Lattice-Based zk-SNARKs with Hybrid Verification Technique

Supriya Adhikary, Puja Mondal, and Angshuman Karmakar

Indian Institute of Technology Kanpur, India
{adhikarys,pujamondal,angshuman}@cse.iitk.ac.in

Abstract. Zero-knowledge succinct arguments of knowledge (zkSNARK) provide short privacy-preserving proofs for general NP problems. Public verifiability of zkSNARK protocols is a desirable property, where the proof can be verified by anyone once generated. Designated-verifier zero-knowledge is useful when it is necessary that only one or a few individuals should have access to the verification result. All zkSNARK schemes are either fully publicly verifiable or can be verified by a designated verifier with a secret verification key.
In this work, we propose a new notion of a hybrid verification mechanism. Here, the prover generates a proof that can be verified by a designated verifier. For this proof, the designated verifier can generate *auxiliary* information with its secret key. The combination of this proof and the auxiliary information allows any public verifier to verify the proof without any other information. We also introduce necessary security notions and mechanisms to identify a cheating designated verifier or the prover. Our hybrid verification zkSNARK construction is based on module lattices and adapts the zkSNARK construction by Ishai et al. (CCS 2021). In this construction, the designated verifier is required only once after proof generation to create the publicly verifiable proof. Our construction achieves a small constant-size proof and fast verification time, which is linear in the statement size.

1 Introduction

A zero-knowledge proof (ZKP) [21] enables a prover to convince a verifier that a statement belongs to an NP relation without revealing anything else about the statement. Depending on the verification method, the zero-knowledge protocols can be classified into two categories: (i) publicly verifiable protocols and (ii) protocols that can be verified by a designated verifier. For a zero-knowledge succinct argument of knowledge (zkSNARK) [32,25,20], one additionally requires the proof size to be small and a fast verification. While using public verifier zkSNARKs, a prover can provide a proof that can be verified by anyone without any additional information, whereas the designated verifier zkSNARK proofs cannot be verified without a secret verification key that is only available to the designated verifier. Therefore, verification of the designated verifier zkSNARK is restricted to one verifier.

There are many existing zkSNARK constructions that rely on group-based and pairing-based assumptions [23,16,31,33]. These constructions are presumed to be insecure against quantum adversaries. There are several quantum-safe solutions that

are based on hash functions, such as Ligero [3], Aurora [11], Breakdown [22]. In these publicly verifiable schemes, the proof size increases sub-linearly with the witness size. There are also publicly verifiable lattice-based proof systems that generate sub-linear size proofs [12,7,13,1]. For large circuits, these publicly verifiable protocols result in long proving times.

The designated-verifier zero-knowledge proof systems [19,24] offer efficient proving, verification as well as compact proof size even with large circuits. But as we mentioned, the disadvantage in this setting is that the verifier must possess a secret state that is used to verify the proof.

To circumvent this issue Baum et al. [9] formalised the notion of distributed verifier zero-knowledge, where the secret state of the verifier is shared among multiple verifiers using a secret-sharing scheme and the verification is considered valid when t out of n verifiers agree on the validity of the statement.

One of the goals of this work is to bridge the gap between the designated-verifier and the public-verifier, but in a different direction than Baum et al.'s [9] work. We extend the lattice-based designated-verifier zkSNARK framework proposed by Ishai et al. [24] and introduce a hybrid verification technique. Unlike public-verifiable protocols, here the public-verifier cannot directly verify the prover's original proof. Instead, the designated verifier, which can verify the proof, can also generate auxiliary information that can be shared with the public verifier. Any public verifier in possession of this auxiliary information can verify the proof generated by the prover. We must also ensure that this auxiliary information does not leak any information about the secret verification key.

	PQ	**TP**	**PV**	**Proof size**		**Runtime**		**Problem**
				Asymptotic[§]	**Concrete**	**Prover**	**Verifier**	
Groth [23]	○	○	●	1	128 B	$N\log N$	$\|x\|$	Pairing
Marlin [16]	○	●	●	1	704 B	$N\log N$	$\|x\|+\log N$	Pairing
Sonic [31]	○	◐	●	1	1.1 KB	$N\log N$	$\|x\|+\log N$	Pairing
Spartan [33]	○	●	●	$\sqrt{N}$	142 KB	N	$\|x\|+\sqrt{N}$	Groups
Fractal [17]	●	●	●	$\log^2 N$	215 KB	$N\log N$	$\|x\|+\log^2 N$	RO
Labrador [12]	●	●	●	$\log N$	58 KB	$N\log^2 N$	$\|x\|+N\log^2 N$	Lattice
STARK [10]	●	●	●	$\log^2 N$	127 KB	$N\text{polylog}(N)$	$\|x\|+\log^2 N$	RO
ISW [24]	●	○	○	1	16 KB	$N\log N$	$\|x\|$	Lattice
This Work	●	○	◐	1	54.7 KB* 2.1 KB†	$N\log N$	$\|x\|$‡	Lattice

§ All the asymptotic values are given in big O notation ($\mathcal{O}$).
* In our work, the prover generates a proof of size ≈54.7 KB.
† The designated verifier generates the **aux** of size ≈2.1 KB.
‡ The designated verifier's verification time is $\mathcal{O}(|x|)$ but the public verifier verifies in time $\mathcal{O}(1)$. Therefore, the overall time is in $\mathcal{O}(|x|)$.

Table 1: Comparision of asymptotic proof size and runtime for different zkSNARK techniques

In Table 1, we have compared the asymptotic proof size and runtime of recent zkSNARK techniques with our work. The column **PQ** specifies if the scheme is post-quantum or not. Similarly, the columns **TP** and **PV** specify the properties transparent setup and public verifiability, respectively. The white circle ("○") means that the scheme does not satisfy a certain property, the black circle ("●") specifies that the scheme satisfies a property. The shaded circle ("◐") in the **TP** column means that the scheme relies on a trusted setup for a universal CRS. In our work, the shaded circle ("◐") in the **PV** column means that we have the hybrid verification technique. Here N is the constraint size, and the proof sizes are given for an NP relation with $N = 2^{20}$ constraints.

Our work: Primarily, we first introduce the notion of hybrid verification zkSNARK (HV-zkSNARK) in this work. We have also presented the formal definition of HV-zkSNARK and the associated security requirements.

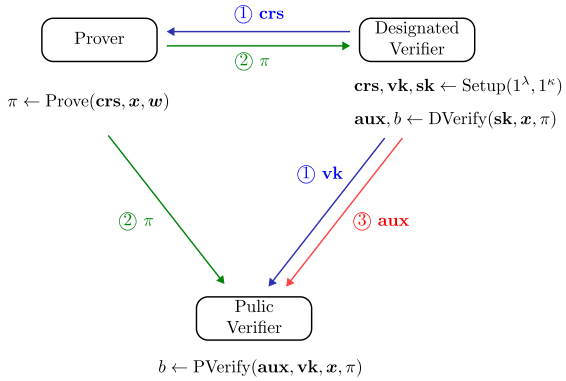


Fig. 1: Hybrid verification framework of the proposed HV-zkSNARK

We provide a brief description of our protocol below. A pictorial description has been shown in Fig. 1.

1. The designated-verifier generates the common reference string **crs** and the verification key **vk** using the secret key **sk**. The (**crs**,**vk**) is made public before the proof generation starts.
2. The prover has a witness $\boldsymbol{w}$ corresponding to a statement $\boldsymbol{x}$. Prover uses the **crs**, $\boldsymbol{x}$ and $\boldsymbol{w}$ to compute the proof π. The proof π is made public. Even though π is made public, this proof can only be verified by the designated verifier.
3. The designated-verifier checks the proof π using the secret verification key **sk** and if it is valid then it generates **aux** and makes it public.
4. A public-verifier can now verify the validity of the proof π using the auxiliary information **aux** and verification key **vk**.

Two adversarial scenarios must be considered. First, the prover can generate an invalid proof. In such a case, even if the designated verifier is honest and generates

the auxiliary information honestly, the verification will fail at the public verifier's end. Second, the public verifier is honest, but the designated verifier generates an invalid **aux**. In this case, the verification will again fail at the public verifier's end. Therefore, a properly functioning HV-zkSNARK protocol must provide a mechanism that allows detection of such cases and identifies the party that has generated the invalid proof or the invalid auxiliary information.

Use case: A key use case of hybrid verifier zkSNARKs emerges in the upcoming Web 4.0 [34,6] or the intelligent web, where the Artificial Intelligence (AI) agents will act as persistent digital counterparts of users, handling tasks such as booking vacations, scheduling meetings, collecting data on behalf of users, etc. These user-agents negotiate with the merchant-agents and adapt plans in real time. Since these agents require access to user-specific data across different merchants, these agents must possess user credentials. Consequently, ensuring the security and privacy of these credentials is critical, especially given the susceptibility of user-agents to malicious interactions.

Zero-knowledge proofs of knowledge can be very useful in this scenario. Here, the service provider is the verifier (can be either public or designated), and the prover is the user. The user can generate a proof of valid credentials for a service provider and delegates it to its agent, which in turn authenticates on the user's behalf. However, if we consider either the traditional public- or the designated-verifier model, there is a security flaw in this arrangement. Consider a malicious agent impersonating a merchant-agent, who has received the proof of valid credentials from the user-agent or the user. Now, this malicious agent can use the proof of credentials it received earlier to impersonate the user-agent to another service provider. The service provider cannot detect that the proof of credentials came from a malicious agent instead of the actual user-agent.

Our hybrid verification method can be used to tackle this problem. First, the user agent would generate the public information (**crs**, **vk**) based on its secret key (say, **sk**) and make it available to the user (prover) and the service provider (public verifier) as in Fig. 1. The user generates now proof (say $\boldsymbol{\pi}$) and makes it available to both the service provider (public verifier) and the user-agent (designated verifier). The user-agent will generate the auxiliary information (say **aux**) and send it to the service provider, so that it can verify whether the proof is valid or not. This verification is possible with the help of **aux** and **vk** that were already provided by the user-agent earlier. Here, the agent proves that it has the secret key **sk** and that the verification of $\boldsymbol{\pi}$ is valid. Therefore, if any other (malicious) agent tries to authenticate itself using the same **aux**, then the service provider can detect that, as it cannot prove that it has the secret key **sk**.

Our approach: In the designated-verifier zero-knowledge system proposed by Ishai et al. [24], the proof generated by the prover is the LWE encryption of a vector $\boldsymbol{\pi}$. The verifier (designated) decrypts the proof and gets $\boldsymbol{\pi}$ and adds a vector **st** (known only to the designated verifier) to compute $\boldsymbol{r} = \boldsymbol{\pi} + \mathbf{st}$. Now, the verifier only checks if $r_1 r_2 - r_3 - r_4 r_5 = 0$ to check the validity of the proof. Here the decryption key $\boldsymbol{S} \in \mathcal{R}_q^{n \times 5}$ and the vector **st** together work as a secret verification key. In Section 2.2, we have briefly recalled the designated-verifier zkSNARK proposed by Ishai et al. [24].

Our target is to make sure that any public entity other than the designated verifier should also be able to verify the validity of the proof. Now, the designated verifier can publish the encryption of $\mathbf{st}$. Since the encryption used by Ishai et al. [24] is a linear encryption, one can easily compute the encryption of $\boldsymbol{r}=\boldsymbol{\pi}+\mathbf{st}$ by adding the encryptions of $\boldsymbol{\pi}$ and $\mathbf{st}$. It then remains to be proven that this $\boldsymbol{r}$ satisfies the quadratic equation $r_1r_2-r_3-r_4r_5=0$.

Attema et al. [5] proposed a protocol for proving multiplicative relations between committed polynomials in $\mathcal{R}_q$. The protocol assumes the elements have been committed with the commitment scheme proposed by Baum et al. [8]. Lyubashevsky et al. [27] used a similar approach to prove arbitrary quadratic relations over $\mathcal{R}_q$, but they introduced a different commitment scheme in their work. One can show that a similar technique can be used to give proof of linear and quadratic relations when the polynomials are committed using the commitment scheme proposed by Lyubashevsky et al. [28]. First, we recall the commitment scheme in [28]. Given a message $\boldsymbol{m}=(m_1,\cdots,m_\ell)\in\mathcal{R}_q^\ell$, prover samples a vector $\boldsymbol{s}\leftarrow\mathcal{R}_q^n$ with $\|\boldsymbol{s}\|$ small, $\boldsymbol{B}\xleftarrow{\$}\mathcal{R}_q^{k\times n}$, $\boldsymbol{a}_i\xleftarrow{\$}\mathcal{R}_q^n$ for $1\leq i\leq\ell$ and commits to $\boldsymbol{m}$ as

$$\begin{aligned}\boldsymbol{t}_s&=\boldsymbol{B}\boldsymbol{s}\\ \boldsymbol{t}_m^{(i)}&=\boldsymbol{a}_i^\top\boldsymbol{s}+m_i\quad\forall i\in[\ell]\end{aligned}$$

Using the technique proposed by Lyubashevsky et al. [27], the prover can prove that the vector $\boldsymbol{m}$ follows some quadratic relation without revealing $\boldsymbol{m}$ or $\boldsymbol{s}$. This approach achieves the desired outcome of verifying the relation without revealing the secret vector.

As we mentioned before, the encryption of $\boldsymbol{r}=\boldsymbol{\pi}+\mathbf{st}$ can be computed by any public entity if both encryptions of $\boldsymbol{\pi}$ and $\mathbf{st}$ are available. The LWE encryption of $\boldsymbol{r}=(r_1,r_2,r_3,r_4,r_5)\in\mathcal{R}_p^5$ would be of the form

$$\boldsymbol{t}_r^{(i)}=\boldsymbol{a}_i^\top\boldsymbol{s}_i+pe_i+r_i\quad\text{for } 1\leq i\leq 5$$

where $s_i\leftarrow\chi^n$, $\boldsymbol{a}_i\xleftarrow{\$}\mathcal{R}_q^n$, $e_i\leftarrow\chi$ and p is an integer less than q. Here χ is a distribution over $\mathcal{R}_q$. Now, the designated-verifier can sample $\boldsymbol{B}_i\xleftarrow{\$}\mathcal{R}_q^{k\times n}$ and compute $\boldsymbol{t}_s^{(i)}=\boldsymbol{B}_i\boldsymbol{s}_i$ for $1\leq i\leq 5$ and make these public. Now, this $\{\boldsymbol{t}_s^{(i)},\boldsymbol{t}_r^{(i)}\}_{1\leq i\leq 5}$ can be considered as a commitment of the vector $\boldsymbol{r}$. Following the footsteps of the work proposed by Lyubashevsky et al. [27], we have proposed a zero-knowledge protocol $\Pi_{\text{Quad}}^{(1)}$ in Section 3.2 that can prove that $\boldsymbol{r}$ satisfies a quadratic relation in $\mathcal{R}_p$ for any arbitrary quadratic relation. This implies that if $\boldsymbol{r}$ is honestly generated, then the designated verifier can provide a proof to convince a public verifier that it is a valid proof without revealing the secret verification key *i.e.* $(\{\boldsymbol{s}_i\}_{1\leq i\leq 5},\mathbf{st})$.

For detection of malicious party in the HV-zkSNARK, we modify the protocol $\Pi_{\text{Quad}}^{(1)}$ and propose the protocol $\Pi_{\text{Quad}}^{(2)}$ in Section 4.1, where we have discussed the technique to detect whether the prover or the designated-verifier has provided incorrect information. Moreover, using the protocol $\Pi_{\text{Quad}}^{(1)}$, the designated verifier can generate proof that the common reference string (CRS) is honestly computed, which is discussed in Section 4.3.

2 Background

We consider N to be a number that is a power of 2. We denote the ring $\mathbb{Z}[X]/\langle X^N+1\rangle$ as $\mathcal{R}$ and we consider the ring $\mathbb{Z}_Q[X]/\langle X^N+1\rangle$ as $\mathcal{R}_Q$, where Q is an integer. We will denote an element in $\mathcal{R}_Q$ (or $\mathcal{R}$) with a lowercase letter. Any vector over $\mathcal{R}_Q$ (or $\mathcal{R}$) is denoted by bold lowercase letters, and a matrix over $\mathcal{R}_Q$ (or $\mathcal{R}$) is denoted by bold uppercase letters. If $m(X)=\sum_{i=0}^{N-1} m_i X^i \in \mathcal{R}_Q$ (or $\mathcal{R}$) then we can write it as the vector $(m_0,m_1,\cdots,m_{N-1})$, which is the vector of all coefficients of $m(X)$. We also denote the scalars in $\mathbb{Z}_Q$ (or $\mathbb{Z}$) as lowercase letters, as they can be considered as elements in $\mathcal{R}_Q$ (or $\mathcal{R}$). We consider prime integer p such that $p \equiv 2d+1 \pmod{4d}$ and we consider $q=q_1 \cdot q_2 \cdots q_t$ where q_i's are all primes satisfying $q_i \equiv 2d+1 \pmod{4d}$ for $1 \leq i \leq t$ for some $1<d\leq N$. The notation $[n]$ represents the set $\{1,2,\cdots,n\}$.

2.1 Definitions

Definition 1 (Rank 1 Circuit Satisfiability [24,11]). *A R1CS system over a finite field $\mathbb{F}$ is specified by a tuple $\mathcal{CS}=(n,N_g,N_\omega,\{\boldsymbol{a}_i,\boldsymbol{b}_i,\boldsymbol{c}_i\}_{i\in[N_g]})$, where $n,N_g,N_\omega \in \mathbb{N}$, $n \leq N_\omega$, and $\boldsymbol{a}_i,\boldsymbol{b}_i,\boldsymbol{c}_i \in \mathbb{F}^{N_\omega+1}$. The system $\mathcal{CS}$ is satisfiable for a statement $\boldsymbol{x}\in\mathbb{F}^n$ if there exists a witness $\boldsymbol{\omega}\in\mathbb{F}^{N_\omega}$ such that*

- $\boldsymbol{x}=(\omega_1,\omega_2,\cdots,\omega_n)^\top$
- $\left[1|\boldsymbol{\omega}^\top\right]\boldsymbol{a}_i \cdot \left[1|\boldsymbol{\omega}^\top\right]\boldsymbol{b}_i = \left[1|\boldsymbol{\omega}^\top\right]\boldsymbol{c}_i \ \forall i \in [N_g]$

We denote this by writing $\mathcal{CS}(\boldsymbol{x},\boldsymbol{\omega})=1$, and refer to n as the statement size, N_ω as the number of variables, and N_g as the number of constraints. Given an R1CS system $\mathcal{CS}$, we define the corresponding relation $\mathcal{R}_{\mathcal{CS}}=\{(\boldsymbol{x},\boldsymbol{\omega})\in\mathbb{F}^n\times\mathbb{F}^{N_\omega} : \mathcal{CS}(\boldsymbol{x},\boldsymbol{\omega})=1\}$

Definition 2 (Linear PCP [24]). *Let $\mathcal{CS}=\{\mathcal{CS}_\kappa\}_{\kappa\in\mathbb{N}}$ be a family of R1CS instances over a finite field $\mathbb{F}$, where $\mathcal{CS}_\kappa=\{n_\kappa,N_{g,\kappa},N_{\omega,\kappa},\{\boldsymbol{a}_{i,\kappa},\boldsymbol{b}_{i,\kappa},\boldsymbol{c}_{i,\kappa}\}\}_{i\in[N_{g,\kappa}]}$. For notational convenience, we write $n=n(\kappa)$ to denote a function where $n_c(\kappa)=n_{c,\kappa}$ for all $\kappa\in\mathbb{N}$. We define $N_g=N_g(\kappa)$, $N_\omega=N_\omega(\kappa)$ similarly. A k-query input-independent linear PCP for $\mathcal{CS}$ with query length $\ell=\ell(\kappa)$ and knowledge error $\varepsilon=\varepsilon(\kappa)$ is a tuple of algorithms $\Pi_{\mathrm{LPCP}}=(\mathcal{Q}_{\mathrm{LPCP}},\mathcal{P}_{\mathrm{LPCP}},\mathcal{V}_{\mathrm{LPCP}})$ with the following properties:*

- $\mathbf{st},\boldsymbol{Q}\leftarrow\mathcal{Q}_{\mathrm{LPCP}}(1^\kappa)$*: The query-generation algorithm takes as input the system index $\kappa\in\mathbb{N}$ and outputs a query matrix $\boldsymbol{Q}\in\mathbb{F}^{\ell\times k}$ and a verification state $\mathbf{st}$.*
- $\boldsymbol{\pi}\leftarrow\mathcal{P}_{\mathrm{LPCP}}(1^\kappa,\boldsymbol{x},\boldsymbol{\omega})$*: On input the system index $\kappa\in\mathbb{N}$, a statement $\boldsymbol{x}\in\mathbb{F}^n$, and a witness $\boldsymbol{\omega}\in\mathbb{F}^{N_\omega}$, the prove algorithm generates a $\boldsymbol{\sigma}\in\mathbb{F}^\ell$ outputs a proof $\boldsymbol{\pi}=\boldsymbol{Q}^\top\boldsymbol{\sigma}\in\mathbb{F}^k$*
- $\mathcal{V}_{\mathrm{LPCP}}(\mathbf{st},\boldsymbol{x},\boldsymbol{\pi})$*: On input the verification state $\mathbf{st}$, the statement $\boldsymbol{x}\in\mathbb{F}^n$, and a vector of responses $\boldsymbol{\pi}\in\mathbb{F}^k$, the verification algorithm outputs a bit $b\in\{0,1\}$.*

The properties of a linear PCP are discussed in detail in [24].

Definition 3 (Quadratic Arithmetic Program [18]). *A quadratic arithmetic program (QAP) $\mathcal{Q}$ over field $\mathbb{F}$ contains three sets of polynomials $\{A_k(x) : k \in \{0,\cdots,N_\omega\}\}$, $\{B_k(x):k\in\{0,\cdots,N_\omega\}\}$, $\{C_k(x):k\in\{0,\cdots,N_\omega\}\}$, and a target polynomial*

$t(x)$, *all from* $\mathbb{F}[x]$. *Let* f *be a function having input variables with labels* $1,\cdots,n$ *and output variables with labels* $N_\omega - n' + 1,\cdots,N_\omega$. *We say that* $\mathcal{Q}$ *is a QAP that computes* f *if the following is true:* $a_1,\cdots,a_n,a_{N_\omega-n'+1},\cdots,a_{N_\omega} \in \mathbb{F}^{n+n'}$ *is a valid assignment to the input/output variables of* f *iff there exist* $(a_{n+1},\cdots,a_{N_\omega-n'}) \in \mathbb{F}^{N_\omega-n-n'}$ *such that* $t(x)$ *divides* $A(x)\cdot B(x) - C(x)$ *where*

$$A(x) = \left(A_0(x) + \sum_{k\in[N_\omega]} a_k A_k(x)\right), \quad B(x) = \left(B_0(x) + \sum_{k\in[N_\omega]} a_k B_k(x)\right)$$

$$C(x) = \left(C_0(x) + \sum_{k\in[N_\omega]} a_k C_k(x)\right)$$

The size of $\mathcal{Q}$ *is* m *and the degree of* $\mathcal{Q}$ *is* $\deg(t(x))$.

One can construct a Linear PCP (LPCP) for an R1CS system using the QAP construction given by [18].

Now, we recall the definitions of a succinct non-interactive argument of knowledge (SNARK) for R1CS:

Definition 4 (Succinct Non-Interactive Argument of Knowledge [24]). *Let* $\mathcal{CS} = \{\mathcal{CS}_\kappa\}_{\kappa\in\mathbb{N}}$ *be a family of R1CS systems over a finite field* $\mathbb{F}$, *where* $|\mathcal{CS}_\kappa| \geq s(\kappa)$ *for some fixed polynomial* $s(\cdot)$. *A succinct non-interactive argument (SNARK) in the preprocessing model for* $\mathcal{CS}$ *is a tuple* $\Pi_{\mathrm{SNARK}} = (\mathrm{Setup},\mathrm{Prove},\mathrm{Verify})$ *with the following properties:*

- $(\mathbf{crs},\mathbf{st}) \leftarrow \mathrm{Setup}(1^\lambda,1^\kappa)$*: On input the security parameter* λ *and the system index* κ, *the setup algorithm outputs a common reference string* **crs** *and verification state* **st**.
- $\pi \leftarrow \mathrm{Prove}(\mathbf{crs},\boldsymbol{x},\boldsymbol{\omega})$*: On input a common reference string* **crs**, *a statement* $\boldsymbol{x}$, *and a witness* $\boldsymbol{\omega}$, *the prove algorithm outputs a proof* π.
- $b \leftarrow \mathrm{Verify}(\mathbf{st},\boldsymbol{x},\pi)$*: : On input the verification state* **st**, *a statement* $\boldsymbol{x}$ *and a proof* π, *the verification algorithm outputs a bit* $b \in \{0,1\}$ *that specifies whether the verification accepts or rejects.*

Moreover, Π_{SNARK} should satisfy the following properties:

- **Completeness**: For all security parameters $\lambda \in \mathbb{N}$, system indices $\kappa \in \mathbb{N}$, and instances $(\boldsymbol{x},\boldsymbol{\omega})$ where $\mathcal{CS}_\kappa(\boldsymbol{x},\boldsymbol{\omega}) = 1$.

  $$\Pr[\mathrm{Verify}(\mathbf{st},\boldsymbol{x},\pi) = 1] = 1,$$

  where $(\mathbf{crs},\mathbf{st}) \leftarrow \mathrm{Setup}(1^\lambda,1^\kappa), \pi \leftarrow \mathrm{Prove}(\mathbf{crs},\boldsymbol{x},\boldsymbol{\omega})$.
- **Knowledge**: For all polynomial-size provers $\mathcal{P}^*$, there exists a polynomial-size extractor $\mathcal{E}$ such that for all security parameters $\lambda \in \mathbb{N}$, system indices $\kappa \in \mathcal{N}$, and auxiliary inputs $z \in \{0,1\}^{\mathrm{poly}(\lambda)}$,

  $$\Pr[\mathrm{Verify}(\mathbf{st},\boldsymbol{x},\pi) = 1 \wedge \mathcal{CS}_\kappa(\boldsymbol{x},\boldsymbol{\omega}) \neq 1] = \mathrm{negl}(\lambda)$$

  where $(\mathbf{crs},\mathbf{st}) \leftarrow \mathrm{Setup}(1^\lambda,1^\kappa), (\boldsymbol{x},\pi) \leftarrow \mathcal{P}^*(1^\lambda,1^\kappa,\mathbf{crs};z)$, and the witness extracted as $\boldsymbol{\omega} \leftarrow \mathcal{E}(1^\lambda,1^\kappa,\mathbf{crs},\mathbf{st},\boldsymbol{x};z)$

- **Efficiency**: There exist a universal polynomial poly (independent of $\mathcal{CS}$) such that Setup and Prove run in time $\text{poly}(\lambda+|\mathcal{CS}_\kappa|)$, Verify runs in time $\text{poly}(\lambda+|\boldsymbol{x}|+\log|\mathcal{CS}_\kappa|)$, and the proof size is $\text{poly}(\lambda+\log|\mathcal{CS}_\kappa|)$.
- **Zero-Knowledge**: $\Pi_{\text{SNARK}}=(\text{Setup},\text{Prove},\text{Verify})$ is zero knowledge (*i.e.* zk-SNARK) if there exists an efficient simulator $\mathcal{S}_{\text{SNARK}}=(\mathcal{S}_1,\mathcal{S}_2)$ such that for all $\kappa\in\mathbb{N}$ and all efficient adversaries $\mathcal{A}$, we have that

$$\Pr[\text{ExptZK}_{\Pi_{\text{SNARK}},\mathcal{A},\mathcal{S}_{\text{SNARK}}}(1^\lambda,1^\kappa)=1]\leq 1/2+\text{negl}(\lambda) \tag{1}$$

  where the experiment $\text{ExptZK}_{\Pi_{\text{SNARK}},\mathcal{A},\mathcal{S}_{\text{SNARK}}}(1^\lambda,1^\kappa)$ is defined as follows:

  1. The challenger samples $b\xleftarrow{\$}\{0,1\}$.
     - If $b=0$, the challenger computes $(\mathbf{crs},\mathbf{st})\leftarrow\text{Setup}(1^\lambda,1^\kappa)$ and gives $(\mathbf{crs},\mathbf{st})$ to $\mathcal{A}$.
     - If $b=1$, the challenger computes $(\widetilde{\mathbf{crs}},\widetilde{\mathbf{st}},\mathbf{st}_{\mathcal{S}})\leftarrow\mathcal{S}_1(1^\lambda,1^\kappa)$ and gives $(\widetilde{\mathbf{crs}},\widetilde{\mathbf{st}})$ to $\mathcal{A}$.
  2. The adversary $\mathcal{A}$ outputs a statement $\boldsymbol{x}$ and a witness $\boldsymbol{\omega}$.
  3. If $\mathcal{CS}(\boldsymbol{x},\boldsymbol{\omega})\neq 1$, then the experiment halts and output 0. Otherwise, the challenger proceeds as follows:
     - If $b=0$, the challenger replies with $\pi\leftarrow\text{Prove}(\mathbf{crs},\boldsymbol{x},\boldsymbol{\omega})$.
     - If $b=1$, the challenger replies with $\widetilde{\pi}\leftarrow\mathcal{S}_2(\mathbf{st}_{\mathcal{S}},\boldsymbol{x})$.

     At the end of the experiment, $\mathcal{A}$ outputs a bit $b'\in\{0,1\}$. The output of the experiment is 1 if $b'=b$ and is 0 otherwise.

Definition 5 (Module Learning With Errors (MLWE) [14]). *Fix a security parameter λ and select $n=n(\lambda),k=k(\lambda)$ and $q=q(\lambda)$. Let χ_s,χ_e be secret and error distribution over $\mathcal{R}_q$, respectively. The (decisional) module learning with errors (MLWE) assumption $\texttt{MLWE}_{n,k,q,\chi_s,\chi_e}$ states that for $\boldsymbol{A}\xleftarrow{\$}\mathcal{R}_q^{k\times n},\boldsymbol{s}\leftarrow\chi_s^n,\boldsymbol{e}\leftarrow\chi_e^k$, and $\boldsymbol{u}\xleftarrow{\$}\mathcal{R}_q^k$, the following two distributions are computationally indistinguishable:*

$$(\boldsymbol{A},\boldsymbol{s}^\top\boldsymbol{A}+\boldsymbol{e}^\top)\quad\text{and}\quad(\boldsymbol{A},\boldsymbol{u}^\top)$$

Definition 6 (Extended MLWE [29]). *Fix a security parameter λ and select $n=n(\lambda),k=k(\lambda)$ and $q=q(\lambda)$. Let χ be a probability distribution over $\mathcal{R}_q$, $\mathcal{C}\subseteq\mathcal{R}_q$ be a challenge space and $\mathfrak{s}$ be a standard deviation. The Extended MLWE assumption* $\text{Extended-}\texttt{MLWE}_{n,k,\chi,\mathfrak{s}}$ *asks the adversary $\mathcal{A}$ to distinguish between the following two cases:*

1. $(\boldsymbol{B},\boldsymbol{Br},c,\boldsymbol{z},\text{sign}(\langle\boldsymbol{z},c\boldsymbol{r}\rangle))$ *for $\boldsymbol{B}\xleftarrow{\$}\mathcal{R}_q^{k\times n}$, a secret $\boldsymbol{r}\leftarrow\chi^n$ and $\boldsymbol{z}\leftarrow\mathcal{D}_{\mathfrak{s}}^n$, $c\leftarrow\mathcal{C}$.*
2. $(\boldsymbol{B},\boldsymbol{u},c,\boldsymbol{z},\text{sign}(\langle\boldsymbol{z},c\boldsymbol{r}\rangle))$ *for $\boldsymbol{B}\xleftarrow{\$}\mathcal{R}_q^{k\times n}$, a secret $\boldsymbol{u}\xleftarrow{\$}\mathcal{R}_q^k$ and $\boldsymbol{z}\leftarrow\mathcal{D}_{\mathfrak{s}}^n$, $c\leftarrow\mathcal{C}$.*

where $\text{sign}(a)=1$ if $a\geq 0$ and 0 otherwise.

Definition 7 (Module Short Integer Solution (MSIS)). *Fix a security parameter λ and select $n=n(\lambda)$, $k=k(\lambda)$, $q=q(\lambda)$ and $B=B(\lambda)$ where $m,n>0$ and $0<B<q$. The MSIS problem states that for $\boldsymbol{A}\xleftarrow{\$}\mathcal{R}_q^{k\times n}$, find a $\boldsymbol{z}\in\mathcal{R}_q^n$ such that $\boldsymbol{Az}=\boldsymbol{0}\pmod q$ and $0<\|\boldsymbol{z}\|_\infty\leq B$. An algorithm $\mathcal{A}$ is said to have advantage ϵ in solving $\texttt{MSIS}_{k,m,B}$ if*

$$\Pr\Big[(0<\|\boldsymbol{z}\|_\infty\leq B)\wedge(\boldsymbol{Az}=\boldsymbol{0})\mid\boldsymbol{A}\xleftarrow{\$}\mathcal{R}_q^{k\times n};\boldsymbol{z}\leftarrow\mathcal{A}(\boldsymbol{A})\Big]\geq\epsilon$$

2.2 Linear Only Encryption Scheme

We first describe the encryption scheme used in [24], which is based on the MLWE [26] assumption. Let N be a power of 2. Fix lattice parameters q,p,n and χ_s,χ_e as secret and error distributions, respectively. We additionally include the following parameters. ℓ as the plaintext dimension, τ as the sparsification parameter and B as the smudging bound, which is explained in the next subsection. Let $\ell' = \ell + \tau$. Now we construct an encryption scheme

$$\Pi_{\text{Enc}} = (\mathbf{Setup}_{\text{Enc}}, \mathbf{Encrypt}_{\text{Enc}}, \mathbf{Add}_{\text{Enc}}, \mathbf{Decrypt}_{\text{Enc}})$$

to encrypt a vector $\boldsymbol{v} \in \mathcal{R}_p^\ell$.

- $\mathbf{Setup}(1^\lambda, 1^\ell)$: Sample matrices $\boldsymbol{A} \xleftarrow{\$} \mathcal{R}_q^{n\times n}$, $\boldsymbol{S} \leftarrow \chi^{n\times \ell'}$, $\boldsymbol{T} \xleftarrow{\$} \mathcal{R}_p^{\tau\times \ell}$ and $\boldsymbol{E} \leftarrow \chi^{n\times \ell'}$. Compute $\boldsymbol{D} \leftarrow \boldsymbol{S}^\top \boldsymbol{A} + p\boldsymbol{E}^\top \in \mathcal{R}_q^{\ell'\times n}$. Output the secret key $\mathbf{sk} = (\boldsymbol{S}, \boldsymbol{T})$ and the public parameters as $\mathbf{pp} = (\boldsymbol{A}, \boldsymbol{D})$.
- $\mathbf{Encrypt}(\mathbf{sk}, \boldsymbol{v})$: On the input of secret key $\mathbf{sk} = (\boldsymbol{S}, \boldsymbol{T})$ and message vector $\boldsymbol{v} \in \mathcal{R}_p^\ell$, we compute the vector $\boldsymbol{u} = \left[\boldsymbol{v}^\top \mid (\boldsymbol{T}\boldsymbol{v})^\top\right]^\top \in \mathcal{R}_p^{\ell'}$. Sample $\boldsymbol{a} \xleftarrow{\$} \mathcal{R}_q^n$, $\boldsymbol{e} \leftarrow \chi^{\ell'}$ and compute $\boldsymbol{c} \leftarrow \boldsymbol{S}^\top \boldsymbol{a} + p\boldsymbol{e} + \boldsymbol{u} \in \mathcal{R}_q^{\ell'}$. Output the ciphertext $\mathbf{ct} = (\boldsymbol{a}, \boldsymbol{c})$.
- $\mathbf{Decrypt}(\mathbf{sk}, \mathbf{ct})$: On the input of secret key $\mathbf{sk} = (\boldsymbol{S}, \boldsymbol{T})$ and the ciphertext $\mathbf{ct} = (\boldsymbol{a}, \boldsymbol{c})$, compute $\boldsymbol{z} \leftarrow \boldsymbol{c} - \boldsymbol{S}^\top \boldsymbol{a} \in \mathcal{R}_q^{\ell'}$. Compute $\boldsymbol{u} = \boldsymbol{z} \pmod p$ and parse $\boldsymbol{u} = \left[\boldsymbol{v}_1^\top \mid \boldsymbol{v}_2^\top\right]^\top$ where $\boldsymbol{v}_1 \in \mathcal{R}_p^\ell$ and $\boldsymbol{v}_2 \in \mathcal{R}_p^\tau$. Output $\boldsymbol{v}_1$ as the decryption if $\boldsymbol{v}_2 = \boldsymbol{T}\boldsymbol{v}_1$ and $\perp$ otherwise.
- $\mathbf{Add}(\mathbf{pp}, \{\mathbf{ct}_i\}_{i\in[r]}, \{\lambda_i\}_{i\in[r]})$: On the input of public parameters $\mathbf{pp} = (\boldsymbol{A}, \boldsymbol{D})$ and the ciphertexts $\mathbf{ct}_i = (\boldsymbol{a}_i, \boldsymbol{c}_i)$ and scalars $\lambda_i \in \mathbb{Z}_p$ for $i \in [r]$, sample $\boldsymbol{r} \leftarrow \chi^n$, $\boldsymbol{e}_a \leftarrow \chi^n$, $\boldsymbol{e}_c \xleftarrow{\$} [-B, B]^{N\ell'}$ and output the ciphertext

$$\mathbf{ct}^* = \left(\sum_{i\in[r]} \lambda_i \boldsymbol{a}_i + \boldsymbol{A}\boldsymbol{r} + p\boldsymbol{e}_a, \sum_{i\in[r]} \lambda_i \boldsymbol{c}_i + \boldsymbol{D}\boldsymbol{r} + p\boldsymbol{e}_c\right) \tag{2}$$

Correctness and Security of Linear Only Encryption: In this section, we provide the main theorems from the work of Ishai et al. [24] on the security and correctness of the construction in Section 2.2. We write γ_R to denote the expansion constant where for all $r, s \in \mathcal{R}$, we have that $\|r \cdot s\|_\infty \leq \gamma_R \|r\|_\infty \cdot \|s\|_\infty$.

For correctnes, we need to make sure that after homomorphically adding the ciphertexts using a linear combination, the resultant ciphertext can be decrypted correctly. The Theorem 1 ensures that we can find correct parameters to ensure that.

Theorem 1 (Additive Homomorphism [24]). *Let λ be a security parameter and p, q, n, ℓ', χ, B be as defined in Section 2.2. Suppose χ is subgaussian with parameter σ. If n, ℓ', σ, N, $\gamma_R = poly(\lambda)$, then for all $r = r(\lambda)$, there exists $q = (pB + rp^2)poly(\lambda)$ such that the construction in Section 2.2 is additively homomorphic with respect to $\mathcal{R}_p^r$. Concretely, let C be a correctness parameter and let B_1, B_2 be bounds. Define the set $S = \{\boldsymbol{y} \in \mathcal{R}_p^r : \|y\|_1 \leq B_1 \text{ and } \|y\|_2 \leq B_2\}$. If $q > 2p(B + \gamma_R B_2 C\sigma + \gamma_R B_1/2 + 2\gamma_R n C^2 \sigma^2) + p$ then the decryption of the ciphertext in Eq. (2) fails with probability $1 - (4n+2)N\ell' \exp(-\pi C^2)$ for all $\boldsymbol{y} \in S$.*

Theorem 2 (CPA security [24]). *Fix a security parameter λ and let p, q, n, ℓ', χ be as defined in Section 2.2. Take any $Q=poly(\lambda)$ and suppose that p, q are coprime. Under the* $\mathrm{MLWE}_{n,m,N,q,\chi}$ *assumption with $m=n+Q$, construction in Section 2.2 is Q-query CPA secure.*

For the prover's security, we additionally need circuit privacy. Circuit privacy says that the ciphertext output by **Add** can be simulated given only the underlying plaintext value, without knowledge of the linear combination used to construct the ciphertext.

Definition 8 (Circuit Privacy [24]). *Let $\Pi_{Enc}=(\boldsymbol{Setup}, \boldsymbol{Encrypt}, \boldsymbol{Decrypt}, \boldsymbol{Add})$ be a secret-key vector encryption scheme over $\mathbb{F}^\ell$. We say that Π_{Enc} satisfies circuit privacy if for all efficient and stateful adversaries $\mathcal{A}$, there exists an efficient simulator $\mathcal{S}$ such that for all security parameters $\lambda\in\mathcal{N}$,*

$$\Pr[\mathrm{ExptCP}_{\Pi_{Enc},\mathcal{A},\mathcal{S}}(1^\lambda)]=\frac{1}{2}+negl(\lambda) \tag{3}$$

The experiment $\mathrm{ExptCP}_{\Pi_{\mathrm{Enc}},\mathcal{A},\mathcal{S}}$ is defined as follows:

1. The challenger samples $(\mathbf{pp},\mathbf{sk})\leftarrow\mathbf{Setup}(1^\lambda)$ and sends it to the adversary $\mathcal{A}$. The aversary replies with vectors $\boldsymbol{v}_1,\boldsymbol{v}_2,\cdots,\boldsymbol{v}_k\in\mathbb{F}^\ell$.
2. The challenger constructs ciphertexts $\mathbf{ct}_i\leftarrow\mathbf{Encrypt}(\mathbf{sk},\boldsymbol{v}_i)$ for $1\le i\le k$ and gives $\{\mathbf{ct}_i\}_{i\in[k]}$ to $\mathcal{A}$. The adversary replies with a collection of coefficients $y_1,\cdots,y_k\in\mathbb{F}$.
3. The challenger computes $\mathbf{ct}_0^*\leftarrow\mathbf{Add}(\mathbf{pp},\{\mathbf{ct}_i\}_{i\in[k]},\{y_i\}_{i\in[k]})$ and $\mathbf{ct}_1^*\leftarrow\mathcal{S}(1^\lambda,\mathbf{pp},\mathbf{sk},\sum_{i\in[k]}y_i\boldsymbol{v}_i)$. It samples a random bit $b\leftarrow\{0,1\}$ and replies to the adversary with $\mathbf{ct}_b^*$.
4. The adversary outputs a bit $b'\in\{0,1\}$. The output of the experiment is 1 if $b'=b$ and 0 otherwise.

Lemma 1 (Smudging Lemma [24]). *Let B, B' be integers. Fix any value $|e_1|\le B'$ and sample $e_2\xleftarrow{\$}[-B,B]$. The statistical distance between the distributions of e_1+e_2 and e_2 is at most B'/B.*

The above Lemma 1, is called the smugding lemma and the bound B is called the smudging bound. Using the following Theorem 3, we can estimate the smudging bound B to make sure that the construction in Section 2.2 satisfies the circuit privacy.

Theorem 3 (Circuit Privacy [24]). *Let λ be a security parameter and p, q, n, ℓ', χ, B be as defined in Section 2.2. Suppose χ is subgaussian with parameter σ. If n, ℓ', σ, N, $\gamma_R=poly(\lambda)$ and $B=2^{\omega(\log\lambda)}rp^2$, then under the* $\mathrm{MLWE}_{n,m,N,q,\chi}$ *assumption with $m=n$, the construction in Section 2.2 is circuit private with respect to the set $S=\mathcal{R}_p^r$. Concretely, let C be a correctness parameter and let B_1, B_2 be bounds. Let $S=\{\boldsymbol{y}\in\mathcal{R}_p^r\ :\ \|y\|_1\le B_1 \text{ and } \|y\|_2\le B_2\}$. Then under the* $\mathrm{MLWE}_{n,m,N,q,\chi}$ *assumption with $m=n$, for every efficient adversary $\mathcal{A}$ restricted to strategies in S, there exists an efficient simulator $\mathcal{S}$ where*

$$\Pr[\mathrm{ExpCP}_{\Pi_{Enc},\mathcal{A},\mathcal{S}}(1^\lambda)=1]\le\frac{1}{2}+\epsilon+negl(\lambda),$$

and

$$\epsilon=(4n+2)N\ell'\exp(-\pi C^2)+\frac{d\ell'(\gamma_R B_2 C\sigma+\gamma_R B_1/2+2\gamma_R nC^2\sigma^2)}{B} \tag{4}$$

2.3 zkSNARK from Linear Only Encryption and LPCP

Let $\mathcal{CS}=\{\mathcal{CS}_\kappa\}_{\kappa\in\mathbb{N}}$ be a family of R1CS systems over the field $\mathbb{F}$. One can construct a Linear PCP (LPCP) for an R1CS system using the QAP construction given by [18]. The construction relies on the following building blocks:

- Let $\Pi_{\text{LPCP}}=(\mathcal{Q}_{\text{LPCP}},\mathcal{P}_{\text{LPCP}},\mathcal{V}_{\text{LPCP}})$ be a ℓ-query input-oblivious linear PCP for $\mathcal{CS}$. Also Let N_c be the query length.

- Let $\Pi_{\text{Enc}}=(\mathbf{Setup}_{\text{Enc}},\mathbf{Encrypt}_{\text{Enc}},\mathbf{Add}_{\text{Enc}},\mathbf{Decrypt}_{\text{Enc}})$ be a secret-key linear only vector encryption scheme over $\mathbb{F}^\ell$.

The designated verifier zkSNARK $\Pi_{\text{SNARK}}=(\mathbf{Setup},\mathbf{Prove},\mathbf{Verify})$ for $\mathcal{R}_{\mathcal{CS}}$ is as follows:

- $\mathbf{Setup}(1^\lambda,1^\kappa)$: On input of security parameter λ and index κ, the setup runs $(\boldsymbol{Q},\mathbf{st}_{\text{LPCP}})\leftarrow\mathcal{Q}_{\text{LPCP}}(1^\kappa)$ where $\boldsymbol{Q}\in\mathbb{F}^{N_c\times\ell}$. For each $i\in[N_c]$, let $\boldsymbol{q}_i^\top\in\mathbb{F}^\ell$ denote the i-th row of $\boldsymbol{Q}$. Then sample $(\mathbf{pp},\mathbf{sk})\leftarrow\mathbf{Setup}_{\text{Enc}}(1^\lambda,1^\kappa)$ and compute $\mathbf{ct}_i\leftarrow\mathbf{Encrypt}_{\text{Enc}}(\mathbf{sk},\boldsymbol{q}_i^\top)$ for $i\in[N_c]$. Output the common reference string as $\mathbf{crs}=(\kappa,\mathbf{pp},\{\mathbf{ct}_i\}_{i\in[N_c]})$ and $\mathbf{st}=(\mathbf{sk},\mathbf{st}_{\text{LPCP}})$ as verification key.

- $\mathbf{Prove}(\mathbf{crs},\boldsymbol{x},\boldsymbol{\omega})$: On input of $\mathbf{crs}=(\kappa,\mathbf{pp},\{\mathbf{ct}_i\}_{i\in[N_c]})$, the statement $\boldsymbol{x}$ and the witness $\boldsymbol{\omega}$, the prover constructs $\boldsymbol{\sigma}\in\mathbb{F}^{N_c}$ as shown in algorithm $\mathcal{P}(1^\kappa,\boldsymbol{x},\boldsymbol{\omega})$ in Section A. The prover then homomorphically computes linear operation $\mathbf{ct}^*\leftarrow\mathbf{Add}_{\text{Enc}}(\mathbf{pp},\{\mathbf{ct}_i\}_{i\in[N_c]},\{\sigma_i\}_{i\in[N_c]})$. It outputs the proof $\pi=\mathbf{ct}^*$.

- $\mathbf{Verify}(\mathbf{st},\boldsymbol{x},\pi)$: On the input of the verification key $\mathbf{st}=(\mathbf{sk},\mathbf{st}_{\text{LPCP}})$, the statement $\boldsymbol{x}$ and the proof $\pi=\mathbf{ct}^*$, the verifier first decrypts the encrypted proof and gets $\boldsymbol{a}\leftarrow\mathbf{Decrypt}_{\text{Enc}}(\mathbf{sk},\mathbf{ct}^*)$. If $\boldsymbol{a}=\perp$ the verifier outputs 0, otherwise it outputs $\mathcal{V}(\mathbf{st}_{\text{LPCP}},\boldsymbol{x},\boldsymbol{a})$.

2.4 Linear Commitment Scheme

We are using the commitment scheme described in [29]. Suppose we want to commit a message vector $\boldsymbol{m}=(m_1,m_2,\cdots,m_\ell)\in\mathcal{R}_q^\ell$ for $\ell\geq 1$ and that module ranks k and λ are required for MSIS and MLWE security, respectively. Then we consider $n=k+\lambda+\ell$ and in key generation, a matrix $\boldsymbol{B}\xleftarrow{\$}\mathcal{R}_q^{k\times n}$ and vectors $\boldsymbol{a}_1,\boldsymbol{a}_2,\cdots,\boldsymbol{a}_\ell\xleftarrow{\$}\mathcal{R}_q^n$ are generated and output as public parameters. To commit to the message $\boldsymbol{m}$, we first sample $\boldsymbol{s}\leftarrow\chi^n$. Now, there are two parts of the commitment scheme: the binding part and the message encoding part. In particular, we compute

$$\begin{aligned}\boldsymbol{t}_s&=\boldsymbol{B}\boldsymbol{s}\\ \boldsymbol{t}_m^{(i)}&=\boldsymbol{a}_i^\top\boldsymbol{s}+m_i\quad\text{ for } 1\leq i\leq\ell\end{aligned}$$

The $\boldsymbol{t}_s$ forms the binding part and each $\boldsymbol{t}_m^{(i)}$ encodes a message polynomial m_i. The hiding property of the commitment scheme is established via the duality between the decisional Knapsack and MLWE problem. Also, the binding property of the commitment scheme is established via the duality between the search Knapsack and MSIS problem. We refer to the work by Baum et al. [8] for a more detailed discussion.

2.5 Rejection Sampling

In lattice-based zero knowledge schemes [28,29,27], the prover outputs a vector $\boldsymbol{z}$ whose distribution should be independent of a secret randomness vector $\boldsymbol{s}$, so that $\boldsymbol{z}$ does not leak any information about $\boldsymbol{s}$. The prover computes $\boldsymbol{z}=c\boldsymbol{s}+\boldsymbol{y}$, where c is a challenge polynomial sampled from challenge space $\mathcal{C}$ and $\boldsymbol{y}$ is the masking vector. The rejection sampling is used to remove the dependency of $\boldsymbol{z}$ on $\boldsymbol{s}$.

$\underline{\mathrm{Rej}_1(\boldsymbol{z},\boldsymbol{v},\sigma)}$

1. **If** $\langle \boldsymbol{z},\boldsymbol{v}\rangle < 0$
2. return 1 (i.e. *reject*)
3. $u \leftarrow [0,1)$
4. **If** $u > \frac{1}{M}\cdot\exp\left(\frac{-2\langle \boldsymbol{z},\boldsymbol{v}\rangle - \|\boldsymbol{v}\|^2}{2\sigma^2}\right)$
5. return 1 (i.e. *reject*)
6. **Else**
7. return 0 (i.e. *accept*)

Fig. 2: Rejection sampling algorithm

Lemma 2 (Rejection Sampling [29]). *Let $V \subseteq \mathcal{R}^\ell$ be a set of polynomials with norm at most T and $\rho : V \rightarrow [0,1]$ be a probability distribution. Fix a standard deviation $\sigma = \gamma T$. Let $M = \exp(1/(2\gamma^2))$. Now sample $\boldsymbol{v} \leftarrow \rho$ and $\boldsymbol{y} \leftarrow D_\sigma^\ell$, set $\boldsymbol{z} = \boldsymbol{v} + \boldsymbol{y}$, and run $b \leftarrow \mathrm{Rej}_1(\boldsymbol{z},\boldsymbol{v},\sigma)$ as defined in Fig. 2. Then the probability that $b=0$ is at least $1/(2M)$ and the distribution of $(\boldsymbol{v},\boldsymbol{z})$, conditioned on $b=0$, is identical to the distribution of $\mathcal{F}$ where $\mathcal{F}$ is defined as follows: sample $\boldsymbol{v} \leftarrow \rho$, $\boldsymbol{z} \leftarrow D_\sigma^\ell$ conditioned on $\langle \boldsymbol{v},\boldsymbol{z}\rangle \geq 0$ and output $(\boldsymbol{v},\boldsymbol{z})$.*

The parameters σ and repetition rate M in Lemma 2 are selected by choosing M to be the upper-bound on:

$$\frac{D_\sigma^\ell}{D_{\boldsymbol{v},\sigma}^\ell} = \exp\left(\frac{-2\langle \boldsymbol{z},\boldsymbol{v}\rangle + \|\boldsymbol{v}\|^2}{2\sigma^2}\right) \leq M \tag{5}$$

Recently, Lyubashevsky et al. [29] proposed a rejection sampling algorithm Fig. 2 where it forces $\boldsymbol{z}$ to satisfy $\langle \boldsymbol{z},\boldsymbol{v}\rangle \geq 0$, otherwise it aborts. With this additional

assumption, we can set M in the following way:

$$\exp\left(\frac{-2\langle \boldsymbol{z},\boldsymbol{v}\rangle+\|\boldsymbol{v}\|^2}{2\sigma^2}\right)\leq \exp\left(\frac{\|\boldsymbol{v}\|^2}{2\sigma^2}\right)=M \tag{6}$$

Hence, for $M\approx 3$ one would select $\sigma=0.675\|\boldsymbol{v}\|$. Note that the probability for $\boldsymbol{z}\leftarrow D_\sigma^\ell$ that $\langle \boldsymbol{z},\boldsymbol{v}\rangle\geq 0$ is at least $1/2$. Hence, the expected number of rejections would be at most $2M=6$.

2.6 Invertible Elements in $\mathcal{R}_q$ and the Challenge Space

Lemma 3 ([30]). *Let $N\geq d>1$ be powers of 2 and $p\equiv 2d+1 \pmod{4d}$ be a prime. Then X^N+1 factors into d irreducible polynomials $X^{N/d}-r_j$ modulo p and any $y\in\mathcal{R}_p\setminus\{0\}$ that satisfies*

$$\|y\|_\infty<\frac{1}{\sqrt{d}}\cdot p^{1/d} \quad \textit{or} \quad \|y\|_2<p^{1/d}$$

is invertible in $\mathcal{R}_p$.

Note that for any integer $q=q_1\cdot q_2\cdot ....\cdot q_t$, $\mathcal{R}_q$ is isomorphic to $\mathcal{R}_{q_1}\times\cdots\times\mathcal{R}_{q_t}$. Therefore, we can say that any element $y\in\mathcal{R}_q$ is invertible if $\|y\|_\infty<\frac{1}{\sqrt{d}}\cdot q_i^{1/d}$ for all $1\leq i\leq t$. We will need the challenge space of our zero-knowledge proof to consist of short elements such that every difference between distinct elements is invertible in $\mathcal{R}_q$ and $\mathcal{R}_p$. This property is crucial to the soundness of our zero-knowledge proof of commitment opening. For practical purposes, we would also like to define our sets so that they are easy to sample from. One common way to define this challenge space is as

$$\mathcal{C}=\{c\in\mathcal{R}\mid \|c\|_\infty=1,\|c\|_1=\nu\}$$

If we would like the size of $\mathcal{C}$ to be at least 2^λ, then we need to set ν such that $\binom{n}{\nu}\cdot 2^\nu>2^\lambda$. For example, if $N=\lambda=128$, then we can set $\nu=31$. Throughout the paper, we will be assuming that the parameters of the ring $\mathcal{R}_q$ and $\mathcal{R}_p$ are set in such a way that all nonzero elements of ℓ_∞-norm at most 2 and are invertible in $\mathcal{R}_q$. This implies that for any two distinct $c,c'\in\mathcal{C}$, the difference $c-c'$ is invertible in $\mathcal{R}_q$ and $\mathcal{R}_p$. For convenience, we define this set of differences as $\bar{\mathcal{C}}=\{c-c'\mid c\neq c'\in\mathcal{C}\}$.

3 Proof of Linear Relation and Quadratic Relation

In this section, we detail the core cryptographic tools that will serve as building blocks for our hybrid verification mechanism. We present protocols for proving linear and quadratic relations over committed data, which will later be used by the designated verifier to generate the publicly verifiable auxiliary information. Recently [28] have given a general framework to prove linear and multiplicative relations. We are following the work [27] to construct the proof of linear and quadratic relation in $\mathcal{R}_p$ instead of $\mathcal{R}_q$, and we are using the commitment scheme given in [28,29].

3.1 Proof of Linear Relation

Suppose that prover has published $\boldsymbol{t}_s = \boldsymbol{B}\boldsymbol{s} + \boldsymbol{B}'\boldsymbol{s}'$ and two encoding of messages m, m' as $\boldsymbol{a}^\top \boldsymbol{s} + m$ and $\boldsymbol{a}'^\top \boldsymbol{s}' + m'$, respectively. Here $\boldsymbol{t}_s$ binds both $\boldsymbol{s}$ and $\boldsymbol{s}'$. Now the prover claims that $m' = h \cdot m \pmod{p}$ for some $h \in \mathcal{R}_p$. In Fig. 3, the interaction between prover and verifier is given.

$\underline{\Pi_{\text{Lin}}}$

Public information : $\boldsymbol{B}, \boldsymbol{B}' \in \mathcal{R}_q^{k \times n}$; $\boldsymbol{a}, \boldsymbol{a}' \in \mathcal{R}_q^n$ that is used to define $\boldsymbol{t}_s = \boldsymbol{B}\boldsymbol{s} + \boldsymbol{B}'\boldsymbol{s}'$, $\boldsymbol{t}_m = \boldsymbol{a}^\top \boldsymbol{s} + m$ and $\boldsymbol{t}'_m = \boldsymbol{a}'^\top \boldsymbol{s}' + m'$. Also, the fact that $m' = h \cdot m$

Prover information: $\boldsymbol{s}, \boldsymbol{s}' \leftarrow \chi^n$, m, m'

Prover		*Verifier*
$\boldsymbol{y}, \boldsymbol{y}' \leftarrow D_\sigma^n$		
$\boldsymbol{\tau} = \boldsymbol{B}\boldsymbol{y} + \boldsymbol{B}'\boldsymbol{y}'$		
$u = h\boldsymbol{a}^\top \boldsymbol{y} - \boldsymbol{a}'^\top \boldsymbol{y}'$	$\xrightarrow{\boldsymbol{\tau}, u}$	
	$\xleftarrow{c}$	$c \xleftarrow{\$} \mathcal{C}$
Rejection Sample:		
$\boldsymbol{z} = c\boldsymbol{s} + y$		
$\boldsymbol{z}' = c\boldsymbol{s}' + \boldsymbol{y}'$	$\xrightarrow{\boldsymbol{z}, \boldsymbol{z}'}$	Accept iff: 1. $\|\boldsymbol{z}\|_2 \leq \sigma\sqrt{2nd}$ and $\|\boldsymbol{z}'\|_2 \leq \sigma\sqrt{2nd}$ and 2. $\boldsymbol{B}\boldsymbol{z} + \boldsymbol{B}'\boldsymbol{z}' = \boldsymbol{\tau} + c\boldsymbol{t}_s$ and 3. $h\boldsymbol{a}^\top \boldsymbol{z} - \boldsymbol{a}'^\top \boldsymbol{z}' = c(h\boldsymbol{t}_m - \boldsymbol{t}'_m) + u \pmod{p}$

Fig. 3: Zero-knowledge proof of linear relation

Lemma 4. *The protocol Π_{Lin} in Fig. 3 is a proof of knowledge $(\bar{\boldsymbol{s}}, \bar{\boldsymbol{s}}', \bar{c}) \in \mathcal{R}_q^n \times \mathcal{R}_q^n \times \bar{\mathcal{C}}$ satisfying*

1. $\boldsymbol{B}\bar{\boldsymbol{s}} + \boldsymbol{B}'\bar{\boldsymbol{s}}' = \boldsymbol{t}_s$
2. $\|\bar{\boldsymbol{s}}\bar{c}\| \leq 2\sigma\sqrt{2nd}$ *and* $\|\bar{\boldsymbol{s}}'\bar{c}\| \leq 2\sigma\sqrt{2nd}$
3. $h(\boldsymbol{t}_m - \boldsymbol{a}^\top \bar{\boldsymbol{s}}) = \boldsymbol{t}'_m - \boldsymbol{a}'^\top \bar{\boldsymbol{s}}' \pmod{p}$
4. *Finding another* $(\bar{\boldsymbol{s}}_1, \bar{\boldsymbol{s}}'_1, \bar{c}_1)$ *is computationally hard.*
5. *Let* $(\boldsymbol{\tau}, u, c_1, \boldsymbol{z}_1, \boldsymbol{z}'_1)$ *and* $(\boldsymbol{\tau}, u, c_2, \boldsymbol{z}_2, \boldsymbol{z}'_2)$ *be two valid transcripts of the protocol. Then it holds that* $\boldsymbol{z}_1 - c_1\bar{\boldsymbol{s}} = \boldsymbol{z}_2 - c_2\bar{\boldsymbol{s}}$ *and* $\boldsymbol{z}'_1 - c_1\bar{\boldsymbol{s}}' = \boldsymbol{z}'_2 - c_2\bar{\boldsymbol{s}}'$.

The proof of Lemma 4 is discussed in Appendix B.

3.2 Proof of Quadratic Relations

The proof system follows from the work in [27]. However, in [27] they have given a proof system that proves that $(\boldsymbol{s}, \boldsymbol{m})$ is a root of the quadratic equation say

$F(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{R}_2 \boldsymbol{x} + \boldsymbol{r}_1^\top \boldsymbol{x} + r_0$ over $\mathcal{R}_q$. Here we are only trying to prove that $\boldsymbol{m}$ is a root of the said quadratic equation in $\mathcal{R}_p$. Therefore, it is enough to consider the commitment scheme in [28]. In Fig. 4, the protocol is depicted.

$\underline{\Pi^{(1)}_{\text{Quad}}}$

Public information : $\boldsymbol{B} \in \mathcal{R}_q^{k \times n}$ and $\boldsymbol{a_i} \in \mathcal{R}_q^n$ for $i \in [\kappa]$, that is used to define

$\boldsymbol{t}_s = \boldsymbol{B}\boldsymbol{s}$

$\boldsymbol{t}_m^{(i)} = \boldsymbol{a}_i^\top \boldsymbol{s} + m_i, \ \ \forall i \in [\kappa]$

and $\boldsymbol{R}_2 \in \mathcal{R}_p^{\kappa \times \kappa}$, $\boldsymbol{r}_1 \in \mathcal{R}_p^\kappa$ and $r_0 \in \mathcal{R}_p$ such that $F(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{R}_2 \boldsymbol{x} + \boldsymbol{r}_1^\top \boldsymbol{x} + r_0$, and a random vector $\boldsymbol{b} \in \mathcal{R}_q^n$.

Prover information: $\boldsymbol{s} \leftarrow \chi^n$ and the message $\boldsymbol{m} = (m_1, m_2, \cdots, m_\kappa)^\top$

Prover		*Verifier*
$\boldsymbol{y} \leftarrow D_\sigma^n$		
$\boldsymbol{\tau} = \boldsymbol{B} \cdot \boldsymbol{y}$		
$\boldsymbol{u} = \begin{bmatrix} -\boldsymbol{a}_1^\top \boldsymbol{y} \\ \vdots \\ -\boldsymbol{a}_\kappa^\top \boldsymbol{y} \end{bmatrix}$		
$g = \boldsymbol{m}^\top \boldsymbol{R}_2 \boldsymbol{u} + \boldsymbol{u}^\top \boldsymbol{R}_2 \boldsymbol{m} + \boldsymbol{r}_1^\top \boldsymbol{u}$		
$h = \boldsymbol{b}^\top \boldsymbol{s} + g$		
$v = \boldsymbol{u}^\top \boldsymbol{R}_2 \boldsymbol{u} + \boldsymbol{b}^\top \boldsymbol{y}$	$\xrightarrow{\boldsymbol{\tau}, h, v}$	
		$c \xleftarrow{\$} \mathcal{C}$
	$\xleftarrow{c}$	
Rejection Sample:		
$\boldsymbol{z} = c\boldsymbol{s} + \boldsymbol{y}$		
	$\xrightarrow{\boldsymbol{z}}$	$\boldsymbol{w} = \begin{bmatrix} c\boldsymbol{t}_m^{(1)} - \boldsymbol{a}_1^\top \boldsymbol{z} \\ \vdots \\ c\boldsymbol{t}_m^{(\kappa)} - \boldsymbol{a}_\kappa^\top \boldsymbol{z} \end{bmatrix}$
		$f = ch - \boldsymbol{b}^\top \boldsymbol{z} + v$
		Accept iff:
		1. $\|\boldsymbol{z}\|_2 \leq \sigma\sqrt{2nd}$ and
		2. $\boldsymbol{B}\boldsymbol{z} = \boldsymbol{\tau} + c \cdot \boldsymbol{t}_s$ and
		3. $\boldsymbol{w}^\top \boldsymbol{R}_2 \boldsymbol{w} + c\boldsymbol{r}_1^\top \boldsymbol{w} + c^2 r_0 = f \pmod p$

Fig. 4: Zero-knowledge proof of quadratic relation

Lemma 5. *There exists a simulator $\mathcal{S}$ that, without access to private information $\boldsymbol{s}$, $\boldsymbol{m}$, outputs a simulation of a commitment $(\boldsymbol{t}_s, \{\boldsymbol{t}_m^{(i)}\}_{i \in [\kappa]})$ along with non-aborting transcript of the protocol between the prover $\mathcal{P}$ and the verifier $\mathcal{V}$ such that for every*

PPT algorithm $\mathcal{A}$ that has advantage ε in distinguishing the simulated commitment and transcript from the real commitment and transcript, whenever the prover does not abort, there is an algorithm $\mathcal{A}'$ with advantage $\varepsilon/2+\mathrm{negl}(\lambda)$ in distinguishing Extended-$\mathtt{MLWE}_{k+\kappa+1,n-k-\kappa-1,\chi,\mathcal{C},\sigma}$, where λ is the security parameter.

Lemma 6. *For soundness, there is an extractor $\mathcal{E}$ with the following properties. When given rewindable black-box access to a probabilistic prover $\mathcal{P}^*$, which convinces $\mathcal{V}$ with probability $\varepsilon \geq 2/|\mathcal{C}|$, extractor $\mathcal{E}$ with probability at least $\varepsilon - 2/|\mathcal{C}|$, either outputs $(\bar{\boldsymbol{s}},\bar{\boldsymbol{m}})\in\mathcal{R}_q^{n+K}$ and $\bar{c}\in\bar{\mathcal{C}}$ such that*

1. $\boldsymbol{t}_s = \boldsymbol{B}\bar{\boldsymbol{s}}$ *and* $\boldsymbol{t}_m^{(i)} = \boldsymbol{a}_i^\top \bar{\boldsymbol{s}} + \bar{m}_i$ *for* $i\in[\kappa]$
2. $\|\bar{c}\|_\infty \leq 2$
3. $\|\bar{c}\bar{\boldsymbol{s}}\| \leq 2\sigma\sqrt{2nd}$
4. $\bar{\boldsymbol{m}}^\top \boldsymbol{R}_2\bar{\boldsymbol{m}} + \bar{r}_1^\top\bar{\boldsymbol{m}} + r_0 = 0$

or a $\mathtt{MSIS}_{k,n,B}$ solution for $B = 4\eta\sigma\sqrt{2nd}$.

The proof of Lemma 5 and Lemma 6 are discussed in Appendix C and Appendix D respectively.

4 Hybrid-Verifier zkSNARK

In this section, we first introduce the notion of Hybrid Verifier zkSNARK (HV-zkSNARK), and then we will introduce a lattice-based construction of HV-zkSNARK.

Definition 9 (HV-zkSNARK). *Let $\mathcal{CS}=\{\mathcal{CS}_\kappa\}_{\kappa\in\mathbb{N}}$ be a family of R1CS systems over a finite field $\mathcal{F}$, where $|\mathcal{CS}_\kappa| \geq s(\kappa)$ for some fixed polynomial $s(\cdot)$. A* HV-zkSNARK *in the preprocessing model for $\mathcal{CS}$ is a tuple with four algorithms, $\Pi_{\mathrm{HV}} =$ (**Setup**,**Prove**,**DVerify**,**PVerify**) with the following properties:*

- $(\mathbf{crs},\mathbf{sk},\mathbf{vk}) \leftarrow \mathbf{Setup}(1^\lambda,1^\kappa)$*: On input the security parameter λ and the system index κ, the setup algorithm outputs a common reference string* **crs***, secret verification key* **sk** *and a public verification key* $\boldsymbol{vk}$*.*
- $\pi \leftarrow \mathbf{Prove}(\mathbf{crs},\boldsymbol{x},\boldsymbol{\omega})$*: On input a common reference string* **crs***, a statement $\boldsymbol{x}$, and a witness $\boldsymbol{\omega}$, the prove algorithm outputs a proof π.*
- $(b,\mathbf{aux}) \leftarrow \mathbf{DVerify}(\mathbf{sk},\boldsymbol{x},\pi)$*: On input the secret verification key* **sk***, a statement $\boldsymbol{x}$ and a proof π, the verification algorithm outputs a bit $b\in\{0,1\}$ that specifies whether the verification accepts or rejects and it generates* **aux** *that helps a public verifier verify the proof.*
- $b' \leftarrow \mathbf{PVerify}(\mathbf{aux},\mathbf{vk},\boldsymbol{x},\pi)$*: On input the auxiliary information* $\boldsymbol{aux}$*, public verification key* **vk***, a statement $\boldsymbol{x}$ and a proof π, the verification algorithm outputs a bit $b'\in\{0,1\}$ that specifies whether the verification accepts or rejects.*

Moreover, Π_{HV} should satisfy the following properties:

- **Completeness**: For all security parameters $\lambda\in\mathbb{N}$, system indices $\kappa\in\mathbb{N}$, and instances $(\boldsymbol{x},\boldsymbol{\omega})$ where $\mathcal{CS}_\kappa(\boldsymbol{x},\boldsymbol{\omega})=1$.

$$\Pr\left[\begin{array}{c} b=1 \\ \wedge \\ \mathbf{PVerify}(\mathbf{aux},\mathbf{vk},\boldsymbol{x},\pi)=1 \end{array}\middle|\begin{array}{c} (\mathbf{crs},\mathbf{sk},\mathbf{vk})\leftarrow\mathbf{Setup}(1^\lambda,1^\kappa) \\ \pi\leftarrow\mathbf{Prove}(\mathbf{crs},\boldsymbol{x},\boldsymbol{\omega}) \\ (b,\mathbf{aux})\leftarrow\mathbf{DVerify}(\mathbf{sk},\boldsymbol{x},\pi) \end{array}\right] = 1$$

- **Knowledge**: For all polynomial-size provers $\mathcal{P}^*$, there exists a polynomial-size extractor $\mathcal{E}$ such that for all security parameters $\lambda \in \mathbb{N}$, system indices $\kappa \in \mathcal{N}$, and auxiliary inputs $z \in \{0,1\}^{\mathrm{poly}(\lambda)}$,

$$\Pr\left[\begin{array}{c|l} b=1 \wedge b'=1 & (\mathbf{crs},\mathbf{sk},\mathbf{vk}) \leftarrow \mathbf{Setup}(1^\lambda,1^\kappa) \\ \wedge & (\boldsymbol{x},\pi) \leftarrow \mathcal{P}^*(1^\lambda,1^\kappa,\mathbf{crs};z) \\ \mathcal{CS}_\kappa(\boldsymbol{x},\boldsymbol{\omega}) \neq 1 & (b,\mathbf{aux}) \leftarrow \mathbf{DVerify}(\mathbf{sk},\boldsymbol{x},\pi) \\ & b' \leftarrow \mathbf{PVerify}(\mathbf{aux},\mathbf{vk},\boldsymbol{x},\pi) \\ & \boldsymbol{\omega} \leftarrow \mathcal{E}(1^\lambda,1^\kappa,\mathbf{crs},\mathbf{sk},\mathbf{vk},\boldsymbol{x};z) \end{array}\right] = \mathrm{negl}(\lambda)$$

- **Efficiency**: There exist a universal polynomial p (independent of $\mathcal{CS}$) such that Setup and Prove run in time $p(\lambda+|\mathcal{CS}_\kappa|)$, Verify runs in time $p(\lambda+|\boldsymbol{x}|+\log|\mathcal{CS}_\kappa|)$, and the proof size is $p(\lambda+\log|\mathcal{CS}_\kappa|)$.
- **Zero-Knowledge**: $\Pi_{\mathrm{HV}} = (\mathbf{Setup},\mathbf{Prove},\mathbf{DVerify},\mathbf{PVerify})$ is zero knowledge if there exists an efficient simulator $\mathcal{S}_{\mathrm{HV}} = (\mathcal{S}_1,\mathcal{S}_2,\mathcal{S}_3)$ such that $\mathcal{S}_3$ can simulate for all $\kappa \in \mathbb{N}$ and all efficient adversaries $\mathcal{A}$, we have that

$$\Pr[\mathrm{ExptZK}^{(1)}_{\Pi_{\mathrm{HV}},\mathcal{A},\mathcal{S}_{\mathrm{HV}}}(1^\lambda,1^\kappa)=1] \leq 1/2+\mathrm{negl}(\lambda) \tag{7}$$

$$\Pr[\mathrm{ExptZK}^{(2)}_{\Pi_{\mathrm{HV}},\mathcal{A},\mathcal{S}_{\mathrm{HV}}}(1^\lambda,1^\kappa)=1] \leq 1/2+\mathrm{negl}(\lambda) \tag{8}$$

where the experiment $\mathrm{ExptZK}^{(1)}_{\Pi_{\mathrm{HV}},\mathcal{A},\mathcal{S}_{\mathrm{HV}}}(1^\lambda,1^\kappa)$ is defined as follows:

1. The challenger samples $b \xleftarrow{\$} \{0,1\}$.
   - If $b=0$, the challenger computes $(\mathbf{crs},\mathbf{sk},\mathbf{vk}) \leftarrow \mathbf{Setup}(1^\lambda,1^\kappa)$ and gives $(\mathbf{crs},\mathbf{sk},\mathbf{vk})$ to $\mathcal{A}$.
   - If $b=1$, the challenger computes $(\widetilde{\mathbf{crs}},\widetilde{\mathbf{sk}},\widetilde{\mathbf{vk}},\mathbf{st}_{\mathcal{S}}) \leftarrow \mathcal{S}_1(1^\lambda,1^\kappa)$ and gives $(\widetilde{\mathbf{crs}},\widetilde{\mathbf{sk}},\widetilde{\mathbf{vk}})$ to $\mathcal{A}$.
2. The adversary $\mathcal{A}$ outputs a statement $\boldsymbol{x}$ and a witness $\boldsymbol{\omega}$.
3. If $\mathcal{CS}(\boldsymbol{x},\boldsymbol{\omega}) \neq 1$, then the experiment halts and output 0. Otherwise, the challenger proceeds as follows:
   - If $b=0$, the challenger replies with $\pi \leftarrow \mathbf{Prove}(\mathbf{crs},\boldsymbol{x},\boldsymbol{\omega})$.
   - If $b=1$, the challenger replies with $\widetilde{\pi} \leftarrow \mathcal{S}_2(\mathbf{st}_{\mathcal{S}},\boldsymbol{x})$.

   At the end of the experiment, $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$. The output of the experiment is 1 if $b'=b$ and is 0 otherwise.

Essentially, winning $\mathrm{ExptZK}^{(1)}_{\Pi_{\mathrm{HV}},\mathcal{A},\mathcal{S}_{\mathrm{HV}}}(1^\lambda,1^\kappa)$ with non-negligible advantage means that some information about the witness $\boldsymbol{\omega}$ is being leaked to the adversary. Similarly, winning $\mathrm{ExptZK}^{(2)}_{\Pi_{\mathrm{HV}},\mathcal{A},\mathcal{S}_{\mathrm{HV}}}(1^\lambda,1^\kappa)$ with non-negligible advantage means that some information about the secret verification key $\mathbf{sk}$ is being leaked to the adversary. The experiment $\mathrm{ExptZK}^{(2)}_{\Pi_{\mathrm{HV}},\mathcal{A},\mathcal{S}_{\mathrm{HV}}}(1^\lambda,1^\kappa)$ is defined as follows

1. The adversary $\mathcal{A}$ outputs a statement $\boldsymbol{x}$ and a witness $\boldsymbol{\omega}$.
2. If $\mathcal{CS}(\boldsymbol{x},\boldsymbol{\omega}) \neq 1$, then the experiment halts and output 0. Otherwise the challenger computes $(\mathbf{crs},\mathbf{sk},\mathbf{vk}) \leftarrow \mathbf{Setup}(1^\lambda,1^\kappa), \pi \leftarrow \mathbf{Prove}(\mathbf{crs},\boldsymbol{x},\boldsymbol{\omega})$ and gives $(\mathbf{crs},\mathbf{vk},\pi)$ to $\mathcal{A}$.
3. The challenger samples $b \xleftarrow{\$} \{0,1\}$.
   - If $b=0$, the challenger computes $(d,\mathbf{aux}) \leftarrow \mathbf{DVerify}(\mathbf{sk},\boldsymbol{x},\pi)$ and gives $\mathbf{aux}$ to $\mathcal{A}$.

- If $b=1$, the challenger computes $(\widetilde{d},\widetilde{\mathbf{aux}})\leftarrow\mathcal{S}_3(1^\lambda,1^\kappa)$ and reports to $\mathcal{A}$. At the end of the experiment, $\mathcal{A}$ outputs a bit $b'\in\{0,1\}$. The output of the experiment is 1 if $b'=b$ and is 0 otherwise.

- **Detection of Malicious Party**: When PVerify algorithm returns 0, *i.e.*, the verification fails, then there can be multiple cases:
  (i) The prover does not have the witness $\boldsymbol{\omega}$ corresponding to the statement $\boldsymbol{x}$ such that $\mathcal{CS}(\boldsymbol{x},\boldsymbol{\omega})=1$ and it has provided an invalid proof π.
  (ii) The prover has the witness and it has generated a valid proof π, but the designated-verifier has generated the **aux** dishonestly.
  (iii) Both prover and designated-verifier have dishonestly generated the π and **aux**, respectively.
  In all of the above cases, the verification fails, but the public-verifier should have the ability to detect which one of the prover and designated-verifier has been dishonest.

In our construction of HV-zkSNARK, the **Setup**, **Prove** are the same as the zk-SNARK construction proposed by Ishai et al. [24], which are briefly described in Section 2.3. The **Dverify** in HV-zkSNARK would not only verify the proof but also generate **aux** that is used by the public verifier to verify the proof using **PVerify**. Therefore, we need to construct the **Dverify** and **Pverify** for our HV-zkSNARK. In the next section, we have described the public verification mechanism. This is essentially the interaction between the designated verifier and the public verifier, *i.e.* the operations **DVerify** and **PVerify**.

4.1 Public Verification

In the designated verifier protocol by Ishai et al. [24], the prover sends encryption $\mathbf{ct}^*$ of $\hat{\boldsymbol{\pi}}\in\mathcal{R}_p^\ell$ as proof. Only the designated verifier with the secret key can decrypt and check the validity of the proof. What we want to achieve here is that the public-verifier can take $\mathbf{ct}^*$ and verify the validity of the proof, but it is not straightforward without the secret key **sk** that was used for encryption. Also, a vector $\hat{\mathbf{g}}$ is to be added to $\hat{\boldsymbol{\pi}}$ to get $\hat{\boldsymbol{r}}$ in the verification stage (see Appendix A). Therefore, we would assume here that $\mathbf{ct}_\mathrm{g}\leftarrow\mathbf{Encrypt}_\mathrm{Enc}(\mathbf{sk},\hat{\mathbf{g}})$ is also made public as **vk** as it is shown in Fig. 1. This means that the public verifier can add the ciphertexts $\mathbf{ct}^*$ and $\mathbf{ct}_\mathrm{g}$ to generate $\mathbf{ct}_\mathrm{r}$, the encryption of $\hat{\boldsymbol{r}}$. All we need now is to find a way to convince the public verifier that a certain quadratic equation satisfies.

Lets look at the encryption $\mathbf{ct}_\mathrm{r}$ of $\hat{\boldsymbol{r}}\in\mathcal{R}_p^\ell$. According to the encryption scheme defined in Section 2.2, we have secret $\mathbf{sk}=(\boldsymbol{S},\boldsymbol{T})\in\mathcal{R}_q^{n\times\ell'}\times\mathcal{R}_p^{\tau\times\ell}$, and there exist random vector $\boldsymbol{a}\in\mathcal{R}_q^n$ and error vector $\boldsymbol{e}\in\mathcal{R}_q^{\ell'}$ such that

$$\mathbf{ct}_\mathrm{r}=\left(\boldsymbol{a},\boldsymbol{S}^\top\boldsymbol{a}+p\boldsymbol{e}+\boldsymbol{u}\right) \tag{9}$$

where $\boldsymbol{u}=\left[\hat{\boldsymbol{r}}^\top\mid(\boldsymbol{T}\hat{\boldsymbol{r}})^\top\right]^\top$. Let us consider $\boldsymbol{S}=\left[\boldsymbol{s}_1^\top\mid\boldsymbol{s}_2^\top\mid\cdots\mid\boldsymbol{s}_{\ell'}^\top\right]^\top$ and $\boldsymbol{u}=(u_1,u_2,\cdots,u_{\ell'})^\top$ such that $\hat{\boldsymbol{r}}=(u_1,u_2,\cdots,u_\ell)^\top$. Then we can rewrite $\mathbf{ct}_\mathrm{r}$ as

$$\left(\boldsymbol{a},\left\{\boldsymbol{a}^\top\boldsymbol{s}_i+p\cdot e_i+u_i\right\}_{i\in[\ell']}\right) \tag{10}$$

Let $v_i = p \cdot e_i + u_i$, then $v_i = u_i \pmod p$ for all $i \in [\ell']$. Therefore, $v_i = \hat{r}_i \pmod p$ for all $i \in [\ell]$. The ciphertext in Eq. (10) can now be written as

$$\left(\boldsymbol{a}, \left\{ \boldsymbol{t}_m^{(i)} = \boldsymbol{a}^\top \boldsymbol{s}_i + v_i \right\}_{i \in [\ell']} \right) \tag{11}$$

Let us consider that at the time of secret key $\mathbf{sk} = \{\boldsymbol{s}_i\}_{i \in [\ell']}$ generation, designated verifier have sampled random matrices $\boldsymbol{B}_i \xleftarrow{\$} \mathcal{R}_q^{k \times n}$ and published $\boldsymbol{t}_s^{(i)} = \boldsymbol{B}_i \boldsymbol{s}_i$ for $i \in [\ell']$. The following information is available to the public verifier:

$$\begin{aligned} \boldsymbol{t}_s^{(i)} &= \boldsymbol{B}_i \boldsymbol{s}_i \\ \boldsymbol{t}_m^{(i)} &= \boldsymbol{a}^\top \boldsymbol{s}_i + v_i \end{aligned} \tag{12}$$

Therefore, the encryption in Eq. (11) can now be considered as the linear commitment scheme that we had earlier discussed in Section 2.4. If $(v_1, v_2, \cdots, v_\ell)$ satisfies a quadratic equation in modulo p, then $(\hat{r}_1, \hat{r}_2, \cdots, \hat{r}_\ell)$ also satisfies the quadratic equation in modulus p since $v_i = \hat{r}_i \pmod p$ for all $i \in [\ell]$.

In Fig. 4, the protocol $\Pi_{\text{Quad}}^{(1)}$ proves a quadratic relation between messages that are encoded under the same secret $\boldsymbol{s}$. But in order to generate auxiliary information that helps the public verifier to verify the proof, we need a protocol where each messages are encoded with different secret keys. Also the quadratic function is defined as $F(x) = x_1 x_2 - x_3 - x_4 x_5$ for $\boldsymbol{x} \in \mathcal{R}_p^5$. Not only that, we also need to make sure that neither the prover nor the designated verifier can generate an invalid proof, and even if they did, the public verifier can identify the cheating party.

Therefore we need to modify the $\Pi_{\text{Quad}}^{(1)}$ protocol. In Fig. 5, the protocol $\Pi_{\text{Quad}}^{(2)}$ is depicted. Later, we will also discuss the cheating detection mechanism. The messages $m_i \in \mathcal{R}_p$ are encoded with secrets $\boldsymbol{s}_i \leftarrow \chi^n$ and $e_i \in [-B, B]$ for all $i \in [\kappa]$, where B is the noise smudging bound. Therefore, we consider that the following information is published for all $i \in [\kappa]$:

$$\begin{aligned} \boldsymbol{t}_s^{(i)} &= \boldsymbol{B}_i \boldsymbol{s}_i \\ \boldsymbol{t}_m^{(i)} &= \boldsymbol{a}_i^\top \boldsymbol{s}_i + m_i + p e_i \end{aligned}$$

where $\boldsymbol{B}_i \xleftarrow{\$} \mathcal{R}_q^{k \times n}$ and $\boldsymbol{a}_i \xleftarrow{\$} \mathcal{R}_q^n$. Next, we have proved that the protocol is zero-knowledge, and we have also proved the knowledge soundness of the protocol.

Lemma 7. *There exists a simulator $\mathcal{S}$ that, without access to private information $\boldsymbol{S} = (\boldsymbol{s}_1, \cdots, \boldsymbol{s}_\kappa)$, $\boldsymbol{m}$ in protocol $\Pi_{Quad}^{(2)}$, outputs a simulation of a commitment $(\{\boldsymbol{t}_s^{(i)}, \boldsymbol{t}_m^{(i)}\}_{1 \le i \le \kappa})$ along with non-aborting transcript of the protocol between the prover $\mathcal{P}$ and the verifier $\mathcal{V}$ such that for every algorithm $\mathcal{A}$, the advantage in distinguishing the simulated commitment and transcript from the real commitment and transcript, whenever the prover does not abort, is negligible.*

Lemma 8. *For soundness of $\Pi_{Quad}^{(2)}$, there is an extractor $\mathcal{E}$ with following properties. When given rewindable black-box access to a probabilistic prover $\mathcal{P}^*$, which convinces $\mathcal{V}$ with probability $\varepsilon \ge 2/|\mathcal{C}|$, extractor $\mathcal{E}$ with probability at least $\varepsilon - 2/|\mathcal{C}|$, either outputs $(\bar{\boldsymbol{s}}_1, \bar{\boldsymbol{s}}_2, \cdots, \bar{\boldsymbol{s}}_\kappa, \bar{\boldsymbol{m}}) \in \mathcal{R}_q^{(n+1)\kappa}$ and $\bar{c} \in \bar{\mathcal{C}}$ such that*

$\underline{\Pi^{(2)}_{\text{Quad}}}$

Public information : For $1 \leq i \leq \kappa$, we have $\boldsymbol{B}_i \in \mathcal{R}_q^{k\times n}$, $\boldsymbol{a}_i \in \mathcal{R}_q^n$, $\boldsymbol{t}_s^{(i)} = \boldsymbol{B}_i\boldsymbol{s}_i$ and $\boldsymbol{t}_m^{(i)} = \boldsymbol{a}_i^\top \boldsymbol{s}_i + m_i + pe_i$. Random vectors $\boldsymbol{b},\boldsymbol{d} \xleftarrow{\$} \mathcal{R}_q^n$ are public. Also, the quadratic equation $F(x) = \boldsymbol{x}^\top \boldsymbol{R}_2\boldsymbol{x} + \boldsymbol{r}_1^\top \boldsymbol{x} + r_0$ is known.

Prover information: $\boldsymbol{s}_i \leftarrow \chi^n$ and the messages $\boldsymbol{m} = (m_1,\cdots,m_\kappa)^\top$

Prover		*Verifier*
For each $1\leq i\leq\kappa$ $\boldsymbol{y}_i \leftarrow D_\sigma^n$ $\boldsymbol{\tau}_i = \boldsymbol{B}_i\cdot\boldsymbol{y}_i$ $\boldsymbol{u} = \begin{bmatrix} -\boldsymbol{a}_1^\top\boldsymbol{y}_1 \\ \vdots \\ -\boldsymbol{a}_\kappa^\top\boldsymbol{y}_\kappa \end{bmatrix}$ $\boldsymbol{u}_h = \lfloor \boldsymbol{u}/p \rfloor$ and $\boldsymbol{u}_l = \boldsymbol{u} \pmod p$ $g = \boldsymbol{m}^\top\boldsymbol{R}_2\boldsymbol{u}_l + \boldsymbol{u}_l{}^\top\boldsymbol{R}_2\boldsymbol{m} + \boldsymbol{r}_1^\top\boldsymbol{u}_l$ $h = \boldsymbol{b}^\top\boldsymbol{s}_1 + g$ $v = \boldsymbol{u}_l{}^\top\boldsymbol{R}_2\boldsymbol{u}_l + \boldsymbol{b}^\top\boldsymbol{y}_1$ $g' = \boldsymbol{m}^\top\boldsymbol{R}_2\boldsymbol{m} + \boldsymbol{r}_1^\top\boldsymbol{m} + r_0$ $h' = \boldsymbol{d}^\top\boldsymbol{s}_2 + g'$ $v' = \boldsymbol{d}^\top\boldsymbol{y}_2$		
	$\xrightarrow{\substack{\{\boldsymbol{\tau}_i\}_{1\leq i\leq\kappa}, \\ h,h',\boldsymbol{u}_h,v,v'}}$	
	$\xleftarrow{\quad c \quad}$	$c \xleftarrow{\$} \mathcal{C}$
Rejection Sample: $\boldsymbol{z}_i = c\boldsymbol{s}_i + \boldsymbol{y}_i \quad 1\leq i\leq\kappa$	$\xrightarrow{\{\boldsymbol{z}_i\}_{1\leq i\leq\kappa}}$	
		$\boldsymbol{w} = \begin{bmatrix} c\boldsymbol{t}_m^{(1)} - \boldsymbol{a}_1^\top\boldsymbol{z}_1 \\ \vdots \\ c\boldsymbol{t}_m^{(\kappa)} - \boldsymbol{a}_\kappa^\top\boldsymbol{z}_\kappa \end{bmatrix} - p\boldsymbol{u}_h \pmod p$ $b = ch' - \boldsymbol{d}^\top\boldsymbol{z}_2 + v'$ $f' = ch - \boldsymbol{b}^\top\boldsymbol{z}_1 + v - cb$ Accept iff: 1. For each $1\leq i\leq\kappa$ (a) $\|\boldsymbol{z}_i\|_2 \leq \sigma\sqrt{2nd}$ and (b) $\boldsymbol{B}_i\boldsymbol{z}_i = \boldsymbol{\tau}_i + c\cdot\boldsymbol{t}_s^{(i)}$ 2. $\boldsymbol{w}^\top\boldsymbol{R}_2\boldsymbol{w} + c\boldsymbol{r}_1^\top\boldsymbol{w} + c^2r_0 = f' \pmod p$ 3. $b = 0 \pmod p$

Fig. 5: Zero knowledge proof of quadratic relation with multiple commitments (modified)

1. $\boldsymbol{t}_s^{(i)} = \boldsymbol{B}_i\bar{\boldsymbol{s}}_i$ *and* $\boldsymbol{t}_m^{(i)} = \boldsymbol{a}_i^\top\bar{\boldsymbol{s}}_i + \bar{m}_i$ *for* $1\leq i\leq\kappa$
2. $\|\bar{c}\|_\infty \leq 2$
3. $\|\bar{c}\bar{\boldsymbol{s}}_i\| \leq 2\sigma\sqrt{2nd}$ *for* $1\leq i\leq\kappa$
4. $\bar{\boldsymbol{m}}^\top\boldsymbol{R}_2\bar{\boldsymbol{m}} + \boldsymbol{r}_1^\top\bar{\boldsymbol{m}} + r_0 = 0 \pmod p$

or can solve $\texttt{MSIS}_{k,n,B}$ *problem for* $B = 4\sigma\sqrt{2nd}$.

The proof of Lemma 7 and Lemma 8 are discussed in Appendix E and Appendix F respectively.

4.2 Validity of Encryption

In Section 2.2, we have seen that to encrypt a vector $\boldsymbol{u} \in \mathcal{R}_p^\ell$, we first select a secret matrix $\boldsymbol{T} \in \mathcal{R}_p^{\tau \times \ell}$ and first compute $\boldsymbol{v} = [\boldsymbol{u}^\top \mid (\boldsymbol{T}\boldsymbol{u})^\top]^\top \in \mathcal{R}_p^{\ell'}$, where $\ell' = \ell + \tau$ and this vector $\boldsymbol{v}$ is encrypted. At the time of decryption one first parse $\boldsymbol{v} = [\boldsymbol{v}_1^\top \mid \boldsymbol{v}_2^\top]^\top$ and then checks if $\boldsymbol{T}\boldsymbol{v}_1 = \boldsymbol{v}_2 \pmod p$. Therefore, the designated verifier should also be able to provide proof that the final ciphertext provided by the prover satisfies the above condition without revealing $\boldsymbol{T}$. First we sample a secrets $\tilde{\boldsymbol{S}} \leftarrow \chi^{n \times \tau}$, $\tilde{\boldsymbol{E}} \leftarrow \chi^{\tau \times \ell}$ and random matrices $\tilde{\boldsymbol{A}} \xleftarrow{\$} \mathcal{R}_q^{n \times \ell}$, $\tilde{\boldsymbol{B}}_i \xleftarrow{\$} \mathcal{R}_q^{k \times n}$ for $i \in [\tau]$ and we public the following information

$$\begin{aligned} \tilde{\boldsymbol{t}}_s^{(i)} &= \tilde{\boldsymbol{B}}_i \tilde{\boldsymbol{s}}_i \\ \boldsymbol{C} &= \tilde{\boldsymbol{S}}^\top \tilde{\boldsymbol{A}} + p\tilde{\boldsymbol{E}} + \boldsymbol{T} \end{aligned} \tag{13}$$

where $\tilde{\boldsymbol{s}}_i$ is the i-th column of $\tilde{\boldsymbol{S}}$. Observe that the (i,j)-th element of $\boldsymbol{C}$ is

$$c_{i,j} = \langle \tilde{\boldsymbol{a}}_i, \tilde{\boldsymbol{s}}_j \rangle + p \cdot \tilde{e}_{i,j} + t_{i,j}$$

where $\tilde{e}_{i,j}$, $t_{i,j}$ are the (i,j)-th element of $\tilde{\boldsymbol{E}}$, $\boldsymbol{T}$ respectively and $\tilde{\boldsymbol{a}}_j$ is the j-th column of $\tilde{\boldsymbol{A}}$ for all $i \in [\tau]$ and $j \in [\ell]$. And let us consider the encryption of the vector $\boldsymbol{v} = [\boldsymbol{v}_1^\top \mid \boldsymbol{v}_2^\top]^\top$ along with $\boldsymbol{C}$ is available to the public verifier. Now, the equality $\boldsymbol{T}\boldsymbol{v}_1 = \boldsymbol{v}_2$ can be written as $v_{\ell+i} = \sum_{j \in [\ell]} t_{i,j} v_j$ for all $i \in [\tau]$. Therefore, the designated verifier can prove these τ quadratic equations to the public-verifier using the protocol $\Pi_{\text{Quad}}^{(2)}$.

4.3 Consistency Check for CRS

While our security model considers malicious parties, the honesty of the CRS generation has not yet been addressed. One can argue that in order to verify if the prover's proof is valid or not, the designated verifier has to generate the **crs** honestly. But then, if the designated verifier's sole purpose is to deny any proof provided by the prover, then the above assumption doesn't provide a sound protocol. Therefore, all information provided by the designated verifier has to be checked/validated by the prover or public verifier before the proof generation. The **crs** is generated as the encryption of each column of the following matrix (see Appendix A)

$$\boldsymbol{Q} = \begin{bmatrix} \hat{t} & 0 & 0 & \hat{V}_{n+1} & \cdots & \hat{V}_{N_\omega} & 0 & 0 & \cdots & 0 \\ 0 & \hat{t} & 0 & \hat{W}_{n+1} & \cdots & \hat{W}_{N_\omega} & 0 & 0 & \cdots & 0 \\ 0 & 0 & \hat{t} & \hat{Y}_{n+1} & \cdots & \hat{Y}_{N_\omega} & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & \hat{1} & \hat{\tau} & \cdots & \hat{\tau}^{N_g} \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \end{bmatrix}^\top$$

Let the encryptions of i-th row $\hat{\boldsymbol{q}}_i$ be

$$\left(\boldsymbol{a}, \boldsymbol{S}^\top \boldsymbol{a} + p \cdot \boldsymbol{e} + \hat{\boldsymbol{q}}_i\right)$$

for some $\boldsymbol{a} \xleftarrow{\$} \mathcal{R}_q^n$, $\boldsymbol{S} \leftarrow \chi^{n \times \ell'}$, $\boldsymbol{u} \leftarrow \chi^{\ell'}$ and $\boldsymbol{u}_i = [\hat{\boldsymbol{q}}_i^\top \mid (\boldsymbol{T}\hat{\boldsymbol{q}}_i)^\top]^\top$ Then, we can extract the following:

$$\begin{array}{l} \langle \boldsymbol{a}_1, \boldsymbol{s}_1 \rangle + p \cdot e_1 + \hat{t}, \\ \langle \boldsymbol{a}_2, \boldsymbol{s}_2 \rangle + p \cdot e_2 + \hat{t}, \\ \langle \boldsymbol{a}_3, \boldsymbol{s}_3 \rangle + p \cdot e_3 + \hat{t}, \\ \left\{ \langle \boldsymbol{a}_i'', \boldsymbol{s}_4 \rangle + p \cdot e_i'' + \hat{\tau}^i \right\}_{i \in \{0,1,\cdots,N_g\}} \end{array} \quad \left\{ \begin{array}{l} \langle \boldsymbol{a}_i', \boldsymbol{s}_1' \rangle + p \cdot e_{i,1}' + \hat{V}_i, \\ \langle \boldsymbol{a}_i', \boldsymbol{s}_2' \rangle + p \cdot e_{i,2}' + \hat{W}_i, \\ \langle \boldsymbol{a}_i', \boldsymbol{s}_3' \rangle + p \cdot e_{i,3}' + \hat{Y}_i \end{array} \right\}_{i \in \{n+1,\cdots,N_\omega\}} \tag{14}$$

Observe that given encryptions of $\hat{\tau}^i$ for $i \in \{0,1,\cdots,N_g\}$, one can compute encryption of any $v \in \mathcal{R}_p$, where v is a linear combination of $\{\hat{\tau}^j\}_{j \in \{0,1,\cdots,N_g\}}$. Since $\hat{t}$ and $\{\hat{V}_j, \hat{W}_j, \hat{Y}_j\}_{j \in \{n+1,\cdots,N_\omega\}}$ are linear combinations of $\{\hat{\tau}^j\}_{j \in \{0,1,\cdots,N_g\}}$, we can compute valid encryptions of the vectors under the secret $\boldsymbol{s}_4$. Using the protocol Π_{Lin} in Fig. 3, the designated verifier can provide proof that the underlying vectors of the computed encryptions under $\boldsymbol{s}_4$ are the same as the vectors that were committed for **crs** generation. Therefore, the designated verifier can provide proof for the consistency of **crs** as long as the encodings of $\{\hat{\tau}^j\}_{j \in [N_g]}$ are assumed to be generated honestly.

For any $j \in [N_g]$, consider the triplet $(\hat{\tau}, \hat{\tau}^j, \hat{\tau}^{j+1})$. Also, we know that $\hat{\tau} \cdot \hat{\tau}^j = \hat{\tau}^{j+1} \pmod p$ for all $j \in [N_g]$. Therefore, for each $j \in [N_g]$, we can find a quadratic relation, and we can use the protocol $\Pi_{\text{Quad}}^{(1)}$ in Fig. 4 to provide a zero-knowledge proof of such a relation. Thus, once the designated verifier generates the **crs**, it can also provide all the necessary proof to prove its consistency.

4.4 Detection of Dishonest Party

In Fig. 5, we have depicted the protocol $\Pi_{\text{Quad}}^{(2)}$. Here, the designated-verifier (Prover in the protocol) can encode the evaluation of the function as a committed message and send it to the public-verifier (Verifier in the protocol). The public-verifier can then check if this evaluation is non-zero or not and it also checks if the committed messages follow the actual equation or not. The check equations in $\Pi_{\text{Quad}}^{(2)}$ are as follows:

- $\|\boldsymbol{z}_i\|_2 \leq \sigma\sqrt{2nd}$ and $\boldsymbol{B}_i \boldsymbol{z}_i = \boldsymbol{\tau}_i + c \cdot \boldsymbol{t}_s^{(i)}$ for $1 \leq i \leq \kappa$ (15)
- $\boldsymbol{w}\boldsymbol{R}_2\boldsymbol{w} + c\boldsymbol{r}_1^\top \boldsymbol{w} + c^2 r_0 = f' \pmod p$ (16)
- $b = 0 \pmod p$ (17)

The first verification in Eq. (15) verifies if the prover (designated verifier in the case) possesses the secret key. Therefore, failing this equation would mean that the designated verifier generated an invalid **aux**. In the protocol $\Pi_{\text{Quad}}^{(2)}$, our cheat detection mechanism works by forcing the designated verifier to commit to the result of the verification equation itself. Specifically, we introduce a new variable, g', which is defined as

the value of the quadratic relation being tested ($g' = \boldsymbol{m}^\top \boldsymbol{R}_2 \boldsymbol{m} + \boldsymbol{r}_1^\top \boldsymbol{m} + r_0$). The designated verifier must commit to this value using a secret. The equation Eq. (16) checks if this committed value is consistent with the proof that was generated by the prover. The final step of the public verification is to simply check if this committed value was zero.

If an honest prover submitted a valid proof, then $g' = 0$. If the prover was dishonest, $g' \neq 0$, and the public verifier's check will fail. On the other hand, if the designated-verifier sends invalid information to the public-verifier to make it look like the proof provided by the prover is invalid *i.e.*, the designated verifier wants to prove that check equation in Eq. (17) is false. From Lemma 9, we can say that the designated-verifier can generate a transcript that passes check equation Eq. (16) but fails equation Eq. (17) with negligible probability, whenever the prover provides a valid proof.

Lemma 9. *The prover in $\Pi^{(2)}_{Quad}$ can generate a transcript that passes check equation Eq. (16) but fails equation Eq. (17) with negligible probability, whenever the commitments $\{\boldsymbol{t}_s^{(i)}, \boldsymbol{t}_m^{(i)}\}_{i\in[\kappa]}$ are honestly generated for a message $\boldsymbol{m} = (m_1, \cdots, m_\kappa)^\top$, where $\boldsymbol{m}^\top \boldsymbol{R}_2 \boldsymbol{m} + \boldsymbol{r}_1^\top \boldsymbol{m} + r_0 = 0 \pmod p$*

Proof. Since check equation in Eq. (15) has to be passed in $\Pi^{(2)}_{\text{Quad}}$, we can assume that $\{\boldsymbol{\tau}_i\}_{i\in[\kappa]}$ are generated honestly. Now let us assume that the prover generates $\tilde{g}, \tilde{g}', \tilde{v}, \tilde{v}'$ in a way so that at verifier's side check equation Eq. (16) passes but equation Eq. (17) fails.

Now at verifier's side, it computes $\boldsymbol{w} = c\vec{\boldsymbol{t}}_m - \vec{\boldsymbol{v}} - p\boldsymbol{u}_h$, where $\vec{\boldsymbol{t}}_m = (\boldsymbol{t}_m^{(1)}, \cdots, \boldsymbol{t}_m^{(\kappa)})^\top$ and $\vec{\boldsymbol{v}} = (\boldsymbol{a}_1^\top \boldsymbol{z}_1, \cdots, \boldsymbol{a}_\kappa^\top \boldsymbol{z}_\kappa)^\top$. It also computes

$$\begin{aligned} f &= c\tilde{g} + \tilde{v} + \boldsymbol{b}^\top \boldsymbol{y}_1 \\ b &= c\tilde{g}' + \tilde{v}' + \boldsymbol{d}^\top \boldsymbol{y}_2 \\ f' &= \tilde{v} + \boldsymbol{b}^\top \boldsymbol{y}_1 + c(\tilde{g} - \tilde{v}' - \boldsymbol{d}^\top \boldsymbol{y}_2) - c^2 \tilde{g}' \end{aligned} \tag{18}$$

Observe that $\boldsymbol{w}$ can be simplified as $\boldsymbol{w} = c\boldsymbol{m} + \boldsymbol{u}_l$. Therefore, the check equation Eq. (16) can be simplified as

$$c(\boldsymbol{m}^\top \boldsymbol{R}_2 \boldsymbol{u}_l + \boldsymbol{u}_l{}^\top \boldsymbol{R}_2 \boldsymbol{m} + \boldsymbol{r}_1^\top \boldsymbol{u}_l) + \boldsymbol{u}_l{}^\top \boldsymbol{R}_2 \boldsymbol{u}_l = f' \pmod p \tag{19}$$

Where, $\boldsymbol{u}_l = \boldsymbol{u} \pmod p$ and $\boldsymbol{u}_h = \lfloor \boldsymbol{u}/p \rfloor$. We have ignored the term $\boldsymbol{m}^\top \boldsymbol{R}_2 \boldsymbol{m} + \boldsymbol{r}_1^\top \boldsymbol{m} + r_0$ as it is 0 in modulo p. From equations Eq. (18) and Eq. (19), we get

$$c^2 \tilde{g}' + c(\alpha + \tilde{v}' - \tilde{g}) + \beta - \tilde{v} = 0 \pmod p \tag{20}$$

where $\alpha = \boldsymbol{m}^\top \boldsymbol{R}_2 \boldsymbol{u}_l + \boldsymbol{u}_l{}^\top \boldsymbol{R}_2 \boldsymbol{m} + \boldsymbol{r}_1^\top \boldsymbol{u}_l + \boldsymbol{d}^\top \boldsymbol{y}_2$ and $\beta = \boldsymbol{u}_l{}^\top \boldsymbol{R}_2 \boldsymbol{u}_l - \boldsymbol{b}^\top \boldsymbol{y}_1$ and these α and β are known to the prover. Now, if the prover wants to make sure that at the verifier's side, check equation in Eq. (16) passes but equation in Eq. (17) fails, then it must have

$$c\tilde{g}' + \tilde{v}' + \boldsymbol{d}^\top \boldsymbol{y}_2 \neq 0 \pmod p \tag{21}$$

$$c^2 \tilde{g}' + c(\alpha + \tilde{v}' - \tilde{g}) + \beta - \tilde{v} = 0 \pmod p \tag{22}$$

Now, given $\tilde{g},\tilde{g}',\tilde{v},\tilde{v}'$, the verifier chooses a challenge $c \in \mathcal{C}$ and there are at most two possible challenges in $\mathcal{C}$ that satisfy both of the above conditions. Therefore, the prover in $\Pi^{(2)}_{\text{Quad}}$ generates a transcript that passes check equation in Eq. (16) but fails equation in Eq. (17) with probability at most $2/|\mathcal{C}|$. Therefore, for sufficiently large size of the challenge space $\mathcal{C}$, our claim is true.

4.5 Construction

Let $\mathcal{CS} = \{\mathcal{CS}_\kappa\}_{\kappa\in\mathbb{N}}$ be a family of R1CS systems over the field $\mathbb{F}_p$. The construction relies on the following building blocks:

- Let $\Pi_{\text{LPCP}} = (\mathcal{Q}_{\text{LPCP}},\mathcal{P}_{\text{LPCP}},\mathcal{V}_{\text{LPCP}})$ be a ℓ-query input-oblivious linear PCP for $\mathcal{CS}$. Also Let N_c be the query length.

- Let $\Pi_{\text{Enc}} = (\mathbf{Setup}_{\text{Enc}},\mathbf{Encrypt}_{\text{Enc}},\mathbf{Add}_{\text{Enc}},\mathbf{Decrypt}_{\text{Enc}})$ be a secret-key linear only vector encryption scheme over $\mathbb{F}_p^\ell$ as described in Section 2.2.

The HV-zkSNARK $\Pi_{\text{HV}} = (\mathbf{Setup},\mathbf{Prove},\mathbf{DVerify},\mathbf{PVerify})$ for $\mathcal{R}_{\mathcal{CS}}$ is as follows:

- $\mathbf{Setup}(1^\lambda,1^\kappa)$: On input of security parameter λ and index κ, the setup runs $(\boldsymbol{Q},\mathbf{st}_{\text{LPCP}}) \leftarrow \mathcal{Q}_{\text{LPCP}}(1^\kappa)$ where $\boldsymbol{Q} \in \mathbb{F}^{N_c\times\ell}$. For each $i \in [N_c]$, let $\boldsymbol{q}_i^\top \in \mathbb{F}^\ell$ denote the i-th row of $\boldsymbol{Q}$. Then sample $(\mathbf{pp},\mathbf{sk}) \leftarrow \mathbf{Setup}_{\text{Enc}}(1^\lambda,1^\kappa)$ and compute $\mathbf{ct}_i \leftarrow \mathbf{Encrypt}_{\text{Enc}}(\mathbf{sk},\boldsymbol{q}_i^\top)$ for $i \in [N_c]$. Output the common reference string as $\mathbf{crs} = (\kappa,\mathbf{pp},\{\mathbf{ct}_i\}_{i\in[N_c]})$ and $\mathbf{st} = (\mathbf{sk},\mathbf{st}_{\text{LPCP}})$ as verification key. Compute $\mathbf{ct}_{st} = \mathbf{Encrypt}_{\text{Enc}}(\mathbf{sk},\mathbf{st}_{\text{LPCP}})$ and we generate the commitments $\{\boldsymbol{t}_s^i\}_{i\in[\ell']}$ as shown in Eq. (12), and name it as $\mathbf{com}_{\mathbf{sk}} = \{\boldsymbol{t}_s^{(i)}\}_{i\in[\ell']}$. Also, we generate the commitments $\{\tilde{\boldsymbol{t}}_s^{(i)}\}_{i\in[\tau]}$ and ciphertext $\boldsymbol{C}$ as shown in Eq. (13), and name it as $\mathbf{com}_{\boldsymbol{T}}$ and finally we can generate all the encodings as shown in Eq. (14) and we name it as $\mathbf{com}_{\mathbf{crs}}$. It takes $\mathbf{com}_{\mathbf{crs}}$ and $\mathbf{com}_{\mathbf{sk}}$ and uses the protocol $\Pi^{(1)}_{\text{Quad}}$ to generate the proof $\pi_{\mathbf{crs}}$ as described in Section 4.3. Output the extended common reference string as $\widetilde{\mathbf{crs}} = (\mathbf{crs},\mathbf{com}_{\mathbf{sk}},\mathbf{com}_{\boldsymbol{T}},\mathbf{ct}_{st},\mathbf{com}_{\mathbf{crs}},\pi_{\mathbf{crs}})$ and public verification key $\mathbf{vk} = \{\mathbf{com}_{\mathbf{sk}},\mathbf{com}_{\boldsymbol{T}}\}$.

- $\mathbf{Prove}(\widetilde{\mathbf{crs}},\boldsymbol{x},\boldsymbol{\omega})$: On input of $\widetilde{\mathbf{crs}}$, the statement $\boldsymbol{x}$ and the witness $\boldsymbol{\omega}$, first the prover checks if the $\mathbf{crs}$ is generated honestly with the help of $\pi_{\mathbf{crs}}$. If $\mathbf{crs}$ is not honestly generated then prover aborts. Otherwise the prover constructs $\boldsymbol{\sigma} \in \mathbb{F}^{N_c}$ as shown in algorithm $\mathcal{P}(1^\kappa,\boldsymbol{x},\boldsymbol{\omega})$ in Section A. The prover computes $\mathbf{ct}^* \leftarrow \mathbf{Add}_{\text{Enc}}(\mathbf{pp},\{\mathbf{ct}_i\}_{i\in[N_c]},\{\sigma_i\}_{i\in[N_c]})$. It outputs the proof $\pi = \mathbf{ct}^*$.

- $\mathbf{DVerify}(\mathbf{st},\boldsymbol{x},\pi)$: On the input of the verification key $\mathbf{st} = (\mathbf{sk},\mathbf{st}_{\text{LPCP}})$, the statement $\boldsymbol{x}$ and the proof $\pi = \mathbf{ct}^*$, the designated-verifier first decrypts the encrypted proof and gets $\boldsymbol{a} \leftarrow \mathbf{Decrypt}_{\text{Enc}}(\mathbf{sk},\mathbf{ct}^*)$ and computes the decision value $b \leftarrow \mathcal{V}(\mathbf{st}_{\text{LPCP}},\boldsymbol{x},\boldsymbol{a})$. It also produces the following information:
  (i) It takes $\mathbf{com}_{\boldsymbol{T}}$, $\mathbf{ct}^*$ and $\mathbf{ct}_{st}$ first computes $\mathbf{ct}_{\hat{r}} = \mathbf{ct}^* + \mathbf{ct}_{st}$. It generates the proof $\pi_{\boldsymbol{T}}$ that proves that $\boldsymbol{T}\boldsymbol{v}_1 = \boldsymbol{v}_2$ where $\mathbf{ct}_{\hat{r}}$ is an encryption of $[\boldsymbol{v}_1^\top \mid \boldsymbol{v}_2^\top]$ as it is explained in Section 4.2.

(ii) It takes $\mathbf{com_{sk}}$ and $\mathbf{ct}_{\hat{r}}$ and uses the method described in Section 4.1 to provide a proof that the underlying plaintext corresponding to the ciphertext $\mathbf{ct}_{\hat{r}}$ satisfies a quadratic relation. Let π_{Quad} be the genereted proof.

The it generates $\mathbf{aux}=(\mathbf{ct}_{st},\pi_{\boldsymbol{T}},\pi_{\text{Quad}})$ and outputs $(b,\mathbf{aux})$.

– **PVerify**$(\mathbf{aux},\mathbf{vk},\boldsymbol{x},\pi)$: On the input of $\mathbf{aux}=(\mathbf{ct}_{st},\pi_{\boldsymbol{T}},\pi_{\text{Quad}})$, public verification key $\mathbf{vk}=\{\mathbf{com_{sk}},\mathbf{com}_{\boldsymbol{T}}\}$, the statement $\boldsymbol{x}$ and the proof $\pi=\mathbf{ct}^*$, the public-verifier first computes $\mathbf{ct}_{\hat{r}}=\mathbf{ct}^*+\mathbf{ct}_{st}$ and check:
  (i) Check if $\pi_{\boldsymbol{T}}$ is a valid proof of encryption for $\mathbf{ct}_{\hat{r}}$.
  (ii) Check if π_{Quad} is a valid proof that the underlying plaintext corresponding to the ciphertext $\mathbf{ct}_{\hat{r}}$, satisfies the quadratic equation as described in Section 4.1.

Moreover, as we have discussed in Section 4.4, if the verification **PVerify** fails then the public-verifier can detect the party that has provided invalid information.

4.6 Knowledge Amplification for zkSNARK

In the work by Ishai et al. [24], it is shown that given any LPCP over the field $\mathbb{F}_{p^k}$, we can have an equivalent LPCP over the field $\mathbb{F}_p$, where p is a prime. Therefore, we are only considering LPCP Π_{LPCP} over the field $\mathbb{F}_p$. Now, the linear only encryption in Section 2.2 is over the ring $\mathcal{R}_p$ but the LPCP Π_{LPCP} is defined over the field $\mathbb{F}_p$, with knowledge error $2N_g/(p-N_g)$. To have negligible knowledge error, we need larger fields or we can use repetition. In [24], they have modified a ℓ-query LPCP with knowledge error ε into a $d\ell$-query LPCP with knowledge error ε^d using parallel repetition. In our work, we use a different approach to batch multiple LPCP queries into one.

The prime p we have chosen, satisfies the property $p\equiv 2d+1 \pmod{4d}$, where d is a power of 2 and $1<d\leq N$. Then from Lemma 3, we can say that X^N+1 splits into d irreducible polynomials. In particular $p\equiv 2d+1 \pmod{4d}$ ensures that a primitive d-th root of unity say ζ is in $\mathbb{Z}_p$. In fact, $\{\zeta^{2i+1} \mid i\in\mathbb{Z}_N\}$ are all the primitive $2d$-th roots of unity in $\mathbb{Z}_p$. Moreover, the polynomial X^N+1 factors into d irreducible polynomials modulo p as below

$$X^N+1=\prod_{i=0}^{d-1}\left(X^{N/d}-\zeta^{2i+1}\right) \pmod{p}.$$

Therefore, we can have the following isomorphism

$$\Psi:\mathcal{R}_p\rightarrow\frac{\mathbb{Z}_p[X]}{\langle X^{N/d}-\zeta\rangle}\times\frac{\mathbb{Z}_p[X]}{\langle X^{N/d}-\zeta^3\rangle}\times\cdots\times\frac{\mathbb{Z}_p[X]}{\langle X^{N/d}-\zeta^{2d-1}\rangle}$$

From now on we will denote the ring $\mathbb{Z}_p[X]/\langle X^{N/d}-\zeta^{2i+1}\rangle$ with the notation $\mathcal{R}_p^{(i)}$ for all $i\in\mathbb{Z}_d$.

Now, we will describe an encoding method that encodes an element $\boldsymbol{a}\in\mathbb{Z}_p^d$ into an element in $\mathcal{R}_p$. This encoding method will be useful for this work. First, we consider the map

$$F:\mathbb{Z}_p^d\rightarrow\mathcal{R}_p^{(0)}\times\mathcal{R}_p^{(1)}\times\cdots\times\mathcal{R}_p^{(d-1)}$$

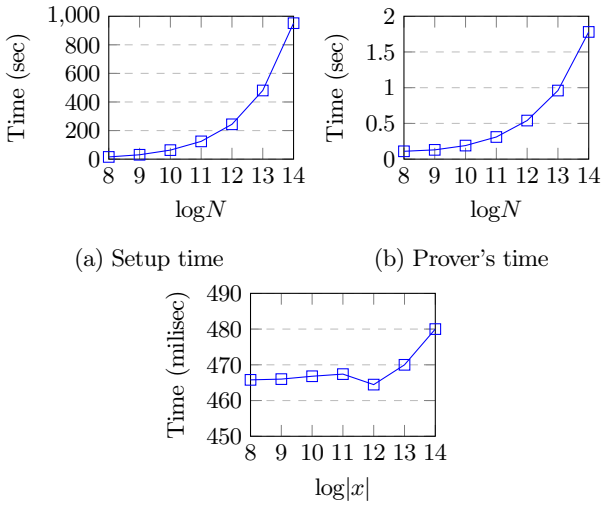
This is the natural mapping from $\mathbb{Z}_p^d$ to $\prod_{i\in\mathbb{Z}_d}\mathcal{R}_p^{(i)}$. We define the encoding $\hat{a}\in\mathcal{R}_p$ of an element $\boldsymbol{a}\in\mathbb{Z}_p^d$ as $\hat{a}=\Psi^{-1}(F(\boldsymbol{a}))$. This encoding is also an isomorphism.

Therefore, given d many LPCP over $\mathbb{Z}_p$, we can use the above isomorphism from $\mathbb{Z}_p^d$ to $\mathcal{R}_p$ and define only one LPCP over $\mathcal{R}_p$ instead of parallel repetition of LPCP. In this case, also the overall knowledge error is ε^d, if ε is the knowledge error for the original LPCP.

5 Implemenation Results

In this section, we discuss the implementation details of our proposed scheme, and we also compare the results with other existing zkSNARK solutions. First, we describe the lattice parameters that we have selected for our implementation.

- The plaintext modulus p is chosen as a 39 bit prime integer that satisfies $p\equiv 2d+1 \pmod{4d}$, where d is chosen as 2^3. As we have mentioned in Section 4.6, we can batch 2^3 QAP queries for knowledge amplification. The linear PCP that we are using has the knowledge error $\varepsilon=2N_g/(p-N_g)$ where N_g is the number of constraints in the R1CS system. Therefore, the overall knowledge error is ε^d which is less than 2^{-128} for our chosen p and d.
- We are working over the ring $\mathcal{R}=\mathbb{Z}[x]/\langle X^N+1\rangle$ where N is chosen as 2^7.
- We have chosen the plaintext dimension as $\ell=5$.
- The smudging bound B is chosen so that $\epsilon<2^{-128}$ in Eq. (4) in Theorem 3.



(a) Setup time

(b) Prover's time

(c) Verifier's time

Fig. 6: Prover's execution and setup time are plotted against log of constraint size N. The verifier's execution time is plotted against the log of the size of the input statement x.

- The parameter τ is chosen so that we have $p^{N\tau} > 2^{128}$. In fact, for $\tau = 1$ we can have this condition satisfied for our chosen p and N.
- The modulus q is also chosen as $q = q_1 \cdot q_2 \cdot q_3 \cdot q_4$ and for all $1 \leq i \leq 4$ these q_i's satisfy the condition $q_i \equiv 2d+1 \pmod{4d}$, where $d = 2^3$. Also, we have considered Theorem 1 to select a suitable bound for q for the correctness of the protocol. In our case, we have taken $q < 2^{250}$. For $1 \leq i \leq 4$, each q_i is chosen to be less than 2^{64}, so that we use the Chinese remainder theorem and utilise the 64-bit architecture.
- For the challenge space $\mathcal{C}$, we have chosen the parameter $\nu = 31$ so that $|\mathcal{C}| > 2^{128}$.
- The module dimension n is chosen to be 8 and we have used the LWE estimator by Albrecht et al. [2] to estimate the module dimension with 128 bit security. Also, the standard deviations for the discrete Gaussian distribution to sample the secret vector and errors from $\mathcal{R}_q$ are chosen as $\sigma_s = 128$ and $\sigma_e = 256$, respectively.

We have benchmarked the implementation of our HV-zkSNARK construction on a laptop running Ubuntu 22.04.5 LTS. The machine has 24×13th Gen Intel Core i9-13900HX at 3.9GHz and 16 GB of RAM. We compile our code using `gcc 11.4.0` for a 64-bit x86 architecture. All of our measurements are taken with a single-threaded execution. We have benchmarked the prover's execution time and plotted it against the logarithm of the constraint size in Fig. 6b. For constraint size from 2^8 to 2^{14} the prover takes less than 2 seconds to generate the proof. Also, the verifier (both designated and public) can verify within 1 second. The most expensive part is the setup. We have given the setup time plotted against the logarithm of the constraint size in Fig. 6a. The most time consuming operation in our construction is polynomial multiplication. Since the primes we have chosen are not number theoretic transformation (NTT) friendly, we had to rely on the standard schoolbook multiplication for polynomial multiplication, which hampers the performance of the setup operation.

6 Conclusion

In this work, we have introduced hybrid verification, which is a new notion of verification for zero-knowledge proof of knowledge. We have also given a construction of HV-zkSNARK based on module lattices. For our chosen parameter, we ensure 128-bit security, our proof size is $\approx$54.7 KB and the auxiliary information has a size of $\approx$2.1 KB, which is required for the public verifier to verify.

To make all the elements in the challenge space described in Section 2.6, we need primes that are not NTT-friendly in our case. This hampers the performance of the setup operation. Therefore, a more efficient approach for the implementation of polynomial multiplication modulo NTT unfriendly primes is a potential direction for improving the setup time.

The CRS size is another concern that increases as the size of the constraint increases. Ishai et al. [24] incorporated the modulus switching technique, which was introduced in terms of fully homomorphic encryption [15]. This modulus switching technique not only reduces the CRS size, but it also reduces the proof size. This technique can also be applied here in our construction to reduce the CRS size and also the proof size.

References

1. Albrecht, M.R., Cini, V., Lai, R.W.F., Malavolta, G., Thyagarajan, S.A.: Lattice-based snarks: Publicly verifiable, preprocessing, and recursively composable. In: Dodis, Y., Shrimpton, T. (eds.) Advances in Cryptology – CRYPTO 2022. pp. 102–132. Springer Nature Switzerland, Cham (2022)
2. Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. Journal of Mathematical Cryptology **9**(3), 169–203 (2015). https://doi.org/doi:10.1515/jmc-2015-0016, https://doi.org/10.1515/jmc-2015-0016
3. Ames, S., Hazay, C., Ishai, Y., Venkitasubramaniam, M.: Ligero: Lightweight sublinear arguments without a trusted setup. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. p. 2087–2104. CCS '17, Association for Computing Machinery, New York, NY, USA (2017). https://doi.org/10.1145/3133956.3134104, https://doi.org/10.1145/3133956.3134104
4. Attema, T., Cramer, R., Kohl, L.: A Compressed Σ-Protocol Theory for Lattices. In: Malkin, T., Peikert, C. (eds.) Advances in Cryptology – CRYPTO 2021. pp. 549–579. Springer International Publishing, Cham (2021)
5. Attema, T., Lyubashevsky, V., Seiler, G.: Practical product proofs for lattice commitments. In: Micciancio, D., Ristenpart, T. (eds.) Advances in Cryptology – CRYPTO 2020. pp. 470–499. Springer International Publishing, Cham (2020)
6. Bahrke, J., Reginer, T.: Towards the next technological transition: Commission presents eu strategy to lead on web 4.0 and virtual worlds (Jul 2023), https://ec.europa.eu/commission/presscorner/detail/en/ip_23_3718
7. Baum, C., Bootle, J., Cerulli, A., del Pino, R., Groth, J., Lyubashevsky, V.: Sub-linear lattice-based zero-knowledge arguments for arithmetic circuits. In: Shacham, H., Boldyreva, A. (eds.) Advances in Cryptology – CRYPTO 2018. pp. 669–699. Springer International Publishing, Cham (2018)
8. Baum, C., Damgård, I., Lyubashevsky, V., Oechsner, S., Peikert, C.: More Efficient Commitments from Structured Lattice Assumptions. In: Catalano, D., De Prisco, R. (eds.) Security and Cryptography for Networks. pp. 368–385. Springer International Publishing, Cham (2018)
9. Baum, C., Jadoul, R., Orsini, E., Scholl, P., Smart, N.P.: Feta: Efficient threshold designated-verifier zero-knowledge proofs. In: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. p. 293–306. CCS '22, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3548606.3559354, https://doi.org/10.1145/3548606.3559354
10. Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Paper 2018/046 (2018), https://eprint.iacr.org/2018/046
11. Ben-Sasson, E., Chiesa, A., Riabzev, M., Spooner, N., Virza, M., Ward, N.P.: Aurora: Transparent succinct arguments for r1cs. In: Ishai, Y., Rijmen, V. (eds.) Advances in Cryptology – EUROCRYPT 2019. pp. 103–128. Springer International Publishing, Cham (2019)
12. Beullens, W., Seiler, G.: Labrador: Compact proofs for r1cs from module-sis. In: Handschuh, H., Lysyanskaya, A. (eds.) Advances in Cryptology – CRYPTO 2023. pp. 518–548. Springer Nature Switzerland, Cham (2023)
13. Bootle, J., Lyubashevsky, V., Nguyen, N.K., Seiler, G.: A non-pcp approach to succinct quantum-safe zero-knowledge. In: Micciancio, D., Ristenpart, T. (eds.) Advances in Cryptology – CRYPTO 2020. pp. 441–469. Springer International Publishing, Cham (2020)

14. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. In: Proceedings of the 3rd Innovations in Theoretical Computer Science Conference. p. 309–325. ITCS '12, Association for Computing Machinery, New York, NY, USA (2012). https://doi.org/10.1145/2090236.2090262, https://doi.org/10.1145/2090236.2090262
15. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) lwe. In: Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science. p. 97–106. FOCS '11, IEEE Computer Society, USA (2011). https://doi.org/10.1109/FOCS.2011.12, https://doi.org/10.1109/FOCS.2011.12
16. Chiesa, A., Hu, Y., Maller, M., Mishra, P., Vesely, N., Ward, N.: Marlin: Preprocessing zksnarks with universal and updatable srs. In: Canteaut, A., Ishai, Y. (eds.) Advances in Cryptology – EUROCRYPT 2020. pp. 738–768. Springer International Publishing, Cham (2020)
17. Chiesa, A., Ojha, D., Spooner, N.: Fractal: Post-quantum and transparent recursive proofs from holography. In: Canteaut, A., Ishai, Y. (eds.) Advances in Cryptology – EUROCRYPT 2020. pp. 769–793. Springer International Publishing, Cham (2020)
18. Gennaro, R., Gentry, C., Parno, B., Raykova, M.: Quadratic Span Programs and Succinct NIZKs without PCPs. In: Johansson, T., Nguyen, P.Q. (eds.) Advances in Cryptology – EUROCRYPT 2013. pp. 626–645. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
19. Gennaro, R., Minelli, M., Nitulescu, A., Orrù, M.: Lattice-based zk-snarks from square span programs. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. p. 556–573. CCS '18, Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3243734.3243845, https://doi.org/10.1145/3243734.3243845
20. Gentry, C., Wichs, D.: Separating succinct non-interactive arguments from all falsifiable assumptions. In: Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing. p. 99–108. STOC '11, Association for Computing Machinery, New York, NY, USA (2011). https://doi.org/10.1145/1993636.1993651, https://doi.org/10.1145/1993636.1993651
21. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems, p. 203–225. Association for Computing Machinery, New York, NY, USA (2019), https://doi.org/10.1145/3335741.3335750
22. Golovnev, A., Lee, J., Setty, S., Thaler, J., Wahby, R.S.: Brakedown: Linear-time and field-agnostic snarks for r1cs. In: Handschuh, H., Lysyanskaya, A. (eds.) Advances in Cryptology – CRYPTO 2023. pp. 193–226. Springer Nature Switzerland, Cham (2023)
23. Groth, J.: On the size of pairing-based non-interactive arguments. In: Fischlin, M., Coron, J.S. (eds.) Advances in Cryptology – EUROCRYPT 2016. pp. 305–326. Springer Berlin Heidelberg, Berlin, Heidelberg (2016)
24. Ishai, Y., Su, H., Wu, D.J.: Shorter and Faster Post-Quantum Designated-Verifier zkSNARKs from Lattices. In: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. p. 212–234. CCS '21, Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3460120.3484572, https://doi.org/10.1145/3460120.3484572
25. Kilian, J.: A note on efficient zero-knowledge proofs and arguments (extended abstract). In: Proceedings of the Twenty-Fourth Annual ACM Symposium on Theory of Computing. p. 723–732. STOC '92, Association for Computing Machinery, New York, NY, USA (1992). https://doi.org/10.1145/129712.129782, https://doi.org/10.1145/129712.129782
26. Langlois, A., Stehlé, D.: Worst-case to average-case reductions for module lattices. Designs, Codes and Cryptography **75**(3), 565–599 (2015). https://doi.org/10.1007/s10623-014-9938-4, https://doi.org/10.1007/s10623-014-9938-4

27. Lyubashevsky, V., Nguyen, N.K., Plançon, M.: Lattice-Based Zero-Knowledge Proofs and Applications: Shorter, Simpler, and More General. In: Dodis, Y., Shrimpton, T. (eds.) Advances in Cryptology – CRYPTO 2022. pp. 71–101. Springer Nature Switzerland, Cham (2022)
28. Lyubashevsky, V., Nguyen, N.K., Seiler, G.: Practical lattice-based zero-knowledge proofs for integer relations. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. p. 1051–1070. CCS '20, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3372297.3417894, https://doi.org/10.1145/3372297.3417894
29. Lyubashevsky, V., Nguyen, N.K., Seiler, G.: Shorter Lattice-Based Zero-Knowledge Proofs via One-Time Commitments. In: Garay, J.A. (ed.) Public-Key Cryptography – PKC 2021. pp. 215–241. Springer International Publishing, Cham (2021)
30. Lyubashevsky, V., Seiler, G.: Short, Invertible Elements in Partially Splitting Cyclotomic Rings and Applications to Lattice-Based Zero-Knowledge Proofs. In: Nielsen, J.B., Rijmen, V. (eds.) Advances in Cryptology – EUROCRYPT 2018. pp. 204–224. Springer International Publishing, Cham (2018)
31. Maller, M., Bowe, S., Kohlweiss, M., Meiklejohn, S.: Sonic: Zero-knowledge snarks from linear-size universal and updatable structured reference strings. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. p. 2111–2128. CCS '19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3319535.3339817, https://doi.org/10.1145/3319535.3339817
32. Micali, S.: Computationally sound proofs. SIAM J. Comput. **30**(4), 1253–1298 (Oct 2000). https://doi.org/10.1137/S0097539795284959, https://doi.org/10.1137/S0097539795284959
33. Setty, S.: Spartan: Efficient and general-purpose zksnarks without trusted setup. In: Advances in Cryptology – CRYPTO 2020: 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17–21, 2020, Proceedings, Part III. p. 704–737. Springer-Verlag, Berlin, Heidelberg (2020). https://doi.org/10.1007/978-3-030-56877-1_25
34. Zhou, Z., Li, Z., Zhang, X., Sun, Y., Xu, H.: A review of gaps between web 4.0 and web 3.0 intelligent network infrastructure (2023), https://arxiv.org/abs/2308.02996

Appendix A Linear PCP for R1CS

Let $\mathcal{CS}=\{\mathcal{CS}_\kappa\}_{\kappa\in\mathbb{N}}$ be a family of R1CS instances over a finite field $\mathbb{F}$, where $\mathcal{CS}_\kappa=\{n_{c,\kappa},N_{g,\kappa},N_{\omega,\kappa},\{\boldsymbol{a}_{i,\kappa},\boldsymbol{b}_{i,\kappa},\boldsymbol{c}_{i,\kappa}\}\}_{i\in[N_{g,\kappa}]}$, $\boldsymbol{a}_{i,\kappa},\boldsymbol{b}_{i,\kappa},\boldsymbol{c}_{i,\kappa}\in\mathbb{F}^{N_{\omega,\kappa}+1}$ (and entries indexed from 0 to $N_{\omega,\kappa}$). For notational convenience, we write $n=n(\kappa)$ to denote a function where $n(\kappa)=n_{c,\kappa}$ for all $\kappa\in\mathbb{N}$. We define $N_g=N_g(\kappa)$, $N_\omega=N_\omega(\kappa)$, $\boldsymbol{a}_i=\boldsymbol{a}_i(\kappa)$,$\boldsymbol{b}_i=\boldsymbol{b}_i(\kappa)$ and $\boldsymbol{c}_i=\boldsymbol{c}_i(\kappa)$ similarly. We additionally define the following components:

- Let $S=\{\alpha_1,\alpha_2,\cdots,\alpha_{N_g}\}$ be an arbitrary subset of $\mathbb{F}$.
- For $i\in\{0,\cdots,N_\omega\}$, Let $V_i,W_i,Y_i:\mathbb{F}\to\mathbb{F}$ be unique polynomial of degree N_g-1, where for all $j\in[N_g]$

$$\boldsymbol{a}_j=(V_i(\alpha_j))_{i=0}^{N_\omega},\ \ \boldsymbol{b}_j=(W_i(\alpha_j))_{i=0}^{N_\omega},\ \ \boldsymbol{c}_j=(Y_i(\alpha_j))_{i=0}^{N_\omega}$$

- Let $t:\mathbb{F}\to\mathbb{F}$ be the polynomial $t(z)=\prod_{i\in[N_g]}(z-\alpha_i)$.

Now the 5-query LPCP for $\mathcal{CS}$ denoted by Π_{LPCP} is defined as below.

- $\mathcal{Q}_{\mathrm{LPCP}}(1^\kappa)$: On input of $\kappa \in \mathbb{N}$, sample $\tau \xleftarrow{\$} \mathbb{F} \setminus S$. Let $\boldsymbol{v} = (V_1(\tau),\cdots,V_n(\tau))$, $\boldsymbol{w}=(W_1(\tau),\cdots,W_n(\tau))$ and $\boldsymbol{y}=(Y_1(\tau),\cdots,Y_n(\tau))$. Output the verification key $\mathbf{st}$ as $\boldsymbol{g}=\big(V_0(\tau)+\boldsymbol{x}^\top\boldsymbol{v},W_0(\tau)+\boldsymbol{x}^\top\boldsymbol{w},Y_0(\tau)+\boldsymbol{x}^\top\boldsymbol{y},0,t(\tau)\big)^\top$ and the query matrix Q as

$$\begin{bmatrix} t(\tau) & 0 & 0 & V_{n+1}(\tau) & \cdots & V_{N_\omega}(\tau) & 0\,0\cdots & 0 \\ 0 & t(\tau) & 0 & W_{n+1}(\tau) & \cdots & W_{N_\omega}(\tau) & 0\,0\cdots & 0 \\ 0 & 0 & t(\tau) & Y_{n+1}(\tau) & \cdots & Y_{N_\omega}(\tau) & 0\,0\cdots & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1\,\tau\cdots & \tau^{N_g} \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0\,0\cdots & 0 \end{bmatrix}^\top$$

- $\mathcal{P}_{\mathrm{LPCP}}(1^\kappa,\boldsymbol{x},\boldsymbol{\omega})$: On input of $\kappa \in \mathbb{N}$ and $(\boldsymbol{x},\boldsymbol{\omega})$ where $\mathcal{CS}(\boldsymbol{x},\boldsymbol{\omega})=1$, sample $\delta_1,\delta_2,\delta_3 \xleftarrow{\$} \mathbb{F}$. Construct the polynomials $V,W,Y:\mathbb{F}\to\mathbb{F}$, each of degree N_g, where

$$V(z)=\delta_1 t(z)+V_0(z)+\sum_{i\in[N_\omega]}\omega_i V_i(z)$$
$$W(z)=\delta_2 t(z)+W_0(z)+\sum_{i\in[N_\omega]}\omega_i W_i(z)$$
$$Y(z)=\delta_3 t(z)+Y_0(z)+\sum_{i\in[N_\omega]}\omega_i Y_i(z)$$

  Let $H(z)=(V(z)\cdot W(z)-Y(z))/t(z)$, and let $\boldsymbol{h}\in\mathbb{F}^{N_g+1}$ be the vector containing the coefficients of H. Now note that first n elements of $\boldsymbol{\omega}$ is exactly $\boldsymbol{x}$. We parse $\boldsymbol{\omega}$ as $\boldsymbol{\omega}^\top=[\boldsymbol{x}^\top|\tilde{\boldsymbol{\omega}}^\top]$ and generate the vector $\boldsymbol{\sigma}=(\delta_1,\delta_2,\delta_3,\tilde{\boldsymbol{\omega}},\boldsymbol{h})\in\mathbb{F}^{4+N_\omega+N_g-n}$. Output the proof vector $\boldsymbol{\pi}=\boldsymbol{Q}^\top\boldsymbol{\sigma}$.

- $\mathcal{V}(\mathbf{st},\boldsymbol{x},\boldsymbol{\pi})$: On input of verification key $\mathbf{st}=\boldsymbol{g}\in\mathbb{F}^5$, the proof $\boldsymbol{\pi}\in\mathbb{F}^5$ the verifier computes $\boldsymbol{r}=\boldsymbol{\pi}+\boldsymbol{g}$ and accepts if

$$r_1 r_2 - r_3 - r_4 r_5 = 0$$

The above-described LPCP is an honest verifier zero-knowledge and has a knowledge error $2N_g/(|\mathbb{F}|-N_g)$. For detailed analysis, we refer to [24, Appendix A-B].

Appendix B Proof of Lemma 4

Proof. Let $(\boldsymbol{\tau},u,c_1,\boldsymbol{z}_1,\boldsymbol{z}_1')$ and $(\boldsymbol{\tau},u,c_2,\boldsymbol{z}_2,\boldsymbol{z}_2')$ be two accepting transcripts which is obtained by rewinding the prover who sends $\boldsymbol{\tau},\boldsymbol{\tau}',u$ in the first step. As the transcripts are accepted, then they satisfy the second and third verification equations, and by subtracting the equalities, we obtain

$$\boldsymbol{B}\bar{\boldsymbol{z}}+\boldsymbol{B}'\bar{\boldsymbol{z}}'=\bar{c}\boldsymbol{t}_s \tag{23}$$

where $\bar{\boldsymbol{z}} = \boldsymbol{z}_1 - \boldsymbol{z}_2$, $\bar{\boldsymbol{z}}' = \boldsymbol{z}'_1 - \boldsymbol{z}'_2$ and $\bar{c} = c_1 - c_2$. Dividing the equality in Eq. (23) by $\bar{c}$, we get the equality in statement 1, where $\bar{\boldsymbol{s}} = \bar{\boldsymbol{z}}/\bar{c}$ and $\bar{\boldsymbol{s}}' = \bar{\boldsymbol{z}}'/\bar{c}$. Since the first verification checks $\|\boldsymbol{z}\|_2 \leq \sigma\sqrt{2nd}$ and $\|\boldsymbol{z}'\|_2 \leq \sigma\sqrt{2nd}$, we know that $\|\bar{\boldsymbol{z}}\|_2 \leq 2\sigma\sqrt{2nd}$ and $\|\bar{\boldsymbol{z}}'\|_2 \leq 2\sigma\sqrt{2nd}$ and so the statement 2 is satisfied. By subtracting the two equalities satisfying the third verification equation, we obtain

$$h\boldsymbol{a}^\top \bar{\boldsymbol{z}} - \boldsymbol{a}^\top \bar{\boldsymbol{z}}' = \bar{c}(h\boldsymbol{t}_m - \boldsymbol{t}'_m) \pmod{p} \tag{24}$$

dividing the above equality by $\bar{c}$ we get the statement 3.

Now suppose that the extractor extracts another secret $(\bar{\boldsymbol{s}}_1, \bar{\boldsymbol{s}}'_1, \bar{c}_1)$ with $(\bar{\boldsymbol{s}}_1, \bar{\boldsymbol{s}}'_1) \neq (\bar{\boldsymbol{s}}, \bar{\boldsymbol{s}}')$, both of these secrets satisfies first two statements. We have the following

$$\boldsymbol{B}\bar{\boldsymbol{s}} + \boldsymbol{B}'\bar{\boldsymbol{s}}' = \boldsymbol{B}\bar{\boldsymbol{s}}_1 + \boldsymbol{B}'\bar{\boldsymbol{s}}'_1 \tag{25}$$

Multiplying equality in Eq. (25) by $\bar{c}\bar{c}_1$ we get

$$\boldsymbol{B}(\bar{\boldsymbol{s}} - \bar{\boldsymbol{s}}_1)\bar{c}\bar{c}_1 + \boldsymbol{B}'(\bar{\boldsymbol{s}}' - \bar{\boldsymbol{s}}'_1)\bar{c}\bar{c}_1 = \boldsymbol{0} \tag{26}$$

We can rewrite the above as

$$\boldsymbol{B}(\bar{\boldsymbol{z}}\bar{c}_1 - \bar{\boldsymbol{z}}_1\bar{c}) + \boldsymbol{B}'(\bar{\boldsymbol{z}}'\bar{c}_1 - \bar{\boldsymbol{z}}'_1\bar{c}) = \boldsymbol{0} \tag{27}$$

Here $\|\bar{\boldsymbol{z}}\|, \|\bar{\boldsymbol{z}}'\|, \|\bar{\boldsymbol{z}}_1\|, \|\bar{\boldsymbol{z}}'_1\| \leq 2\sigma\sqrt{2nd}$ and multiplication by any $\bar{c} \in \bar{\mathcal{C}}$ increases the norm by a factor of 2η, thus $\|\bar{\boldsymbol{z}}\bar{c}_1 - \bar{\boldsymbol{z}}_1\bar{c}\|, \|\bar{\boldsymbol{z}}'\bar{c}_1 - \bar{\boldsymbol{z}}'_1\bar{c}\| \leq 8\eta\sigma\sqrt{2nd}$. If $\mathtt{MSIS}_{k,2n,B}$ is hard for $B = 8\eta\sigma\sqrt{2nd}$, it implies that a PPT adversary can find a non-zero solution to the equation in Eq. (27) with negligible probability, i.e., statement 4 is true.

For statement 5, if possible we consider $\boldsymbol{z}_1 - c_1\bar{\boldsymbol{s}} = \boldsymbol{z}_2 - c_2\bar{\boldsymbol{s}} + \boldsymbol{r}$ and $\boldsymbol{z}'_1 - c_1\bar{\boldsymbol{s}}' = \boldsymbol{z}'_2 - c_2\bar{\boldsymbol{s}}' + \boldsymbol{r}'$ for some $\boldsymbol{r}, \boldsymbol{r}'$. We can rewrite these as $\bar{\boldsymbol{z}}/\bar{c} - \bar{\boldsymbol{s}} = \boldsymbol{r}/\bar{c}$ and $\bar{\boldsymbol{z}}'/\bar{c} - \bar{\boldsymbol{s}}' = \boldsymbol{r}'/\bar{c}$. Since we already know $\bar{\boldsymbol{z}}/\bar{c} = \bar{\boldsymbol{s}}$ and $\bar{\boldsymbol{z}}'/\bar{c} = \bar{\boldsymbol{s}}'$, it implies $\boldsymbol{r} = \boldsymbol{r}' = \boldsymbol{0}$.

Appendix C Proof of Lemma 5

Proof. We can simulate the commitments and the non-aborting transcripts between an honest verifier and an honest prover.

First, we consider a hybrid simulator $\mathcal{S}_0$, which have the knowledge of $\boldsymbol{s}$ and $\boldsymbol{m} = (m_1, m_2, \cdots, m_\kappa)^\top$. Given a challenge $c \in \mathcal{C}$, it honestly generates $\boldsymbol{t}_s, \{\boldsymbol{t}_m^{(i)}\}_{i\in[\kappa]}$ and h under the secret $\boldsymbol{s}$. Now it samples $\boldsymbol{z} \leftarrow D_\sigma^n$ conditioned on $\langle \boldsymbol{s}, \boldsymbol{z} \rangle \geq 0$.Finally it sets $\boldsymbol{\tau} = \boldsymbol{B}\boldsymbol{z} - c\boldsymbol{t}_s$ and $v = \boldsymbol{w}^\top \boldsymbol{R}_2 \boldsymbol{w} + c\boldsymbol{r}_1^\top \boldsymbol{w} + c^2 r_0 - ch + \boldsymbol{b}^\top \boldsymbol{z}$. Then by Lemma 2, the distribution of the commitment and transcript output by $\mathcal{S}_0$ is identical to the one in the actual non-aborting sequence.

Next, we consider the simulator $\mathcal{S}_1$, which still has knowledge of the secret $\boldsymbol{s}, \boldsymbol{m}$. It generates the commitments by first sampling a random $\boldsymbol{r} \xleftarrow{\$} \mathcal{R}_q^{k+\kappa+1}$ and computing

$$\left[\boldsymbol{t}_s^\top \mid \boldsymbol{t}_m^{(1)} \cdots \boldsymbol{t}_m^{(\kappa)} \; h\right]^\top = \boldsymbol{r} + \left[\boldsymbol{0} \mid m_1 \cdots m_\kappa \; g\right]^\top \tag{28}$$

Here the claim is that if there exist a PPT adversary $\mathcal{A}$ that distinguishes between the output of $\mathcal{S}_0$ and $\mathcal{S}_1$ with probability ε, then there exists a PPT adversary $\mathcal{B}$ that

solves Extended-$\mathtt{MLWE}_{k+\kappa+1,n-k-\kappa-1,\chi,\mathcal{C},\sigma}$ with probability at least $\varepsilon/2$. We define $\mathcal{B}$ as follows. Given an Extended-MLWE tuple $(\boldsymbol{C},\boldsymbol{r},\boldsymbol{z},b)$, where

$$\boldsymbol{C}=\left[\boldsymbol{B}^\top|\ \boldsymbol{a}_1^\top|\ \cdots|\ \boldsymbol{a}_\kappa^\top|\ \boldsymbol{b}^\top\right]^\top$$

$\mathcal{B}$ first sets the commitments as in Eq. (C) and simulates rest of the transcript as in $\mathcal{S}_0$ and $\mathcal{S}_1$. It outputs the commitments and the transcript to the PPT adversary $\mathcal{A}$. Let us assume $b=1$. Now if $\boldsymbol{r}=\boldsymbol{C}\boldsymbol{s}$, then output of $\mathcal{B}$ comes from distribution of $\mathcal{S}_0$. Similarly, if $\boldsymbol{r}$ is uniformly random, then the output of $\mathcal{B}$ comes from the distribution of $\mathcal{S}_1$. Hence, given that $b=1$, the adversary $\mathcal{B}$ can solve the Extended-$\mathtt{MLWE}_{k+\kappa+1,n-k-\kappa-1,\chi,\mathcal{C},\sigma}$ problem with probability at least ε. Since the probability that $b=1$ is at least $1/2$, the statement follows.

Appendix D Proof of Lemma 6

Proof. We use the strategy by Attema et al. [4]. We consider a binary matrix $H\in\{0,1\}^{R\times N}$ where the R rows represent the prover's randomness and N columns represent the verifier's randomness, i.e., different choices for the challenge c. For simplicity, we denote $H(r,c)$ to be the entry corresponding to the randomness r and the challenge $c\in\mathcal{C}$. We define an extractor $\mathcal{E}$ in the following manner:

1. $\mathcal{E}$ first samples a random r and challenge $c^{(0)}\leftarrow\mathcal{C}$. Then it checks if $H(r,c^{(0)})$ is 1. If not, $\mathcal{E}$ aborts.
2. Otherwise, $\mathcal{E}$ samples along row r without replacement until it finds two $c^{(1)}$, $c^{(2)}$ such that $H(r,c^{(i)})=1$ for $i=0,1,2$.

According to [4], $\mathcal{E}$ extracts three valid transcripts

$$\mathtt{tr}^{(i)}=(\boldsymbol{\tau},h,v,c^{(i)},\boldsymbol{z}^{(i)})\quad\text{for } i=0,1,2$$

with probability at least $\varepsilon-2/|\mathcal{C}|$.

First we focus on $\mathtt{tr}^{(0)}$ and $\mathtt{tr}^{(1)}$. Define

$$\bar{c}=c^{(0)}-c^{(1)}\quad\text{and}\quad\bar{\boldsymbol{s}}=\frac{\boldsymbol{z}^{(0)}-\boldsymbol{z}^{(1)}}{c^{(0)}-c^{(1)}}$$

By the definition of $\mathcal{C}$, $\|\bar{c}\|_\infty\le 2$, $\|\bar{c}\bar{\boldsymbol{s}}\|\le 2\sigma\sqrt{2nd}$. Also we have $\boldsymbol{B}\bar{\boldsymbol{s}}=\boldsymbol{t}_s$. We define the extracted message as $\bar{\boldsymbol{m}}=(\bar{m}_1,\bar{m}_2,\cdots,\bar{m}_\kappa)$ where $\bar{m}_i=\boldsymbol{t}_m^{(i)}-\langle\boldsymbol{a}_i,\bar{\boldsymbol{s}}\rangle$ and $\bar{g}=h-\boldsymbol{b}^\top\bar{\boldsymbol{s}}$. Now, let $\boldsymbol{y}_1=\boldsymbol{z}^{(0)}-c^{(0)}\bar{\boldsymbol{s}}=\boldsymbol{z}^{(1)}-c^{(1)}\bar{\boldsymbol{s}}$. Consider the third transcript $\mathtt{tr}^{(2)}$ and we define $\boldsymbol{y}_2=\boldsymbol{z}^{(2)}-c^{(2)}\bar{\boldsymbol{s}}$. Since $\mathtt{tr}^{(i)}$ are non-aborting transcripts, we have

$$\begin{aligned}\boldsymbol{B}(\boldsymbol{y}_1-\boldsymbol{y}_2)&=\boldsymbol{B}\bar{\boldsymbol{z}}'-\bar{c}'\boldsymbol{B}\bar{\boldsymbol{s}}\\&=\boldsymbol{B}\bar{\boldsymbol{z}}'-\bar{c}'\boldsymbol{t}_s\\&=0\end{aligned}$$

where $\bar{\boldsymbol{z}}'=\boldsymbol{z}^{(0)}-\boldsymbol{z}^{(2)}$ and $\bar{c}'=c^{(0)}-c^{(2)}$. Therefore, either we have $\boldsymbol{y}_1=\boldsymbol{y}_2$ or the extractor solves $\mathtt{MSIS}_{k,n,B}$ for $B=4\sigma\sqrt{2nd}$. We will assume the formal from now on.

Next we define $\bar{\boldsymbol{u}}=(\bar{u}_1,\bar{u}_2,\cdots,\bar{u}_\kappa)^\top$, where $\bar{u}_i=-\boldsymbol{a}_i^\top\boldsymbol{y}_2$ for all $i\in[\kappa]$. Then from the verification equation 3 of $\Pi^{(1)}_{\mathrm{Quad}}$ we have for $i=0,1,2$:

$$(\boldsymbol{w}^{(i)})^\top\boldsymbol{R}_2\boldsymbol{w}^{(i)}+c^{(i)}\boldsymbol{r}_1^\top\boldsymbol{w}^{(i)}+(c^{(i)})^2r_0=c^{(i)}h-\boldsymbol{b}^\top\boldsymbol{z}+v \pmod p$$

where

$$\boldsymbol{w}^{(i)}=\begin{bmatrix}c^{(i)}\boldsymbol{t}_m^{(1)}-\langle\boldsymbol{a}_1,\boldsymbol{z}^{(i)}\rangle\\ \vdots\\ c^{(i)}\boldsymbol{t}_m^{(\kappa)}-\langle\boldsymbol{a}_\kappa,\boldsymbol{z}^{(i)}\rangle\end{bmatrix}=c^{(i)}\bar{\boldsymbol{m}}+\bar{\boldsymbol{u}}$$

Therefore, we have

$$(c^{(i)})^2\big(\bar{\boldsymbol{m}}^\top\boldsymbol{R}_2\bar{\boldsymbol{m}}+\boldsymbol{r}_1^\top\bar{\boldsymbol{m}}+r_0\big)+c^{(i)}g_1'+g_0'=0 \pmod p \quad \text{for} \quad i=0,1,2$$

where

$$\begin{aligned}g_1'&=\bar{\boldsymbol{m}}^\top\boldsymbol{R}_2\bar{\boldsymbol{u}}+\bar{\boldsymbol{u}}^\top\boldsymbol{R}_2\bar{\boldsymbol{m}}+\boldsymbol{r}_1^\top\bar{\boldsymbol{u}}-g\\ g_0'&=\bar{\boldsymbol{u}}^\top\boldsymbol{R}_2\bar{\boldsymbol{u}}+\boldsymbol{b}^\top\boldsymbol{y}_2-v\end{aligned}$$

We can write three equations as follows:

$$\begin{bmatrix}1 & c^{(0)} & (c^{(0)})^2\\ 1 & c^{(1)} & (c^{(1)})^2\\ 1 & c^{(2)} & (c^{(2)})^2\end{bmatrix}\begin{bmatrix}g_0'\\ g_1'\\ \bar{\boldsymbol{m}}^\top\boldsymbol{R}_2\bar{\boldsymbol{m}}+\boldsymbol{r}_1^\top\bar{\boldsymbol{m}}+r_0\end{bmatrix}=\begin{bmatrix}0\\0\\0\end{bmatrix} \pmod p$$

Since difference of two challenges in $\{c^{(0)},c^{(1)},c^{(2)}\}$ is invertible in $\mathcal{R}_p$, we must have $\bar{\boldsymbol{m}}^\top\boldsymbol{R}_2\bar{\boldsymbol{m}}+\boldsymbol{r}_1^\top\bar{\boldsymbol{m}}+r_0=0 \pmod p$.

Appendix E Proof of Lemma 7

Proof. We can simulate the commitments and the non-aborting transcripts between an honest verifier and an honest prover.

First, we consider a hybrid simulator $\mathcal{S}_0$, which have the knowledge of $\{\boldsymbol{s}_i\}_{1\le i\le\kappa}$ and $\boldsymbol{m}=(m_1,m_2,\cdots,m_\kappa)^\top$. Given a challenge $c\in\mathcal{C}$, it honestly generates $\{\boldsymbol{t}_s^{(i)},\boldsymbol{t}_m^{(i)}\}_{1\le i\le\kappa}$ and h,h' under the secrets. Now it samples $\boldsymbol{z}_i\leftarrow D_\sigma^n$ conditioned on $\langle\boldsymbol{s}_i,\boldsymbol{z}_i\rangle\ge 0$ for $1\le i\le\kappa$. Finally it sets $\boldsymbol{\tau}_i=\boldsymbol{B}_i\boldsymbol{z}_i-c\cdot\boldsymbol{t}_s^{(i)}$ for $1\le i\le\kappa$, $b=\boldsymbol{m}^\top\boldsymbol{R}_2\boldsymbol{m}+\boldsymbol{r}_1^\top\boldsymbol{m}+r_0$, $v=\boldsymbol{w}^\top\boldsymbol{R}_2\boldsymbol{w}+c\boldsymbol{r}_1^\top\boldsymbol{w}+c^2r_0+cb-ch+\boldsymbol{b}^\top\boldsymbol{z}_1$ and $v'=b-ch'+d^\top\boldsymbol{z}_2$. Then by Lemma 2, the distribution of the commitment and transcript output by $\mathcal{S}_0$ is identical to the one in the actual non-aborting sequence.

Next, we consider the simulator $\mathcal{S}_1$, which still have knowledge of the secret $\{\boldsymbol{s}_i,m_i\}_{1\le i\le\kappa}$. It generates the commitment by first sampling random $\boldsymbol{\delta}_1\xleftarrow{\$}\mathcal{R}_q^{k+2}$ and computes

$$\begin{bmatrix}\boldsymbol{t}_s^{(1)}\\ \boldsymbol{t}_m^{(1)}\\ h\end{bmatrix}=\boldsymbol{\delta}_1+\begin{bmatrix}\boldsymbol{0}\\ m_1\\ g\end{bmatrix} \tag{29}$$

Rest of the commitments $\{\boldsymbol{t}_s^{(i)},\boldsymbol{t}_m^{(i)}\}_{2\leq i\leq\kappa}$ are generated honestly as per the protocol. Now, the claim is that if there exist a PPT adversary $\mathcal{A}_1$ that distinguishes between the output of $\mathcal{S}_0$ and $\mathcal{S}_1$ with advantage ε_1, then there exists a PPT adversary $\mathcal{B}_1$ that solves Extended-$\mathtt{MLWE}_{k+2,n-k-2,\chi,\mathcal{C},\sigma}$ instances with advantage at least $\frac{\varepsilon_1}{2}$. We define $\mathcal{B}_1$ as follows: given Extended-$\mathtt{MLWE}$ tuple $\{(\boldsymbol{C}_1,\boldsymbol{\delta}_1,\boldsymbol{z}_1,\mathrm{sgn}_1)\}$, where

$$\boldsymbol{C}_1=\begin{bmatrix}\boldsymbol{B}_1\\ \boldsymbol{a}_1\\ \boldsymbol{b}\end{bmatrix}$$

$\mathcal{B}_1$ first sets the commitments as in Eq. (29) and simulates rest of the transcript as in $\mathcal{S}_0$ and $\mathcal{S}_1$. It outputs the commitments and the transcript to the PPT adversary $\mathcal{A}_1$. Let us assume $\mathrm{sgn}_1=1$. Now if $\boldsymbol{\delta}_1=\boldsymbol{C}_1\boldsymbol{s}_1$, then output of $\mathcal{B}_1$ comes from distribution of $\mathcal{S}_0$. Similarly, if $\boldsymbol{\delta}_1$ is uniformly random, then the output of $\mathcal{B}_1$ comes from the distribution of $\mathcal{S}_1$. Hence, given that $\mathrm{sgn}_1=1$, the adversary $\mathcal{B}_1$ can solve Extended-$\mathtt{MLWE}$ instance with probability at least ε_1. Since the probability that $\mathrm{sgn}_1=1$ is at least $1/2$, $\mathcal{B}_1$ can solve Extended-$\mathtt{MLWE}$ with advantage at least $\frac{\varepsilon_1}{2}$.

Next, we consider the simulator $\mathcal{S}_2$, which still have knowledge of the secret $\{\boldsymbol{s}_i,m_i\}_{1\leq i\leq\kappa}$. It generates $\{\boldsymbol{t}_s^{(1)},\boldsymbol{t}_m^{(1)},h\}$ similar to simulator $\mathcal{S}_1$ and then samples random $\boldsymbol{\delta}_2\xleftarrow{\$}\mathcal{R}_q^{k+2}$ and computes

$$\begin{bmatrix}\boldsymbol{t}_s^{(2)}\\ \boldsymbol{t}_m^{(2)}\\ h'\end{bmatrix}=\boldsymbol{\delta}_2+\begin{bmatrix}\boldsymbol{0}\\ m_2\\ g'\end{bmatrix} \tag{30}$$

Rest of the commitments $\{\boldsymbol{t}_s^{(i)},\boldsymbol{t}_m^{(i)}\}_{3\leq i\leq\kappa}$ are generated honestly as per the protocol. Here also we can show that, if there exist a PPT adversary $\mathcal{A}_2$ that distinguishes between the output of $\mathcal{S}_1$ and $\mathcal{S}_2$ with advantage ε_2, then there exists a PPT adversary $\mathcal{B}_2$ that solves Extended-$\mathtt{MLWE}_{k+2,n-k-2,\chi,\mathcal{C},\sigma}$ instances with advantage at least $\frac{\varepsilon_2}{2}$.

For $3\leq i\leq\kappa$, we consider a simulator $\mathcal{S}_i$, which have the knowledge of the secret $\{\boldsymbol{s}_i,m_i\}_{1\leq i\leq\kappa}$. It generates $\{\boldsymbol{t}_s^{(j)},\boldsymbol{t}_m^{(j)}\}_{1\leq j<i}$ similar to the simulator $\mathcal{S}_{i-1}$ and samples $\boldsymbol{\delta}_i\leftarrow\mathcal{R}_q^{k+1}$ and computes

$$\begin{bmatrix}\boldsymbol{t}_s^{(i)}\\ \boldsymbol{t}_m^{(i)}\end{bmatrix}=\boldsymbol{\delta}_i+\begin{bmatrix}\boldsymbol{0}\\ m_i\end{bmatrix} \tag{31}$$

Now, with same logic as before, we can say that if there exist a PPT adversary $\mathcal{A}_i$ that distinguishes between the output of $\mathcal{S}_{i-1}$ and $\mathcal{S}_i$ with advantage ε_i, then there exists a PPT adversary $\mathcal{B}_i$ that solves Extended-$\mathtt{MLWE}_{k+1,n-k-1,\chi,\mathcal{C},\sigma}$ instances with advantage at least $\frac{\varepsilon_i}{2}$.

Assume there exists a PPT adversary $\mathcal{A}$ that distinguishes the output of simulators $\mathcal{S}_0$ and $\mathcal{S}_\kappa$ with advantage ε. Therefore, $\varepsilon\leq\sum_{i=1}^{\kappa}\varepsilon_i$. Now for $i=1,2$ algorithm $\mathcal{B}_i$ solves for Extended-$\mathtt{LWE}_{k+2,n-k-2,\chi,\mathcal{C},\sigma}$ instances with advantage $\varepsilon_i/2$ and for $3\leq i\leq\kappa$ algorithm $\mathcal{B}_i$ solves for Extended-$\mathtt{LWE}_{k+1,n-k-1,\chi,\mathcal{C},\sigma}$ instances with advantage $\varepsilon_i/2$.

Let us consider that $\varepsilon_{\mathrm{mlwe}} = \max_{1\le i\le\kappa}\{\varepsilon_i/2\}$. Then we have

$$\kappa\varepsilon_{\mathrm{mlwe}} \ge \sum_{i=1}^{\kappa}\varepsilon_i/2 \ge \varepsilon/2$$

Therefore, $\varepsilon \le \kappa\varepsilon_{\mathrm{mlwe}}$. If solving Extended-$\mathtt{LWE}_{t,n-t,\chi,\mathcal{C},\sigma}$ have negligible advantage for $t=k+1$ and $t=k+2$, the statement of the theorem follows.

s

Appendix F Proof of Lemma 8

Proof. Similar to the proof of Lemma 6, we consider a binary matrix $H\in\{0,1\}^{R\times N}$ where the R rows represent the prover's randomness and N columns represent the verifier's randomness. We define an extractor $\mathcal{E}$ in the following manner:

1. $\mathcal{E}$ first samples a random r and challenge $c^{(0)}\leftarrow\mathcal{C}$. Then it checks if $H(r,c^{(0)})$ is 1. If not, $\mathcal{E}$ aborts.
2. Otherwise, $\mathcal{E}$ samples along row r without replacement until it finds two $c^{(1)}$, $c^{(2)}$ such that $H(r,c^{(i)})=1$ for $i=0,1,2$.

According to [4], $\mathcal{E}$ extracts three valid transcripts

$$\mathtt{tr}^{(i)} = (\{\boldsymbol{\tau}_j\}_{1\le j\le\kappa}, h, h', \hat{\boldsymbol{u}}, v, v', c^{(i)}, \{\boldsymbol{z}_j^{(i)}\}_{1\le j\le\kappa}) \quad \text{for } i=0,1,2$$

with probability at least $\varepsilon - 2/|\mathcal{C}|$.

First we focus on $\mathtt{tr}^{(0)}$ and $\mathtt{tr}^{(1)}$. Define

$$\bar{c} = c^{(0)} - c^{(1)} \quad \text{and} \quad \bar{\boldsymbol{s}}_j = \frac{\boldsymbol{z}_j^{(0)} - \boldsymbol{z}_j^{(1)}}{c^{(0)} - c^{(1)}} \qquad \text{for } 1\le j\le\kappa$$

By the definition of $\mathcal{C}$, $\|\bar{c}\|_\infty \le 2$, $\|\bar{c}\bar{\boldsymbol{s}}_j\| \le 2\sigma\sqrt{2nd}$, and we have $\boldsymbol{B}_j\bar{\boldsymbol{s}}_j = \boldsymbol{t}_s^{(j)}$ for all $1 \le j \le \kappa$. We define the extracted message as $\bar{\boldsymbol{m}} = (\bar{m}_1, \bar{m}_2, \cdots, \bar{m}_\kappa)$ where $\bar{m}_j = \boldsymbol{t}_m^{(j)} - \boldsymbol{a}_j^\top\bar{\boldsymbol{s}}_j \pmod p$ and $\bar{g} = h - \boldsymbol{b}^\top\bar{\boldsymbol{s}}_1$. Now, let $\boldsymbol{y}_j^{(1)} = \boldsymbol{z}_j^{(0)} - c^{(0)}\bar{\boldsymbol{s}}_j = \boldsymbol{z}_j^{(1)} - c^{(1)}\bar{\boldsymbol{s}}_j$. Consider the third transcript $\mathtt{tr}^{(2)}$ and we define $\boldsymbol{y}_j^{(2)} = \boldsymbol{z}_j^{(2)} - c^{(2)}\bar{\boldsymbol{s}}_j$. Since $\mathtt{tr}^{(i)}$ are non-aborting transcripts, for all $1\le j\le\kappa$ we have:

$$\begin{aligned}\boldsymbol{B}_j(\boldsymbol{y}_j^{(1)} - \boldsymbol{y}_j^{(2)}) &= \boldsymbol{B}_j\bar{\boldsymbol{z}}_j' - \bar{c}'\boldsymbol{B}_j\bar{\boldsymbol{s}}_j \\ &= \boldsymbol{B}\bar{\boldsymbol{z}}_j' - \bar{c}'\boldsymbol{t}_s^{(j)} \\ &= 0\end{aligned}$$

where $\bar{\boldsymbol{z}}_j' = \boldsymbol{z}_j^{(0)} - \boldsymbol{z}_j^{(2)}$ and $\bar{c}' = c^{(0)} - c^{(2)}$. Therefore, either we have $\boldsymbol{y}_j^{(1)} = \boldsymbol{y}_j^{(2)}$ for all $1\le j\le\kappa$ or the extractor solves $\mathtt{MSIS}_{k,n,B}$ for $B = 4\sigma\sqrt{2nd}$. We will assume the formal from now on.

Next we define $\bar{\boldsymbol{u}} = (\bar{u}_1, \bar{u}_2, \cdots, \bar{u}_\kappa)^\top$, where $\bar{u}_j = -\boldsymbol{a}_j^\top \boldsymbol{y}_j^{(2)}$ for all $1 \le j \le \kappa$. Then from the verification equation (2) of $\Pi_{\mathrm{Quad}}^{(2)}$ for $i = 0,1,2$ we have

$$(\boldsymbol{w}^{(i)})^\top \boldsymbol{R}_2 \boldsymbol{w}^{(i)} + c^{(i)} \boldsymbol{r}_1^\top \boldsymbol{w}^{(i)} + (c^{(i)})^2 r_0 = f^{(i)} - c^{(i)} b^{(i)} \pmod{p}$$

where

$$\begin{aligned} \boldsymbol{w}^{(i)} &= \begin{bmatrix} c^{(i)} \boldsymbol{t}_m^{(1)} - \langle \boldsymbol{a}_1, \boldsymbol{z}^{(i)} \rangle \\ \vdots \\ c^{(i)} \boldsymbol{t}_m^{(\kappa)} - \langle \boldsymbol{a}_\kappa, \boldsymbol{z}^{(i)} \rangle \end{bmatrix} - p\boldsymbol{u}_h \pmod{p} \\ &= c^{(i)} \bar{\boldsymbol{m}} + \boldsymbol{u}_l \pmod{p} \end{aligned}$$

where, $\boldsymbol{u}_l = \bar{\boldsymbol{u}} \pmod{p}$ and $\boldsymbol{u}_h = \lfloor \bar{\boldsymbol{u}}/p \rfloor$, $f^{(i)} = c^{(i)} h - \boldsymbol{b}^\top \boldsymbol{z}_1 + v$, $b^{(i)} = c^{(i)} h' - \boldsymbol{d}^\top \boldsymbol{z}_2 + v'$. Since the third verification check condition says that $b^{(i)} = 0$ for all $i = 0,1,2$. for $i = 0,1,2$ we have:

$$(c^{(i)})^2 \big(\bar{\boldsymbol{m}} \boldsymbol{R}_2 \bar{\boldsymbol{m}} + \boldsymbol{r}_1^\top \bar{\boldsymbol{m}} + r_0 \big) + c^{(i)} g_1' + g_0' = 0 \pmod{p}$$

where

$$\begin{aligned} g_1' &= \bar{\boldsymbol{m}} \boldsymbol{R}_2 \boldsymbol{u}_l + \boldsymbol{u}_l \boldsymbol{R}_2 \bar{\boldsymbol{m}} + \boldsymbol{r}_1^\top \boldsymbol{u}_l \\ g_0' &= \boldsymbol{u}_l \boldsymbol{R}_2 \boldsymbol{u}_l + \boldsymbol{b}^\top \boldsymbol{y}_1^{(2)} - v \end{aligned}$$

We can write three equations as follows:

$$\begin{bmatrix} 1 & c^{(0)} & (c^{(0)})^2 \\ 1 & c^{(1)} & (c^{(1)})^2 \\ 1 & c^{(2)} & (c^{(2)})^2 \end{bmatrix} \begin{bmatrix} g_0' \\ g_1' \\ \bar{\boldsymbol{m}} \boldsymbol{R}_2 \bar{\boldsymbol{m}} + \boldsymbol{r}_1^\top \bar{\boldsymbol{m}} + r_0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \pmod{p}$$

Since difference of two challenges in $\{c^{(0)}, c^{(1)}, c^{(2)}\}$ is invertible in $\mathcal{R}_p$, we must have $\bar{\boldsymbol{m}} \boldsymbol{R}_2 \bar{\boldsymbol{m}} + \boldsymbol{r}_1^\top \bar{\boldsymbol{m}} + r_0 = 0 \pmod{p}$.