efficiency. *Journal of Cryptology*, 37(3):31, 2024.

[GMNO18] Rosario Gennaro, Michele Minelli, Anca Nitulescu, and Michele Orrù. Lattice-based zk-SNARKs from square span programs. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pages 556–573, 2018.

[Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. In *Advances in Cryptology–EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II 35*, pages 305–326. Springer, 2016.

[HSS24] Intak Hwang, Jinyeong Seo, and Yongsoo Song. Concretely efficient lattice-based polynomial commitment from standard assumptions. In *Annual International Cryptology Conference*, pages 414–448. Springer, 2024.

[IKO07] Yuval Ishai, Eyal Kushilevitz, and Rafail Ostrovsky. Efficient arguments without short pcps. In *Twenty-Second Annual IEEE Conference on Computational Complexity (CCC'07)*, pages 278–291. IEEE, 2007.

[ISW21] Yuval Ishai, Hang Su, and David J Wu. Shorter and faster post-quantum designated-verifier zkSNARKs from lattices. In *Proceedings of the 2021 ACM SIGSAC conference on computer and communications security*, pages 212–234, 2021.

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments. In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 723–732, 1992.

[Lip24] Helger Lipmaa. Polymath: Groth16 is not the limit. In *Annual International Cryptology Conference*, pages 170–206. Springer, 2024.

[LNS21] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. Shorter lattice-based zero-knowledge proofs via one-time commitments. In *IACR International Conference on Public-Key Cryptography*, pages 215–241. Springer, 2021.

[LPR13] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-LWE cryptography. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 35–54. Springer, 2013.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 700–718. Springer, 2012.

[Nit19] Anca Nitulescu. Lattice-based zero-knowledge SNARGs for arithmetic circuits. In *Progress in Cryptology–LATINCRYPT 2019: 6th International Conference on Cryptology and Information Security in Latin America, Santiago de Chile, Chile, October 2–4, 2019, Proceedings 6*, pages 217–236. Springer, 2019.

[NS24] Ngoc Khanh Nguyen and Gregor Seiler. GreyHound: fast polynomial commitments from lattices. In *Annual International Cryptology Conference*, pages 243–275. Springer, 2024.

[NYI+20] Ken Naganuma, Masayuki Yoshino, Atsuo Inoue, Yukinori Matsuoka, Mineaki Okazaki, and Noboru Kunihiro. Post-Quantum zk-SNARK for arithmetic circuits using QAPs. In *2020 15th Asia Joint Conference on Information Security (AsiaJCIS)*, pages 32–39. IEEE, 2020.

[Set20] Srinath Setty. Spartan: Efficient and general-purpose zkSNARKs without trusted setup. In *Annual International Cryptology Conference*, pages 704–737. Springer, 2020.

[SSE+24] Ron Steinfeld, Amin Sakzad, Muhammed F Esgin, Veronika Kuchta, Mert Yassi, and Raymond K Zhao. LUNA: Quasi-Optimally succinct Designated-Verifier Zero-Knowledge Arguments from Lattices. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 3167–3181, 2024.

[WW23] Hoeteck Wee and David J Wu. Lattice-based functional commitments: Fast verification and cryptanalysis. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 201–235. Springer, 2023.