# MPC-in-the-Head Framework without Repetition and its Applications to the Lattice-based Cryptography

Weihao Bai
*Institute of Software,*
*Chinese Academy of Sciences;*
*University of Chinese Academy of Sciences*
*Email: weihao2018@iscas.ac.cn*

Long Chen
*Institute of Software,*
*Chinese Academy of Sciences*
*Email: chenlong@iscas.ac.cn*

Qianwen Gao
*Institute of Software,*
*Chinese Academy of Sciences;*
*University of Chinese Academy of Sciences*
*Email: qianwen2021@iscas.ac.cn*

Zhenfeng Zhang
*Institute of Software,*
*Chinese Academy of Sciences*
*Email: zhenfeng@iscas.ac.cn*

***Abstract*—The MPC-in-the-Head framework has been proposed as a solution for Non-Interactive Zero-Knowledge Arguments of Knowledge (NIZKAoK) due to its efficient proof generation. However, most existing NIZKAoK constructions using this approach require multiple MPC evaluations to achieve negligible soundness error, resulting in proof size and time that are asymptotically at least $\lambda$ times the size of the circuit of the NP relation. In this paper, we propose a novel method to eliminate the need for repeated MPC evaluations, resulting in a NIZKAoK protocol for any NP relation that we call *Diet*. The proof size and time of *Diet* are asymptotically only polylogarithmic with respect to the size of the circuit $C$ of the NP relation, but are independent of the security parameter $\lambda$. Hence, both the proof size and time can be significantly reduced.**

**Moreover, Diet offers promising concrete efficiency for proving Learning With Errors (LWE) problems and its variants. Our solution provides significant advantages over other schemes in terms of both proof size and proof time, when considering both factors together. Specifically, Diet is a promising method for proving knowledge of secret keys for lattice-based key encapsulation mechanisms (KEMs) such as Frodo and Kyber, offering a practical solution to future post-quantum certificate management. For Kyber 512, our implementation achieves an online proof size of 83.65 kilobytes (KB) with a preprocessing overhead of 152.02KB. The implementation is highly efficient, with an online proof time of only 0.68 seconds and a preprocessing time of 0.81 seconds. Notably, our approach provides the first reported implementation of proving knowledge of secret keys for Kyber 512 using post-quantum primitives-based zero-knowledge proofs.**

*Weihao Bai & Long Chen led efforts with equal contribution and should be considered co-first authors. Long Chen and Zhenfeng Zhang are the corresponding authors.*

## 1. Introduction

Zero-knowledge (ZK) proofs [1], [2], [3] and their non-interactive variants [4], [5], [6] are among the most fundamental and versatile cryptographic primitives for both theory and practice. Non-Interactive Zero-Knowledge Arguments of Knowledge (NIZKAoK) is a protocol that enables a computationally bounded prover to convince a verifier that they know a witness for a specific statement, without disclosing any further information about the witness. These protocols have been crucial building blocks for real-world cryptography applications. For instance, a Certificate Authority (CA) may require an applicant to provide a NIZKAoK as proof of knowledge of a secret key to prevent rogue key attacks [7], [8], [9].

Ishai et al. [10] provided an ingenious method for constructing zero-knowledge proofs from secure multi-party computation (MPC) protocols. The MPC-in-the-Head paradigm leverages MPC protocols to enable a prover to convince a verifier of their knowledge of a witness $\omega$ with respect to a statement $x$, for any NP relation $R(x, \omega)$. This approach computes a function $f$ for the circuit of $R$ using shares $\omega_1, \ldots, \omega_n$ of $\omega$ as inputs. The prover emulates the MPC protocol in her head, yielding one transcript per party. By revealing a subset of transcripts and checking their consistency, the verifier is convinced that the prover knows $\omega$. The MPC-in-the-Head paradigm provides a powerful tool for constructing ZK proofs based on MPC protocols.

In recent years, several notable works [11], [12], [13], [14], [15], [16] have demonstrated the impressive efficiency of NIZKAoK proofs based on semi-honest MPC protocols. However, these constructions require repeating the MPC evaluation during the proof, which may be computationally expensive. Intuitively, for a cheating prover to violate soundness, they must generate at least one pair of inconsistent views. To detect such behavior with probability $1/\binom{n}{2}$ by randomly revealing the views of two players among $n$ play-

ers, the soundness error must be reduced to $2^{-\lambda}$ for security parameter $\lambda$. This requires the ZK prover to repeat the MPC evaluation $O(\lambda n^2)$ times. Even if a constant number of MPC players is chosen, for an arbitrary circuit $C$, the proof size and proving complexity are both at least $O(\lambda|C|)$, where $\lambda$ is the security parameter.

Ishai et al. [10] propose constructing NIZK proofs from maliciously secure MPC protocols, rather than semi-honest MPC protocols, to enhance efficiency and avoid redundant MPC evaluations. The underlying intuition is that security against $t$ malicious players ensures that any attempt to violate the correctness of the MPC protocol $\Pi_f$ (or the soundness of the ZK protocol) must result in inconsistencies between the views that are "well spread" in such a way that opening $t$ random views reveals an inconsistency with overwhelming probability. Consequently, malicious security benefits the verifier by enabling them to detect inconsistencies with higher probability from a single execution of the protocol. By leveraging highly efficient and perfectly robust MPC protocols with minimal communication [17], [18], it is possible to construct NIZKAoK protocols without repeating the MPC evaluation. The proof size and computational complexity for the proved circuit $C$ are both $O(|C|)+\mathsf{poly}(\lambda)(\lambda, \log|C|)$. Notably, avoiding the repeated MPC evaluation may cause the proof size and the proving complexity to be both proportional to the circuit size, with a *constant factor*, which would be a significant efficiency improvement for practical applications.

Although NIZKAoK protocols based on maliciously secure Multi-Party Computation (MPC) protocols are asymptotically more efficient than those based on semi-honest MPC protocols, their practical efficiency has not been fully explored. It remains an open question whether avoiding repeating the MPC evaluation can also significantly improve the concrete efficiency of the NIZKAoK protocols. Ishai et al.'s black-box construction [10] suggests leveraging the MPC protocols proposed by Damgård et al. [17], [18], which have optimal asymptotic overheads. However, these protocols are considerably complex, which may raise doubts about their concrete efficiency. Indeed, all of these protocols rely on complex information-theoretic Verifiable Secret Sharing (VSS), which may impact their practical efficiency. Moreover, these schemes rely on packed Shamir's secret sharing, which requires one to convert the circuit into one consisting of $l$-fold gates. In this case, the basic operation of the MPC protocol is based on a vector-wise approach. Rearranging the circuit may be challenging and limit the applicability of these schemes to specific problems.

It would be interesting to explore whether the concrete efficiency of the current MPC-in-the-Head framework can be improved by avoiding the repetition of the MPC evaluation. Additionally, we hope that any new framework developed will be beneficial to real-world applications.

## 1.1. Our Contributions

In this work, we aim to improve the concrete efficiency of NIZKAoK protocols derived from MPC-in-the-Head via avoiding the repetition of the MPC evaluation. Our key observation is that leveraing a full-fledged perfectly robust MPC protocol, as proposed by Ishai et al. [10], is not the only way to reduce the soundness error. Specifically, we observe that carefully modifying the criteria allows for simple Shamir's secret sharing, instead of complex information-theoretic VSS, to suffice in guaranteeing the soundness of NIZK with negligible error. This eliminates the need to repeat MPC evaluations $O(\lambda)$ times at low cost, reducing computational and communication overheads.

Moreover, we observe that it is always cheaper to verify each gate than to compute them. When verifying the correctness of a multiplication gate, involving a random share of intermediate results for the multiplication gate can avoid the generation of double randomness in the MPC preprocessing phase. This may further reduce the computational and communication overheads associated with MPC-in-the-Head approaches, enabling the design of a more efficient and practical NIZKAoK protocol for real-world applications.

Based on the aforementioned observations, we propose a NIZKAoK system named "Diet". Our system not only imposes an asymptotically polylogarithmic overhead for any NP relation, but also achieves significant concrete efficiency improvements compared to all existing NIZK proving techniques in multiple specific scenarios, particularly for proving large-scale arithmetic circuits. By improving concrete efficiency, our approach significantly reduces the computational and communication complexity of proving lattice problems such as plain LWE and ring-LWE, making it more practical and efficient for real-world applications, especially for post-quantum Public Key Infrastructure (PKI) systems.

**NIZKAoK for lattice.** Zero-knowledge proofs of knowledge have been extensively studied in the literature [13], [19], [20], [21] for lattice problems, such as plain LWE and ring-LWE. Two main approaches have been proposed: direct construction of a zero-knowledge proof from the algebraic properties of the problem, or using a generic technique like SNARKs [22] or MPCitH [10].

For the first approach, Lyubashevsky et al. [19], [20], [21] proposed the best-known zero-knowledge proof of knowledge protocol for LWE, achieving an impressive proof size of 33.3KB. However, achieving such a proof size may require a rejection sampling rate as large as 0.85 (Table 3 in [15]), which means that the repetition time for proof generation is likely to exceed 6. To our best knowledge, these constructions have never been reported to be implemented. It is worth noting that the size-optimal protocols require that $q$ be an NTT-friendly prime, which limits their applicability to arbitrary arithmetic circuits and makes them unsuitable for proving plain LWE-based schemes with power-of-2 modules, such as FrodoKEM.

For the second approach, SNARKs offer an asymptotically constant rate proof size, but the runtime of existing concrete constructions [23] is estimated to be in the range of dozens of seconds [24]. Currently, the only promising practical construction for LWE problems may be from Baum and Nof [13], who presented an MPCitH-based approach

with preprocessing for zero-knowledge proof of binary LWE secrets. Their approach achieves a proof time of 2.4 seconds and a proof size of 4.1MB.

In contrast to Baum and Nof's work, Diet achieves a reduction of at least 80% in proof size while still maintaining a competitive running time. A detailed comparison between Diet and Baum and Nof's approach [24] for binary LWE secrets is provided in Table 1. This demonstrates the potential of Diet as a practical solution for zero-knowledge proofs of knowledge in lattice problems. Such proofs could be used to provide proof of knowledge of the lattice-based commitment scheme [25] or to prove the well-formedness of somewhat homomorphic encryption ciphertexts [26].

**Applications for post-quantum PKI systems.** Public Key Infrastructure (PKI) systems enable secure communication over a network using public key cryptography. In PKI, CAs issue digital certificates to help users verify each other's identity and public keys. In PKI systems, proof of the knowledge of secret keys (KOSK) is necessary to ensure the security of the system against rogue key attacks. Without such proof, an attacker could potentially impersonate a legitimate user by selecting an arbitrary public key and applying for a certificate from a CA by falsely claiming to possess the corresponding secret key. By requiring proof of the knowledge of the secret key, a legitimate user can demonstrate that they actually know the secret key associated with a given public key in the issued certificate and should be trusted to use that key for secure communication. This helps to prevent unauthorized access to sensitive information and ensures that only authorized parties have access to the resources protected by the PKI system.

Diet is a promising method for proving the knowledge of secret keys in lattice-based key encapsulation mechanisms (KEM), offering a practical solution to certificate management for future post-quantum secure protocols like the KEM TLS [30]. We have developed an implementation of Diet for FrodoKEM 640 [31] that provides proof of knowledge of secret keys with a size of 473.51 kilobytes (KB) and a preprocessing overhead of 123.36 KB. The implementation has a proof time of 12.81 seconds and a preprocessing time of 6.5 seconds and a verification time of 7.3 seconds. Similarly, for Kyber 512 [32], our implementation achieves a proof size of 83.65 kilobytes (KB) with a preprocessing overhead of 152.02 KB. The implementation is highly efficient, with a proof time of only 0.68 seconds, a preprocessing time of 0.81 seconds, and a verification time of 0.84 seconds. To the best of our knowledge, our implementation is the first to provide proof of knowledge of secret keys for these post-quantum KEMs within seconds.

## 1.2. Technique Overviews

Below is a summary of the novel techniques utilized in the construction of Diet.

**Consistency check on Shamir's secret shares.** A notable observation is that the semi-honest secure BGW MPC protocol [33] can be utilized to construct NIZK protocols without the need for repeated evaluation of the verification circuit via MPC, if the verifier additionally checks the consistency of all broadcasted secret shares in the final output gates. This approach offers an efficient alternative to constructing NIZK protocols while avoiding redundant MPC evaluations. In contrast, the general approach presented by Ishai et al. [10] requires MPC protocols to have robustness against malicious players, which may impose additional computational and communication overheads. This, in turn, may impact the efficiency of the NIZK.

The BGW protocol is an MPC protocol based on Shamir's secret sharing. It achieves perfect security against semi-honest adversaries who control a minority of parties. In this protocol, the parties compute shares of the output of a circuit gate for each gate of an algebraic circuit. These shares are computed given shares of the input wires of that gate. The addition gates in the circuit can be emulated using local computation only. However, the parties must interact in order to emulate the computation of multiplication gates. Finally, the parties reconstruct the secrets from the shares of the output wires of the circuit to obtain their output.

When converting a BGW protocol to a NIZK protocol via the MPC-in-the-Head paradigm, the witness is shared via Shamir's secret sharing instead of the additional secret sharing used by Ishai et al. [10]. Let $R_{\mathcal{L}}$ denote a relation corresponding to an NP language $\mathcal{L}$. That is, $R_{\mathcal{L}}(x, \omega) = 1$ if and only if $x \in \mathcal{L}$ and $\omega$ is a witness for $x$. The ZK protocol $\Pi_{ZK}$ begins with the prover carrying out all the steps of an $n$-party BGW MPC protocol for the circuit $R_{\mathcal{L}}(x, \cdot)$ in the prover's head. First, the prover secretly shares $\omega$ into $\omega_1, \ldots, \omega_n$ via Shamir's secret sharing and executes the BGW MPC protocol among $n$ virtual parties to produce the protocol transcript of inputs, initial randomness, and messages broadcast during the execution of MPC. The prover then sends commitments of the transcript to the verifier. Next, the verifier selects a random set $S$ of $|S| < n$ parties and challenges the prover to open the commitments to the private inputs, their randomness, and all messages sent or received by parties in $S$.

Our approach differs from the one proposed by Ishai et al. [10] mainly in the acceptance criteria of the verifier. While in [10] the verifier accepts if the openings form consistent views of the MPC execution and every party in the set $S$ follows the protocol and finally outputs 1, we require the verifier to additionally verify that the broadcasted secret shares in the final output gates belong to the same polynomial during the reconstruction process. The intuition behind this additional verification is that if a malicious prover intends to modify the final result of the MPC evaluation, they must alter $n - d$ broadcast shares in the final reconstruction process. If less than $n-d$ shares are modified, the verifier will reject the shares as they do not belong to the same polynomial. The soundness error arises from whether the selected subset $S$ contains a modified party. The soundness error is bounded by $\binom{d}{t}/\binom{n}{t}$, which is negligible for appropriately chosen parameters.

**Committing random shares of intermediate results.** Our

TABLE 1: Comparison of Proof Sizes and Runtime for Several Schemes

| Scheme | Technique | PLWE[2] | | Frodo640[3] | | Kyber512[4] | |
|---|---|---|---|---|---|---|---|
| | | Size | Time(s) | Size | Time(s) | Size(KB) | Time(s) |
| Stern [27] | ZKP from SIS | 4.3MB | × | × | × | × | × |
| [24] | Σ-protocol | 444KB | × | × | × | × | × |
| Ligero [28] | zkSNARK from PCPs | 200KB | × | × | × | × | × |
| Aurora [23] | zkSNARK for R1CS | 71KB | × | × | × | × | × |
| [29] | ZKP from MLWE & MSIS | × | × | × | × | 19 | × |
| BN20 [13] | MPCitH | 4.1MB | 2.4 | $\geq$ 8.42 MB | × | × | × |
| Ours[1] | MPCitH | 245.88KB (350KB) | 0.528 (5.596) | 473.51KB(123.36KB) | 12.81(6.5) | 83.65(152.02) | 0.68(0.81) |

"×" indicates estimates for parameter regimes not available in the original paper or subsequent literature.
[1] $(\cdot)$ means the preprocessing time and size.
[2] PLWE parameters: modulus $q = 2^{61}$,number of secret entries $|(s, e)| = 4096$, binary secrets $\{0, 1\}$, 128 bit security level.
[3] Frodo640 parameters: modulus $q = 2^{15}$,number of secret entries $|(s, e)| = 10640$, secrets $[0, \pm 12]$, 128-bit security level.
[4] Kyber512 parameters: modulus $q = 3329$,number of secret entries $|(s, e)| = 1024$, secrets $[0, \pm 2]$, 128-bit security level.

second observation is that for NIZK constructions, it is always more cost-effective to verify each multiplication gate by previously committing shares of intermediate results.

In Shamir's secret sharing scheme over a finite field $\mathbb{F}$, a polynomial $p(x)$ of degree $d$ with constant term $s$ is randomly selected from $\mathbb{F}[x]$. The share of the $i$-th party $P_i$ is then set to $p(\alpha_i)$, where $\alpha_1, \ldots, \alpha_n$ are distinct non-zero field elements, and $s \in \mathbb{F}$ is the secret to be shared. Given shares $p(\alpha_i)$ and $q(\alpha_i)$ of the two input wires to a multiplication gate, we can obtain shares of a polynomial $r(x)$ with constant term $p(0) \cdot q(0)$ as desired by computing $r(\alpha_i) = p(\alpha_i) \cdot q(\alpha_i)$. However, the degree of $r(x)$ is $2d$, since the degrees of $p(x)$ and $q(x)$ are both $d$. To further compute the protocol, MPC protocols usually involve a complex interactive procedure to reduce the degree of the polynomial $r(x)$ back to $d$. For example, this may involve resharing every coefficient of $r(x)$ [33] or using a preprocessing phase to generate double random sharings [34] or Beaver's triples [35]. However, the resharing method incurs $O(n^2)$ communication complexity, while the pre-generated double random sharings or Beaver's triples must be verified via the cut-and-choose technique [11], [13]. All of these approaches significantly increase the NIZK-proof complexity and size during the MPCitH.

We propose a method to reduce the NIZK proof complexity and size by requiring the prover to commit a new set of random shares of all intermediate results for each multiplication gate in MPC protocols. Specifically, instead of using the traditional approach of reducing the degree of the polynomial $r(x)$ back to $d$, the prover can provide another random polynomial $r'(x)$ with degree $d$ and prove that $p(0) \cdot q(0) - r'(0) = 0$ by outputting all shares $p(\alpha_i) \cdot q(\alpha_i) - r'(\alpha_i)$. Then, further computations can be carried out on the degree-$d$ polynomial $r'$. Our approach only requires the prover to commit one more polynomial $r'(x)$, making it more efficient than other methods that involve generating double random sharings or using Beaver's triples.

**MPCitH via Packed Secret Sharing.** To further reduce the communication and computation complexity of MPC protocols, the packed secret-sharing technique introduced by Franklin and Yung [36] can be used instead of the traditional Shamir's secret sharing. This technique can also be adapted for the MPC-in-the-Head approach. For large-scale algebraic circuits, the amortized proof size and computational cost for each gate can be reduced from $O(\lambda)$ to $O(1)$ by using this technique. This reduction in complexity is achieved by packing multiple secrets into a single tuple of secret sharing, thereby reducing the number of rounds and messages required to perform the computation.

When using the packed secret-sharing technique, each addition and multiplication gate operates on vectors coefficient-wise. However, permuting the coefficients in a vector can be challenging. To address this issue, we propose leveraging a preprocessing phase to handle any linear transformations of vectors. For a linear transformation $\mathcal{T}$, our approach involves generating multiple pairs of linear transforms of the form $[\vec{r}]_d$ and $[\mathcal{T}(\vec{r})]_d$ in the preprocessing phase. Here, $\vec{r}$ is a randomly chosen vector and $\mathcal{T}$ is the corresponding linear transformation matrix. To prove the correctness of a linear transformation $\mathcal{T}$ on a vector $\vec{x}$ using our proposed approach, the emulated players reconstruct $\vec{x} + \vec{r}$ by computing $[\vec{x} + \vec{r}]_d$, compute the linear transformation $\mathcal{T}(\vec{x} + \vec{r})$ directly and then secret-share it. The emulated players can then subtract shares of $[\mathcal{T}(\vec{x}+\vec{r})]_d$ and $[\mathcal{T}(\vec{r})]_d$ locally to obtain $[\mathcal{T}(\vec{x})]_d$. This process allows for the efficient and secure verification of linear transformations on vectors.

In addition, it is necessary to verify the random pairs $[\vec{r}]_d$ and $[\mathcal{T}(\vec{r})]_d$ in the MPC-in-the-Head scenario. To achieve this, we propose an efficient and secure cut-and-choose technique. To generate random pairs $[\vec{r}]_d, [\mathcal{T}(\vec{r})]_d$, the prover generates and commits $k + v + 1$ random vector pairs $(\vec{f_0}, \mathcal{T}(\vec{f_0})), \ldots, (\vec{f_{k+v}}, \mathcal{T}(\vec{f_{k+v}}))$. To verify these pairs, the prover constructs a polynomial of vectors $\vec{f_0} + \vec{f_1}x + \cdots + \vec{f_{k+v}}x^{k+v}$. To ensure the validation of all $[\vec{f_j}]_d, [\mathcal{T}(\vec{f_j})]_d$ for $j = 1, \ldots, k + v + 1$, the verifier chooses random elements $\alpha_1, \ldots, \alpha_k$ and checks whether $\mathcal{T}(\vec{f_0} + \cdots + \vec{f_{k+v}}\alpha_i^{k+v})$ and $\mathcal{T}(\vec{f_0}) + \cdots + \mathcal{T}(\vec{f_{k+v}})\alpha_i^{k+v}$ are equal for $i = 1, \ldots, k$. If this check passes, the verifier chooses another random element $\alpha_1, \ldots, \alpha_v$ and computes $\vec{f_0} + \cdots + \vec{f_{k+v}}\alpha_i^{k+v} = \vec{r_j}$ and $\mathcal{T}(\vec{f_0}) + \cdots + \mathcal{T}(\vec{f_{k+v}})\alpha_i^{k+v} = \mathcal{T}(\vec{r_j})$. Notably, since the preprocessing phase for a linear transformation $\mathcal{T}$ only needs to be executed once and can be reused for multiple gates with the same type of transformation, the cost of pre-

computation can be eventually amortized.

### 1.3. Other Related Works

**Verifiable Key Generation vs. Proof of Knowledge of Secret Key.** Guneysu et al. [9] recently proposed a verifiable key generation method for lattice-based KEMs such as Kyber and Frodo. Their algorithm generates a proof of possession simultaneously with the key generation process. However, due to the concurrent generation procedure, their technique can only be used during certificate issuance and not during other periods of the certificate's lifecycle, such as revocation.

In comparison to their approach, our proposed proof guarantees that the lattice public key is well-formed, providing an additional level of security guarantee. This is in contrast to their proof, which fails to ensure that the generated public key is well-formed, potentially allowing dishonest parties to register an ill-formed public key that is not an LWE instance.

Moreover, several works [7], [8] have noted that proofs of possession of a secret key are insufficient to resist rogue key attacks, where the adversary may choose a public key as a function of an honest user's key. To fundamentally prevent rogue-key attacks, it is necessary to require proof of knowledge of the secret key during public key registration to a CA, rather than just providing proof of possession. Hence, our proposed NIZKAoK provides stronger security than the proof of possession by Guneysu et al. [9].

**MPCitH from Shamir's secret sharing.** Feneuil and Rivain [16] recently also suggested that the MPCitH from Shamir's secret sharing may be more efficient than the additional secret sharing-based approach. Braun et al. [37] extend the MPC-in-the-head framework, used in recent efficient zero-knowledge protocols, to work over the ring $\mathbb{Z}_{2^k}$, which is compatible with any threshold linear secret sharing scheme and draw inspiration from MPC protocols adapted for ring operations. Compared to these works, our approach achieves improved efficiency by leveraging packed secret sharing to reduce computation and proof size. We have devised an efficient method for computing a linear transformation for the packed secret vector, which is essential for making packed secret sharing work. Moreover, we have developed novel methods for handling multiplication gates without the need to verify preprocessed Beaver's triples. This further reduces computational overhead and improves the efficiency of our approach.

## 2. Preliminaries

Throughout the paper, we will assume that elements in $\mathbb{Z}_q$ are represented by integers in the range $\left[-\frac{q-1}{2}, \frac{q-1}{2}\right]$ for a prime $q$. We will represent vectors in bold-face letters and matrices in bold-face capital letters. A ring $R$, which is often taken to be a degree-$n$ polynomial ring of the form $R = \mathbb{Z}[x]/(f(X))$. The elements of $R$ can be canonically represented by their modulo $(f(X))$, which are integer polynomials of degree less than $n$. We define $R_q = R/qR = \mathbb{Z}_q[x]/(f(X))$, whose canonical representatives are polynomials of degree less than $n$ with coefficients from some set of canonical representatives of $\mathbb{Z}_q$. Throughout this work, we use $\lambda$ to denote the security parameter. We denote by $[n]$ the set of integers $\{1, \ldots, n\}$ and by $[m, n]$ the set $\{m, \ldots, n\}$.

### 2.1. Packed Secret-Sharing

We will use the packed secret-sharing technique introduced by Franklin and Yung [36] in our NIZK construction. This is similar to standard Shamir's secret-sharing [38] over $\mathbb{F}$, but here a block of $l$ different values $\vec{x} = (x_1, \ldots, x_l) \in \mathbb{F}^l$ are shared at once using a polynomial that evaluates to $x_1, \ldots, x_l$ in $l$ distinct points. Let $N$ be the number of parties and $l$ be the number of secrets that are packed in one sharing. Assume that $|\mathbb{F}| > 2N$ such that $\beta_1, \ldots, \beta_l$ and $\alpha_1, \ldots, \alpha_N$ are fixed $N + l$ distinct elements in $\mathbb{F}$. A $d$-degree packed Shamir sharing of $\vec{x} = (x_1, \ldots, x_l) \in \mathbb{F}^l$ is a vector $(w_1, \ldots, w_N)$ for which there exists a polynomial $f(\cdot) \in \mathbb{F}[X]$ of degree at most $d$ such that $f(\beta_i) = x_i$ for all $i \in \{1, \ldots, l\}$ and $f(\alpha_i) = w_i$ for all $i \in \{1, \ldots, N\}$. The $i$-th share $w_i$ is held by party $P_i$. Reconstructing a degree-$d$ packed Shamir sharing requires $d+1$ shares and can be done by Lagrange interpolation. For a random degree-$d$ packed Shamir sharing of $\vec{x}$, any $d-l+1$ shares are independent of the secret $\vec{x}$. Any vector of shares $\{s_1, \ldots, s_N\}$ among $N$ parties is called $d$-consistent if the shares correctly match a degree at most $d$ polynomial in the $N$ first points and therefore uniquely defines a block of secrets. In the following, denote by $[\vec{x}]_d$ a packed secret-sharing of the $l$ dimensional vector $\vec{x}$ using a polynomial of degree at most $d$. The $i$-th share of $[\vec{x}]_d$ is denoted as $[\vec{x}]_d(i)$.

By employing packed secret sharing, it is possible to execute secure addition or multiplication on a set of $l$ values concurrently [36]. This observation directly expedites the parallel evaluation of the same circuit with $l$ independent inputs using the MPC protocol. In particular, the evaluation of the circuit $C$ should consist of three types of operations on $l$-dimensional blocks in $\mathbb{F}^l$ (gates for blocks), namely:

- $l$-addition:

  $l\text{−}\mathsf{Add}((x_1, \ldots, x_l), (y_1, \ldots, y_l)) = (x_1{+}y_1, \ldots, x_l{+}y_l),$
- $l$-multiplication:

  $l\text{−}\mathsf{Mult}((x_1, \ldots, x_l), (y_1, \ldots, y_l)) = (x_1{\cdot}y_1, \ldots, x_l{\cdot}y_l),$
- linear transformation $\mathcal{T}$:[1]

$$\mathcal{T}(x_1, x_2, \ldots, x_l) = (x_1, x_2, \ldots, x_l) \cdot \mathcal{T}.$$

Operations (addition and multiplication) between two packed Shamir sharings are coordinate-wise. Obviously, we have $[\vec{x}]_d + [\vec{y}]_d = [\vec{x}+\vec{y}]_d$ and $[\vec{x}]_d \cdot [\vec{y}]_d = [\vec{x} \star \vec{y}]_{2d}$, where $\star$

1. We can view $\mathcal{T}$ as a matrix in $\mathbb{F}^{l \times l}$, allowing us to achieve the linear transformation by multiplying the vector with the matrix.

denotes the coordinate-wise multiplication operation. These properties directly follow from the computation of the underlying polynomials. To compute a $[\vec{x}]_d$ multiply with a constant vector $\vec{c} = (c_1, \ldots, c_l)$, we can compute $[\vec{c}]_d$ by making $f(\beta_i) = c_i$ for all $i \in \{1, \ldots, l\}$ and $f(\alpha_i) = 1$ for all $i \in \{1, \ldots, d-l+1\}$. These $d$ points already determine a unique polynomial $f$, so one makes $w_i = f(\alpha_i)$ for all $i \in \{d-l+2, \ldots, N\}$. Then one can get $[\vec{x}]_d \cdot [\vec{c}]_d = [\vec{x} \star \vec{c}]_{2d}$. Moreover, for two packed secret sharings with two different degrees $[\vec{x}]_{2d}$ and $[\vec{y}]_d$, then $[\vec{x}]_{2d} + [\vec{y}]_d = [\vec{x} + \vec{y}]_{2d}$.

Recall that $t$ is the number of corrupted parties. Also, recall that a degree-$d$ packed Shamir secret sharing scheme is of threshold $d - l + 1$. To ensure that the packed Shamir secret sharing scheme has threshold $t$ and is multiplication-friendly, we choose $l$ such that $t \leq d - l + 1$ and $2d \leq N$.

### 2.2. Schwartz-Zippel Test

Let $f(x)$ and $g(x)$ be two (single variable) polynomials with coefficients in a finite field $\mathbb{F}$. The Schwartz-Zippel test [39] can be used to determine whether $f$ and $g$ are identical or not.

**Lemma 1** ( Schwartz-Zippel Lemma)**.** *Let $p(x_1, \ldots, x_n)$ be a polynomial of degree $d$. Let $S$ be any set of numbers and $a_1, \ldots, a_n$ be $n$ random numbers drawn from $S$. Then* $\Pr[p(a_1, \ldots, a_n) = 0] \leq d/|S|$.

### 2.3. Zero-Knowledge Arguments of Knowledge

Let $L_R \in \{0,1\}^*$be an NP language and $R$ be its related NP-relation for circuits over $\mathbb{F}$. Thus $(x = (C, y), \omega) \in R$ iff $(C, y) \in L_R$ and $C(\omega) = y$. We write $R_x = \{\omega | (x, \omega) \in R\}$ for the set of witnesses for a fixed $x$.

**Definition 1** (Honest Verifier Zero-Knowledge Argument of Knowledge)**.** *Assume $(P, V)$ is a pair of probabilistic polynomial time interactive Turing machines and let $p \to [0, 1]$ be a function. We say that $(P, V)$ is a zero-knowledge argument of knowledge for the relation $R$ if the following properties hold:*

***Completeness:** If $P$ and $V$ follow the protocol on input $x \in L_R$ and private input $\omega \in R_x$ to $P$, then $V$ always outputs 1.*

***Knowledge Soundness:** There exists a probabilistic algorithm $\mathcal{E}$ called the knowledge extractor, such that for every interactive prover $\hat{P}$ and every $x \in L_R$, the algorithm $\mathcal{E}$ satisfies the following condition: let $\delta(x)$ the probability that $V$ accepts on input $x$ after interacting with $\hat{P}$. If $\delta(x) > p(x)$, then upon input $x \in L_R$ and oracle access to $\hat{P}$, the algorithm $\mathcal{E}$ outputs a vector $\omega \in R_x$ in expected number of steps bounded by $\frac{1}{\delta(x) - p(x)}$.*

***Honest Verifier Zero-Knowledge:** Let $\mathsf{view}_V^P(x, \omega)$ be a random variable describing the random challenge of $V$ and the messages $V$ receives from $P$ with input $\omega$ during the joint computation on common input x. Then, there exists a PPT simulator S, such that for all $x \in L_R$,$\omega \in R_x$: $S(x) \approx_c \mathsf{view}_V^P(x, \omega)$.*

## 3. The Construction Framework

In this section, we present a novel non-interactive zero-knowledge argument of knowledge (NIZKAoK) that is built upon the MPC-in-the-Head paradigm and utilizes packed secret sharing. Here we assume the circuit consists of three types of $l$-fold gates: $l$-fold addition gates, $l$-fold multiplication gates, and linear transformation gates. To optimize computation and communication costs, we propose different novel methods for proving each type of gate in the context of packed secret sharing. Additionally, we provide rigorous security proof for our NIZKAoK that establishes its soundness and zero-knowledge properties.

### 3.1. NIZKAoK from Packed Secret Sharing

We consider a zero-knowledge proof system designed for an NP language $\mathcal{L}$ that corresponds to a relation $R$. Given a statement $x \in \mathcal{L}$, we assume the existence of a witness $\omega$ such that $R(x, \omega) = 1$ and the function $R(x, \cdot)$ can be converted into an arithmetic circuit $C$ over the field $\mathbb{F}$ such that $C(\omega) = 1$.

Without loss of generality, we assume that $\omega$ can be represented as a collection of $n$ $l$-dimensional vectors, denoted as $\vec{\omega}_1, \ldots, \vec{\omega}_n \in \mathbb{F}^l$, where each $\vec{\omega}_i$ is a vector over the field $\mathbb{F}$. Based on the findings presented in [18], it is possible to efficiently convert the circuit $C$ into a circuit $C'$ that consists of three types of $l$-fold gates: $l$-fold addition gates, $l$-fold multiplication gates, and linear transformation gates. Consequently, the function $R(x, \cdot)$ can be transformed into an arithmetic circuit $C'$ over the field $\mathbb{F}^l$ such that $C'(\vec{\omega}_1, \ldots, \vec{\omega}_n) = \vec{1}$.

To establish the NP relation $R$, the prover begins by computing the circuit $C'$ and carefully recording all intermediate results associated with the multiplication gates and linear transformation gates. Subsequently, the prover simulates $N$ virtual parties and employs a degree-$d$ packed secret sharing scheme to divide each vector $\vec{\omega}_i \in \mathbb{F}^l, i = 1, \ldots n$ into $N$ shares denoted as $[\vec{\omega}_i]_d = ([\vec{\omega}_i]_d(1), \ldots, [\vec{\omega}_i]_d(N))$. Each party $j$ securely holds a distinct share $[\vec{\omega}_i]_d(j)$. Given that the prover possesses knowledge of the inputs and outputs of each gate, the corresponding proof can be systematically generated. Figure 1 provides an overview of the proof generation process for each gate type.

In the case of an $l$-fold addition gate, represented as $\vec{x} + \vec{y} = \vec{z}$, the virtual parties independently perform local computations to obtain the packed secret sharing $[\vec{z}]_d = [\vec{x}]_d + [\vec{y}]_d$ using the shares $[\vec{x}]_d$ and $[\vec{y}]_d$. Subsequently, they transmit $[\vec{z}]_d$ as the inputs for the subsequent gates.

In the context of an $l$-fold multiplication gate denoted as $\vec{x} \star \vec{y} = \vec{z}$, the prover instructs the virtual parties to locally compute $[\vec{z}]_{2d} = [\vec{x}]_d \cdot [\vec{y}]_d$ by multiplying their respective shares. However, using $[\vec{z}]_{2d}$ directly as input for subsequent gates is not feasible due to its secret share degree of $2d$. To address this, the prover generates a new random packed secret sharing $[\vec{z}]_d$ for $\vec{z}$ and provides proof that $[\vec{z}]_{2d} - [\vec{z}]_d$ is a secret share of $\vec{0}$. To ensure the correctness of the computation, the verifier employs a random

**Prove:** The circuit $C'$ consists of $l$-fold addition gates, $l$-fold multiplication gates and $h$ different types of linear transformations $\mathcal{T}_1, \ldots, \mathcal{T}_h$ corresponding to NP relation $R$ and the statement $x$. Specifically, the witness $\omega$ can be encoded into $\vec{\omega}_1, \ldots, \vec{\omega}_n$ where $\vec{\omega}_j \in \mathbb{F}^l$ and $C'(\vec{\omega}_1, \ldots, \vec{\omega}_n) = \vec{1}$. $H_{com}$ and $H_C$ are hash functions.

*Input Gate:*

The prover encodes the symbol $\omega$ as $\vec{\omega}_1, \ldots, \vec{\omega}_n$, where each $\vec{\omega}_j$ belongs to the field $\mathbb{F}^l$. For each $\vec{\omega}_j$, the prover generates packed secret sharings of degree $d$ and records the $i$-th share as $[\vec{\omega}_j]_d(i)$ in the set $\mathcal{V}_i$. Let $[\vec{\omega}]_d(i) = ([\vec{\omega}_1]_d(i), \ldots, [\vec{\omega}_n]_d(i))$.

*Preprocessing:*

When the circuit $C$ contains $v$ linear transformation gates of type $\mathcal{T}$, the prover does the following:

1) The prover generates $k+v+1$ random vector $\vec{f_j} \in \mathbb{F}^l$ for $j = 0, \ldots, k+v$.
2) The prover generates the $d$ degree packed secret sharing for each $\vec{f_j}$ and commits the $i$-th share of $[\vec{f_j}]_d$ and $[\mathcal{T}(\vec{f_j})]_d$ as $\mathcal{T}com_i = H_{com}\left([\vec{\omega}]_d(i), \{[\vec{f_j}]_d(i), [\mathcal{T}(\vec{f_j})]_d(i)\}_{j\in[0,k+v]}\right)$ for $i = 1, \ldots, N$. The prover also records $\mathcal{T}com_i$ in $\mathcal{V}_i$.
3) The prover computes $k+v$ random elements: $H_C(\mathcal{T}com_1, \ldots, \mathcal{T}com_N) = \alpha_1, \ldots, \alpha_{k+v} \in \mathbb{F}^l$ and computes $[\vec{\beta_j}]_d(i) = [\vec{f_0}]_d(i) + \cdots + [\vec{f_{k+v}}]_d(i)\alpha_j^{k+v}$ and $[\vec{\gamma_j}]_d(i) = [\mathcal{T}(\vec{f_0})]_d(i) + \cdots + [\mathcal{T}(\vec{f_{k+v}})]_d(i)\alpha_j^{k+v}$ for $j = 1, \ldots, k$. The prover records all shares of $[\vec{\beta_j}]_d(i)$ and $[\vec{\gamma_j}]_d(i)$ for $j = 1, \ldots, k$ in each $\mathcal{V}_i$.
4) The prover computes $[\vec{r}_{j-k}]_d(i) = [\vec{f_0}]_d(i) + \cdots + [\vec{f_{k+v}}]_d(i)\alpha_j^{k+v}$ and $[\mathcal{T}(\vec{r}_{j-k})]_d(i) = [\mathcal{T}(\vec{f_0})]_d(i) + \cdots + [\mathcal{T}(\vec{f_{k+v}})]_d(i)\alpha_j^{k+v}$ for $j = k+1, \ldots, k+v$.

The prover will perform the following computations sequentially for each gate of $C'$ and update the view of each virtual party $\mathcal{V}_i$ accordingly.

*$l$-fold Addition Gate:*

For the $l$-fold addition gate that $\vec{x} + \vec{y} = \vec{z}_{add}$, the prover computes the packed secret sharing $[\vec{z}_{add}]_d = [\vec{x}]_d + [\vec{y}]_d$ from $[\vec{x}]_d$ and $[\vec{y}]_d$.

*$l$-fold Multiplication Gate:*

For the $l$-fold multiplication gate that $\vec{x} \star \vec{y} = \vec{z}_{mult}$, the prover does the following:

1) The prover computes the packed secret sharing $[\vec{z}_{mult}]_{2d} = [\vec{x}]_d \cdot [\vec{y}]_d$.
2) The prover reshares secret $\vec{z}_{mult}$ and generates a new packed secret sharing $[\vec{z}_{mult}]_d$. He also adds the $i$-th share of $[\vec{z}_{mult}]_d(i)$ to the list $\mathcal{V}_i$.
3) The prover computes $[\vec{u}]_{2d} = [\vec{z}_{mult}]_{2d} - [\vec{z}_{mult}]_d$ and outputs the shares of $[\vec{u}]_{2d}(i)$ in $\mathcal{V}_i$.

*Linear Transformation $\mathcal{T}$:*

For each linear transformation gate $\mathcal{T}$ in $C$, the prover picks one unused random linear transformation pair $([\vec{r}]_d, [\mathcal{T}(\vec{r})]_d)$ and does the following:

1) The prover computes $[\vec{r} + \vec{x}]_d$ and reveals $\vec{x} + \vec{r}$ in each $\mathcal{V}_i$.
2) The prover deterministically reshares $\mathcal{T}(\vec{x} + \vec{r})$ and gets $[\mathcal{T}(\vec{x} + \vec{r})]_d$ for the virtual parties.
3) The prover computes $[\mathcal{T}(\vec{x})]_d(i) = [\mathcal{T}(\vec{x} + \vec{r})]_d(i) - [\mathcal{T}(\vec{r})]_d(i)$.

*Output Gate:*

For the output $y$ of the circuit $C$, the prover has already generated $[\vec{y}]_{2d}$ from the previous computation. The prover records all shares of $[\vec{y}]_{2d}$ in each $\mathcal{V}_i$.

*Challenge:*

After running all gates, the prover commits all views $(com_1, \ldots, com_N) = (H_{com}(\mathcal{V}_1), \ldots, H_{com}(\mathcal{V}_N))$ and computes challenge set $H_C(com_1, com_2, \ldots, com_N) = \mathcal{I} \subset [N]$ where $|\mathcal{I}| = t$.

*Output proof:*

Finally, the prover produces the proof $\pi = (\pi_{pre}, \pi_{online})$

$$\pi_{pre} = \left(\left\{\{[\vec{f_j}]_d(i), [\mathcal{T}_\tau(\vec{f_j})]_d(i)\}_{j\in[k+v]}\right\}_{i\in\mathcal{I},\tau\in[h]}, \left\{\{[\vec{\beta_j}]_d(i), [\vec{\gamma_j}]_d(i)\}_{j\in[k]}, \mathcal{T}_\tau com_i, \right\}_{i\notin[\mathcal{I}],\tau\in[h]}\right);$$

$$\pi_{online} = \left(\{[\vec{\omega}]_d(i), \{[\vec{z}_{mult}]_d(i)\}_{\forall\text{mul gate}}\}_{i\in\mathcal{I}}, \mathcal{I}, \left\{com_i, \{[\vec{u}]_{2d}(i)\}_{\forall\text{mul-gate}}, \{[\vec{x} + \vec{r}]_d(i)\}_{\forall\mathcal{T}_\tau\text{-gate},\tau\in[h]}\right\}_{i\notin[\mathcal{I}]}\right).$$

Figure 1: The proof algorithm of Diet

selection process to choose a subset $\mathcal{I}$ of indices from the set $[N]$, where $|\mathcal{I}| < d$. Subsequently, the prover reveals the corresponding shares of $[\vec{x}]_d$, $[\vec{y}]_d$, and $[\vec{z}]_d$ within the challenge set $\mathcal{I}$, along with all shares in $[\vec{x}]_d \cdot [\vec{y}]_d - [\vec{z}]_d$. The equation $[\vec{x}]_d \cdot [\vec{y}]_d - [\vec{z}]_d = [\vec{x} \star \vec{y}]_{2d} - [\vec{z}]_d = [\vec{0}]_{2d}$ holds true. Therefore, the verifier's task is to verify the consistency between the revealed shares of $\vec{x}$, $\vec{y}$, and $\vec{z}$ and their corresponding shares in $[\vec{x}]_d \cdot [\vec{y}]_d - [\vec{z}]_d$. Additionally, it is crucial to ensure that the shares of $[\vec{x}]_d \cdot [\vec{y}]_d - [\vec{z}]_d$ can be represented by a polynomial and accurately reflect the share of $\vec{0}$. Next the parties transmit $[\vec{z}]_d$ as inputs for subsequent gates.

For a linear transformation gate $\mathcal{T}$ denoted as $\vec{y} = \mathcal{T}(\vec{x})$ operating on a packed shared secret vector $\vec{x}$, the prover initiates the computation by precomputing pairs of the form $[\vec{r}]_d$, $[\mathcal{T}(\vec{r})]_d$ for random vectors $\vec{r}$ and the corresponding linear transformation matrix $\mathcal{T}$. Given $[\vec{r}]_d$ and $[\vec{x}]_d$, all virtual parties perform a local computation to reconstruct $\vec{x} + \vec{r}$ by adding $[\vec{r}]_d$ and $[\vec{x}]_d$, and then compute $\mathcal{T}(\vec{x} + \vec{r})$. Consequently, the prover generates $[\mathcal{T}(\vec{x} + \vec{r})]_d$. Given that $[\mathcal{T}(\vec{x} + \vec{r})]_d = [\mathcal{T}(\vec{x}) + \mathcal{T}(\vec{r})]_d$ and $[\mathcal{T}(\vec{r})]_d$, the virtual players locally subtract shares to obtain $[\mathcal{T}(\vec{x})]_d$, which can be utilized as inputs for subsequent gates.

The remaining question pertains to the generation of a random pair $[\vec{r}]_d$, $[\mathcal{T}(\vec{r})]_d$ that can be verified in the MPCitH scenario. To address this, we employ the cut-and-choose method, utilizing $k$ pairs for correctness verification and the remaining $v$ pairs for proof. Specifically, we leverage the following fact: let $\vec{f_j} = (f_j^1, \ldots, f_j^l)^T \in \mathbb{F}^l$ and a random challenge $\alpha$, one have:

$$\begin{aligned}\vec{r} &= \left(\vec{f_0}, \vec{f_1}, \ldots, \vec{f_{k+v}}\right) \cdot (1, \alpha, \ldots, \alpha^{k+v})^T \\ &= \vec{f_0} + \vec{f_1}\alpha + \cdots + \vec{f_{k+v}}\alpha^{k+v}\end{aligned}$$

while

$$\begin{aligned}&\left(\mathcal{T}(\vec{f_0}), \mathcal{T}(\vec{f_1}), \ldots, \mathcal{T}(\vec{f_{k+v}})\right) \cdot \left(1, \alpha, \ldots, \alpha^{k+v}\right)^T \\ =&\mathcal{T}\left(\vec{f_0}, \vec{f_1}, \ldots, \vec{f_{k+v}}\right)\left(1, \alpha, \ldots, \alpha^{k+v}\right)^T = \mathcal{T}\left(\vec{r}\right)\end{aligned}.$$

To generate $v$ random pairs $[\vec{r}]_d$, $[\mathcal{T}(\vec{r})]_d$, the prover initiates the process by generating $k+v+1$ random vector pairs $(\vec{f_0}, \mathcal{T}(\vec{f_0})), \ldots, (\vec{f_{k+v}}, \mathcal{T}(\vec{f_{k+v}}))$. By utilizing random elements $\alpha_1, \ldots, \alpha_k$, the prover computes $[\vec{\beta_j}]_d = [\vec{f_0}]_d + \cdots + [\vec{f_{k+v}}]_d \alpha_j^{k+v}$ and $[\vec{\gamma_j}]_d = [\mathcal{T}(\vec{f_0})]_d + \cdots + [\mathcal{T}(\vec{f_{k+v}})]_d \alpha_j^{k+v}$ for each $j \in [k]$. The prover presents the evidence of correctness when $\mathcal{T}(\vec{\beta_j}) = \vec{\gamma_j}$ holds for all $i \in [k]$. Subsequently, the prover employs the remaining $\alpha_j, j \in [k+1, k+v]$, to compute the pair of linear transformations $[\vec{r}]_d$, $[\mathcal{T}(\vec{r})]_d$. To ensure negligible soundness errors, we rely on the Schwartz-Zippel lemma, which guarantees $\left(\frac{k+v}{|\mathbb{F}|}\right)^k < \frac{1}{2^\lambda}$. Notably, this procedure only needs to be performed once for $v$ linear transformation gates with the same type of linear transformation since a single preprocessing phase can generate $v$ random pairs $[\vec{r}]_d$, $[\mathcal{T}(\vec{r})]_d$.

The verification process, illustrated in Figure 2, consists of several steps. Firstly, the verifier recomputes the linear transformation pairs and verifies their correctness. Next, the verifier recomputes the circuit and checks if the output is accurate. Finally, the verifier verifies the correctness of the opened commitments of $\mathcal{V}_i$. In particular, for each secret share that requires resharing, the verifier additionally ensures that all shares correspond to points on the same reconstructed polynomial.

TABLE 2: Asymptotic Complexity of Proof Size and Runtime

| Scheme | $l$ parallel Multip. | | $l$ parallel Add. | | Linear Trans. | |
|---|---|---|---|---|---|---|
| | Size | Cost[1] | Size | Cost | Size | Cost |
| MPCitH[2] | $O(\lambda \cdot l)$ | $O(\lambda \cdot l)$ | free | $O(\lambda \cdot l)$ | free | $O(\lambda \cdot l)$ |
| Diet | $O(l)$ | $O(l)$ | free | $O(l)$ | $O(l)$ | $O(l)$ |

[1] Cost refers to the basic arithmetic operation
[2] the MPCitH constructions [11], [13], [14] with Additional secret sharing

## 3.2. Security Proof

In this subsection, we will present our fixed theorem and provide a formal security proof for Diet. We refer the interested reader to Appendix A for a more detailed and rigorous proof.

**Theorem 1.** *Let us assume that $H_C$ is a random oracle, $H_{com}$ is collision-resistant, and the packed secret sharing parameters are denoted as $(l, t)$. The protocol, as depicted in Figure 1, when instantiated with the parameter $(N, t)$, serves as a non-interactive zero-knowledge argument of knowledge for any NP relation $\mathcal{R}$. The protocol achieves a soundness error of $\frac{\binom{2(t+l)}{t}}{\binom{N}{t}} + \left(\frac{k+v}{||F||}\right)^k \cdot \frac{t}{2(t+l)+1} \cdot \binom{N-t}{t+2l+1}$. Furthermore, if a malicious prover successfully convinces the honest verifier $V$ to accept with a probability $\hat{\epsilon} := \Pr[\langle P^*, V\rangle(x) \to 1] > \epsilon$, then there exists an efficient probabilistic extraction algorithm $E_0$. Given rewindable black-box access to $P^*$, this algorithm produces, on average, a witness $\omega$ satisfying $C(x, \omega) = 1$ by making a bounded number of calls to $P^*$, which is upper-bounded by $\frac{4}{\hat{\epsilon}-\epsilon} \cdot \left(1 + \hat{\epsilon} \cdot \frac{8 \cdot (N-t)}{\hat{\epsilon}-\epsilon}\right)$.*

## 3.3. Asymptotic Performance

In this subsection, we analyze the asymptotic complexity of Diet in Figure 1. To ensure the optimized size and computational complexity of Diet, we set that the packed size $l$ is equal to $t$, where $t \approx O(\lambda)$ corresponds to the number of opened views. As a result, we obtain $d = t + l = 2t$, and we establish $N \geq 2d + 1 = 4t + 1$. Additionally, we let $N = ct$, where $c$ is a small constant greater than 4. Here, $d$ represents the degree of the polynomials applied in the system, while $N$ denotes the number of virtual MPC players.

For an arithmetic circuit with $l$-fold gates, the proof size of Diet is only $l$ times the number of gates and constant to the security parameter. For each $l$-fold multiplication gate and transformation gate in the circuit, the proof needs to include $N$ more field elements, of which the size is $O(l)$. For each linear transformation gate, the preprocessing will

**Verify**: The verifier inputs the statement $x$ and the proof $\pi$ and outputs whether it is a validly proof.

Set: $\pi_{pre} = \left( \left\{ \{[\vec{f_j}]_d(i), [\mathcal{T}_\tau(\vec{f_j})]_d(i)\}_{j\in[k+v]} \right\}_{i\in\mathcal{I},\tau\in[h]}, \left\{ \{[\vec{\beta_j}]_d(i), [\vec{\gamma_j}]_d(i)\}_{j\in[k]}, \mathcal{T}_\tau com_i, \right\}_{i\notin[\mathcal{I}],\tau\in[h]} \right)$;

$\pi_{online} = \left( \{[\vec{\omega}]_d(i), \{[\vec{z}_{mult}]_d(i)\}_{\forall \text{mul gate}}\}_{i\in\mathcal{I}}, \mathcal{I}, \left\{ com_i, \{[\vec{u}]_{2d}(i)\}_{\forall \text{mul-gate}}, \{[\vec{x}+\vec{r}]_d(i)\}_{\forall \mathcal{T}_\tau\text{-gate},\tau\in[h]} \right\}_{i\notin[\mathcal{I}]} \right)$.

*Check the random linear transformation pairs*

1) For each $i \in \mathcal{I}$ and each type of the linear transformation $\mathcal{T}$, the verifier recalculates $\mathcal{T}com_i$ using $[\vec{\omega}]_d(i)$ and $\{[\vec{f_j}]_d(i), [\mathcal{T}(\vec{f_j})]_d(i)\}_{j\in[0,k+v]}$ provided in the proof. The verifier then checks if $H_C(\mathcal{T}com_1, \ldots, \mathcal{T}com_N) = (\alpha_1, \ldots, \alpha_{k+v}) \in \mathbb{F}^k$.
2) According to $\alpha_1, \ldots, \alpha_k$, the verifier recomputes $\{[\vec{\beta_j}]_d(i), [\vec{\gamma_j}]_d(i)\}_{j\in[k]}$ for each $i \in [\mathcal{I}]$ via the polynomial evaluation.
3) By combining the sets $\{[\vec{\beta_j}]_d(i), [\vec{\gamma_j}]_d(i)\}_{j\in[k]}^{i\notin\mathcal{I}}$ retrieved from the proof, the verifier reconstructs the sets $\{\vec{\beta_j}, \vec{\gamma_j}\}_{j\in[k]}$ and checks: $\mathcal{T}(\vec{\beta_j})? = \vec{\gamma_j}$ for each $j \in [k]$.
4) For each $j \in [k]$, the verifier validates whether all shares of $[\vec{\beta_j}]_d$ correspond to points on the same reconstructed polynomial. Likewise, the verifier checks if all shares $[\vec{\gamma_j}]_d$ correspond to points on the same reconstructed polynomial.

*Check the evaluation of the circuit*

Extract the $\{[\vec{\omega}]_d(i)\}_{i\in\mathcal{I}}$ from the proof and verify the accuracy of the evaluation using the following procedure.

1) For an $l$-fold addition gate, compute $[\vec{z}_{add}]_d(i) = [\vec{x}]_d(i) + [\vec{y}]_d(i)$ for $i \in \mathcal{I}$.
2) For an $l$-fold multiplication gate, the verifier's task is as follows.
   a) The verifier computes the share $[\vec{z}_{\text{mult}}]_{2d}(i)$, which is the product of $[\vec{x}]_d(i)$ and $[\vec{y}]_d(i)$, for each $i$ in the set $\mathcal{I}$.
   b) The verifier retrieves the $i$-th share of $[\vec{z}_{\text{mult}}]_d(i)$ for each $i$ in the set $\mathcal{I}$ and compute $[\vec{u}]_{2d}(i)$ is equal to the difference between $[\vec{z}_{\text{mult}}]_{2d}(i)$ and $[\vec{z}_{\text{mult}}]_d(i)$.
   c) The verifier retrieves the other shares of $[\vec{u}]_{2d}(i)$ for $i \notin \mathcal{I}$ from the proof. The verification process involves checking if validates whether all shares $[\vec{u}]_{2d}$ correspond to points on the same polynomial.
   d) The verifier checks if the reconstruction of $[\vec{u}]_{2d}$ is $\vec{0}$.

   According to the views $\{\omega_i\}_{i\in\mathcal{I}}$, the verifier computes the $\{[\vec{u}]_{2d}(i)\}_{i\in\mathcal{I}}$.
3) For the $j$th linear transformation gate $\mathcal{T}$ in $C$, the prover does the following:
   a) The verifier recomputes the polynomial evaluation $[\vec{r}]_d(i) = [\vec{f_0}]_d(i) + \cdots + [\vec{f}_{k+v}]_d(i)\alpha_{j+k}^{k+v}$ and $[\mathcal{T}(\vec{r})]_d(i) = [\mathcal{T}(\vec{f_0})]_d(i) + \cdots + [\mathcal{T}(\vec{f}_{k+v})]_d(i)\alpha_{j+k}^{k+v}$ for $i \in \mathcal{I}$.
   b) The verifier retrieves $\{[(\vec{x}+\vec{r})]_d(i)\}_{i\notin\mathcal{I}}$ from the proof and reconstructs $\vec{x}+\vec{r}$.
   c) The verifier verifies if all the shares $[\vec{x}+\vec{r}]_d$ correspond to points on the reconstructed polynomial.
   d) The verifier generates $[\mathcal{T}(\vec{x}+\vec{r})]_d$ and computes $[\mathcal{T}(\vec{x})]_d(i) = [\mathcal{T}(\vec{x}+\vec{r})]_d(i) - [\mathcal{T}(\vec{r})]_d(i)$ for $i \in \mathcal{I}$.
4) To check the output gate for $[y]_{2d}$, the verifier first reconstructs the value of $\vec{y}$ and checks if it is equal to $\vec{1}$. Additionally, the verifier checks if all shares correspond to points on the same reconstructed polynomial.

*Check the commitment opened successfully*

If the above check fails, then outputs 0; otherwise:

1) The verifier recomputes $com_i = H_{com}(\mathcal{V}_i)$ for each $i \in [\mathcal{I}]$.
2) According to $\{com_i\}_{i\notin\mathcal{I}}$ and $\mathcal{I}$ in $\pi$, the verifier checks: $\mathcal{I}? = H_C(com_1, com_2, \ldots, com_N)$.

Figure 2: The verification algorithm of Diet

include $\{[\vec{\beta_j}]_d(i), [\vec{\gamma_j}]_d(i)\}_{j\in[k]}^{i\notin\mathcal{I}}, \{\mathcal{T}com_i\}^{i\notin\mathcal{I}}$ in the proof. $\{[\vec{\beta_j}]_d(i), [\vec{\gamma_j}]_d(i)\}_{j\in[k]}^{i\notin\mathcal{I}}$ consists $2(N-t)k$ elements while the size of $\{\mathcal{T}com_i\}^{i\notin\mathcal{I}}$ is $N-t$ elements. Since this preprocessing procedure can generate random tuples for $v$ independent linear transformations, the amortized proof size for each linear transformation gate is $\frac{2(N-t)k+(N-t)}{v} = \frac{2(cl-l)k+(c-1)l}{v}$, which is $O(1)$ when $v = O(l)$.

Similarly, the computational cost of Diet is only $l$ times the number of $l$-fold gates and constant in the security parameter. For each $l$-fold addition gate in the circuit, the computational cost needs to include $N$ more field elements operations, of which the cost is $O(l)$. For each $l$-fold multiplication gate and transformation gate in the circuit, the computational cost needs to include $N+d$ more field elements operations, of which the cost is $O(l)$. For the linear transformation $\mathcal{T}$, it contains a reveal operation and matrix multiplication. A reveal operation consists of $2N+d$ elements while the computational cost of matrix multiplication is $2l^2$ elements. Therefore, the computational cost for each linear transformation gate is $\frac{2N+d+2l^2}{v}$, which is $O(l)$ when $v = O(l)$.

The analysis presented above can be summarized in Table 2, which demonstrates that for $l$ parallel multiplications or additions, the proof size and computational cost of our proposed Diet protocol remain constant with respect to the security parameter. Further, the amortized size and computational cost of each gate is $O(1)$. In contrast, the complexity of similar operations, except for linear transformations, in MPC-in-the-Head constructions [11], [13], [14] would be at least $O(\lambda)$ times that of Diet.

To obtain a more accurate estimate of the asymptotic proof size and computational cost, it is necessary to consider the overhead incurred by converting a traditional arithmetic circuit $C$ into a circuit $C'$ consisting of $l$-fold gates. To this end, Damgard, Ishai, and Krogstrup [18] propose leveraging Beneš networks [40] to transform any arithmetic circuit $C$ consisting of ordinary fan-in-2 gates into a circuit $C'$ consisting of $l$ addition gates, $l$ multiplication gates, and different kinds of permutations within blocks. The new circuit $C'$ computes the same function as the original circuit $C$, but more efficiently with respect to operations on blocks.

Specifically, the intuition behind the circuit transformation in [18] is to rearrange the circuit so that the values are computed block by block. Each block contains $l$ values. The transformation first divides the original circuit C into a minimal amount of layers so that every layer now only consists of one type of multiplication or addition gate. The size of the rearranged circuit $C_1$ remains the same as $O(C)$. Next, a permutation subcircuit is inserted into two layers to ensure that the circuit can be computed by only permuting blocks, permuting within blocks, or doing nothing between two layers. The transformed circuit $C_2$ will involve $O((C\log C + \mathsf{depth}(C)^2 C\log^3 C)/l)$ permutations within blocks. Finally, when we transform the above circuit $C_2$ into $C_3$, which only consists of l-fold gates, the number of l-fold addition and multiplication gates in $C_3$ is no more than $|C_2|/l = O(|C|/l)$. Moreover, $C_3$ will involve $O((C\log C + \mathsf{depth}(C)^2 C\log^3 C)/l)$ fold permutation gates except for l-fold addition and multiplication gates. Specifically, they prove the following lemma:

**Lemma 2.** *Given an arithmetic circuit C that is at least $l$ gates wide, there is an efficient algorithm to transform it into another circuit $C'$ with the following properties:*

1) *$C'(x) = C(x)$ for all inputs $x$.*
2) *$C'$ consists of $l$-fold gate.*
3) *$|C'| = O((|C|\log|C| + \mathsf{depth}(C)^2|C|\log^3|C|)/l)$.*
4) *$\mathsf{depth}(C') = O(\log^2|C|\mathsf{depth}(C))$.*

According to Lemma 2 and Table 2, we have:

**Theorem 2.** *Given an arithmetic circuit C that is at least $l$ gates wide, then there is an efficient NIZKAoK with a proof size of $O\left(|C|\log|C| + \mathsf{depth}(C)^2 n\log^3|C| + |\omega|\right) + \mathsf{poly}(\lambda)$ and computational cost of $O\left(|C|\log|C| + \mathsf{depth}(C)^2 n\log^3|C|\right) + O(\lambda)$.*

In comparison to previous works, which have a proof size and computational cost of $O(\lambda|C|)$, our proposed NIZKAoK protocol, Diet, exhibits a polylogarithmic overhead of $|C|$ that is independent of $\lambda$. This result highlights the effectiveness and efficiency of our approach for proving knowledge of arbitrary arithmetic circuits $C$.

**Remark.** The asymptotic results provided only capture the worst-case scenario for computation and communication complexity. However, it is important to note that these results may vary significantly in specific cases due to the inefficiency of the general circuit transformation process, which can introduce unnecessary permutation gates. Nevertheless, for most circuits, particularly those with a large circuit width and fewer permutation gates between layers, the size of the transformed circuit is approximately $O(C)/l$. This characteristic gives our algorithm a distinct advantage. As a result, our algorithm shows promise in scenarios involving lattice problems and in verifying the legality of multiple identical statements simultaneously, such as the verification of multiple AES ciphertexts [41]. Concrete examples will be presented in Section 6.

## 4. NIZKAoK for Lattice Problems

In this section, we present our construction of a NIZKAoK for lattice problems, with a particular emphasis on the LWE problem. LWE is commonly utilized in numerous post-quantum digital signature and key encapsulation algorithms. However, our approach and analysis are also applicable to other variants of lattice problems, such as Short Integer Solution (SIS) problems.

### 4.1. NIZKAoK for LWE

In this subsection, we present a detailed description of the practical implementation of our framework for proving the LWE problem. This implementation can be applied to demonstrate the key generation of FrodoKEM [31], which

is an alternative candidate algorithm for post-quantum cryptography standardization by NIST. The LWE problem involves the sampling of a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times n}$, along with the generation of two $n \times \overline{n}$ secret matrices $\mathbf{S}$ and $\mathbf{E}$, which contain entries sampled from a distribution $\chi$. The public key comprises of two matrices $(\mathbf{A}, \mathbf{B})$, satisfying $\mathbf{B} = \mathbf{AS}+\mathbf{E}$, while the secret key is $(\mathbf{S}, \mathbf{E})$. In this context, the distribution $\chi$ is relatively small compared to modulus $q$, and we sometimes use the notation $[-\eta, \eta]$ to distinguish between integers sampled from $\chi$ and $\mathbb{Z}_q$.

The central concept of the proof is to employ a combination of linear transformation and multiplication gates to establish the two relations inherent in the problem. The first relation, $\mathbf{AS} + \mathbf{E} = \mathbf{B}$, is established through the use of a linear transformation gate to calculate $\mathbf{AS}$, which is then added to $\mathbf{E}$ through an addition gate to obtain $\mathbf{B}$. To establish the second relation, which requires demonstrating that every element of $\mathbf{S}$ and $\mathbf{E}$ lies within the interval $[-\eta, \eta]$, multiple multiplication gates are utilized to prove the validity of each element of $(\mathbf{S}, \mathbf{E})$. For instance, we employ expression $s_{i,j}(s_{i,j}-1)$ to demonstrate that every coefficient $s_{i,j}$ of $\mathbf{S}$ is in the set $\{0, 1\}$. The proof consists of the following steps.

**Prove the linear relation of $\mathbf{B} = \mathbf{AS} + \mathbf{E}$:**

1) For each secret $\mathbf{S} = [\mathbf{s}_1||\mathbf{s}_2||\dots||\mathbf{s}_{\overline{n}}]$ and $\mathbf{E} = [\mathbf{e}_1||\mathbf{e}_2||\dots||\mathbf{e}_{\overline{n}}]$ where each $\mathbf{s}_i, \mathbf{e}_i \in \mathbb{Z}_q^n$, we generates its corresponding packed secret sharing $[\vec{s_1}]_d, [\vec{s_2}]_d, \dots, [\vec{s_{\overline{n}}}]_d$ and $[\vec{e_1}]_d, [\vec{e_2}]_d, \dots, [\vec{e_{\overline{n}}}]_d$.
2) The prover evaluates the preprocessing phase to generate a linear transformation pair $([\vec{r}_i]_d, [\mathbf{A}\vec{r}_i]_d)$ for each $i \in [\overline{n}]$. We let $\mathbf{R} = [\mathbf{r}_1||\mathbf{r}_2||\dots||\mathbf{r}_{\overline{n}}]$.
3) For each $i \in [\overline{n}]$, the prover computes $[\vec{s}_i]_d + [\vec{r}_i]_d$ and reveal it to get $\mathbf{s}_i + \mathbf{r}_i$.
4) The prover computes $\mathbf{A}(\mathbf{s}_1 + \mathbf{r}_1||\mathbf{s}_2 + \mathbf{r}_2||\dots||\mathbf{s}_{\overline{n}} + \mathbf{r}_{\overline{n}})$ and reshares it to get $[\mathbf{A}(\vec{s}_1 + \vec{r}_1)]_d, \dots, [\mathbf{A}(\vec{s}_{\overline{n}} + \vec{r}_{\overline{n}})]_d$.
5) The prover computes $[\mathbf{A}(\vec{s}_1 + \vec{r}_1)]_d - [\mathbf{A}\vec{r}_1]_d, \dots, [\mathbf{A}(\vec{s}_{\overline{n}} + \vec{r}_{\overline{n}})]_d - [\mathbf{A}\vec{r}_{\overline{n}}]_d$ and gets $[\mathbf{A}\vec{S}]_d$.
6) The prover computes $[\mathbf{A}\vec{S}]_d + [\vec{E}]_d$ and reveals it to prove that it equals to $\mathbf{B}$.

**Prove each elements of S and E lie in $[-\eta, \eta]$:**

1) The prover generates a series of different packed secret sharing of $[\vec{i}_j]_d$, $[\vec{i'}_j]_d$ for each $\vec{i}_j, \vec{i'}_j \in \{-\eta, \dots, \eta\}^n$ and computes $([\vec{s}_j]_d - [\vec{i}_j]_d)$ and $([\vec{e}_j]_d - [\vec{i'}_j]_d), j \in [\overline{n}]$.
2) The prover computes each $([\vec{s}_j]_d - [\vec{i}_j]_d)([\vec{s}_j]_d - [\vec{-i}_j]_d)$ in pairs and reshares it to get degree-$d$ secret sharing.
3) The prover multiplies in pairs continually and finally get the degree-$2d$ shares of $\Pi_{i=-\eta}^{\eta}([\vec{s}_j]_d - [\vec{i}_j]_d)$ and $\Pi_{i=-\eta}^{\eta}([\vec{e}_j]_d - [\vec{i}_j]_d)$ for each $j \in [\overline{n}]$.
4) The prover proves that the packed secrets are all zeros by revealing the final degree-$2d$ shares.

## 4.2. NIZKAoK for Modular LWE Problem

This section presents an exposition of our framework's practical realization for proving the Modular Learning with Errors (MLWE) problem. This implementation can be applied to demonstrate the key generation of KyberKEM [32], which is the selected algorithm for post-quantum cryptography standardization by NIST. Let us consider a ring element $R_q = \mathbb{Z}_q[x]/(x^n + 1)$, where $n$ is a power of 2, and two random ring elements $\mathbf{A} \in R_q^{k\times k}, \mathbf{b} \in R_q^k$. The prover's objective is to demonstrate knowledge of two short vectors $\mathbf{s} \in R_q^k$ and $\mathbf{e} \in R_q^k$ such that $\mathbf{As} + \mathbf{e} = \mathbf{b}$, where each coefficient of $\mathbf{s}$ and $\mathbf{e}$ falls in $[-\eta, \eta]$.

One common and straightforward approach is to convert the MLWE problem into an equivalent LWE problem and then establish the security proof for the resulting LWE problem. However, when considering a ring element $a \in R_q$, its conversion into the corresponding LWE problem involves transforming $a$ into $A \in \mathbb{Z}_q^{n\times n}$, with $n$ denoting the dimension of the ring element. This necessitates a minimum of $n$ operations involving multiplication gates applied to the proof. Such a requirement inherently impacts the efficiency of the cryptographic system.

To address these issues, an alternative approach is to directly establish the security of the MLWE problem instead of proving the transformed LWE problem. In this approach, polynomial multiplication on the ring is performed using the traditional Number Theoretic Transformation (NTT), which converts it into the multiplication of corresponding coefficients after the NTT transformation. Let $\mathsf{NTT}()$ and $\mathsf{NTT}^{-1}()$ be the functions for number theory transformation and its inverse, respectively, satisfying $a \times b = \mathsf{NTT}^{-1}(\mathsf{NTT}(a) \circ \mathsf{NTT}(b))$ for each $a, b \in R_q$, where $\circ$ denotes coefficient multiplication. The NTT transformation can be represented as $\mathsf{NTT}(a) = \mathcal{T} \cdot a$, where $\mathcal{T}$ is a matrix with respect to the primitive 256-th root of unity modulo $q$. The proof procedure consists of the following steps.

**Prove the linear relation of $\mathbf{b} = \mathbf{As} + \mathbf{e}$:**

1) For the matrix $\mathbf{A} = \begin{pmatrix} a_{11}\dots a_{1k} \\ \dots \\ a_{k1}\dots a_{kk} \end{pmatrix} \in R_q^{k\times k}$, the prover computes its NTT form $\mathcal{T}\mathbf{A} = \begin{pmatrix} \mathcal{T}a_{11}\dots\mathcal{T}a_{1k} \\ \dots \\ \mathcal{T}a_{k1}\dots\mathcal{T}a_{kk} \end{pmatrix}$.
2) For each secret $\mathbf{s} = [s_1, s_2, \dots, s_k] \in R_q^k$ and $\mathbf{e} = [e_1, e_2, \dots, e_k] \in R_q^k$, we packed the corresponding ring element coefficients and get its packed secret sharing $[\vec{s_1}]_d, [\vec{s_2}]_d, \dots, [\vec{s_k}]_d$, $[\vec{e_1}]_d, [\vec{e_2}]_d, \dots, [\vec{e_k}]_d$.
3) The prover evaluates the preprocessing phase to generate several transformation pairs $([\vec{r}_i]_d, [\vec{\mathcal{T}r}_i]_d), i \in [2k]$.
4) For each $i \in [k]$, the prover computes $[\vec{s}_i]_d + [\vec{r}_i]_d$ and $[\vec{e}_i]_d + [\vec{r}_{i+k}]_d$ and reveals it to get $s_i + r_i$ and $e_i + r_{i+k}$.
5) For each $i \in [k]$, the prover computes $[\vec{\mathcal{T}(s_i + r_i)}]_d - [\mathcal{T}(\vec{r}_i)]_d, [\mathcal{T}(\vec{e}_i + \vec{r}_{i+k})]_d - [\mathcal{T}(\vec{r}_{i+k})]_d$ and get $[\mathcal{T}(\vec{s}_i)]_d$, $[\mathcal{T}(\vec{e}_i)]_d$.
6) For each $j \in [k]$, the prover computes $\mathcal{T}a_{j1} \cdot [\mathcal{T}(\vec{s}_1)]_d + \dots + \mathcal{T}a_{jk} \cdot [\mathcal{T}(\vec{s}_k)]_d + [\mathcal{T}e_j]_d$ and reveals it to prove that it equals to $\mathcal{T}\mathbf{b} = [\mathcal{T}b_1, \mathcal{T}b_2, \dots, \mathcal{T}b_k]$.

**Prove each elements of s and e lie in $[-\eta, \eta]$:**

1) The prover generates a series of different packed secret sharing of $[\vec{i}_j]_d$, $[\vec{i'}_j]_d$ for each $\vec{i}_j, \vec{i'}_j \in \{-\eta, \dots, \eta\}^n$

publicly and computes $([\vec{s}_j]_d - [\vec{i}_j]_d)$ and $([\vec{e}_j]_d - [\vec{i'}_j]_d)$ for each $j \in [k]$.

2) The prover computes each $([\vec{s}_j]_d - [\vec{i}_j]_d)([\vec{s}_j]_d - [\vec{-i}_j]_d)$ in pairs and reshares it to get degree-$d$ secret sharing.
3) The prover multiply in pairs continually and finally gets the degree-$2d$ shares of $\Pi_{i=-\eta}^{\eta}([\vec{s}_j]_d - [\vec{i}_j]_d)$ and $\Pi_{i=-\eta}^{\eta}$ $([\vec{e}_j]_d - [\vec{i}_j]_d)$ for each $j \in [k]$.
4) The prover proves that the packed secrets are all zeros by revealing the final degree-$2d$ shares.

## 5. Application to Knowledge of Secret Key

PKIs issue certificates that associate an entity's public key with identifying information about the entity [42], [43]. To ensure system security against rogue key attacks, it is essential to provide proof of the knowledge of secret keys (KOSK) in PKI systems. Without KOSK proofs, a malicious attacker could potentially impersonate a legitimate user by falsely claiming to possess the corresponding secret key while applying for a certificate from a CA using an arbitrary public key. The PKCS #10 Certificate Signing Request (CSR) [44] is commonly used in many certificate enrollment protocols such as CMP [45], ACME [46], EST [47], and SCEP [48] to allow certificate applicants to prove their possession of secret keys. Typically, CSR contains a fresh signature which can be verified by the public key in the certificate.

Notably, CSRs have gained popularity due to their non-interactive nature, which makes them highly portable. Unlike other certificate enrollment methods, CSRs do not require real-time communication between the entity requesting the certificate and the CA. As a result, CSRs can be validated out of the band, allowing for their transport across different networks. For instance, many web PKI CAs [46] offer simple certificate enrollment workflows that involve pasting a CSR into the CA's web page. This allows for key generation and CSR creation to take place on a production server, while the certificate request is initiated from a workstation outside the production network. Furthermore, even fully automated certificate issuance protocols such as ACME [46] do not require the CSR to contain any protocol state information that would necessitate its generation as part of the certificate issuance exchange.

The current design of CSRs has a significant limitation in that they require a digital signature, which restricts their use to proving possession of digital signature-type keys. This limitation becomes particularly severe in the post-quantum era, where KEM-based authentication is likely to become more popular than signature-based authentication [30]. When the certificate for post-quantum TLS protocols is instantiated using lattice-based KEMs such as Frodo [31] or Kyber [32], there is an urgent need for a post-quantum replacement for CSRs to be used in future PKI systems.

Fortunately, Diet provides practical proof for KOSK for post-quantum KEMs, which offers a viable solution for designing post-quantum PKI systems for KEM TLS. By using the algorithms presented in §4.1 and §4.2, we can easily instantiate NIZKAoK for two lattice-based KEMs, Kyber and FrodoKEM, in Round 3 of the NIST post-quantum cryptography standardization project, and achieve reasonable proof sizes and performance. The benchmark can be found in Table 1 and §6.3.

Note that Guneysu et al. [9] also proposed an efficient algorithm called Combined Proof of Possession (cPOP) to replace current CSRs in the post-quantum era. The cPOP algorithm generates a proof of possession of the secret key simultaneously with the key generation phase. However, this approach cannot be used to authenticate other events in the certificate lifecycle, such as certificate revocation in the PKI Certificate Management Protocol (CMP) [45], because proof generation occurs concurrently with key generation. Moreover, the cPOP algorithm proposed by Guneysu et al. does not strictly ensure that a secret key is perfectly well-formed. It is still possible for a malicious user to register a public key that is maliciously generated. Furthermore, cPOP does not provide a guarantee that the prover has knowledge of the secret key since one cannot extract the secret key even by repeatedly invoking the proving algorithm in the security proof. These weaknesses suggest that the security notion of cPOP is strictly weaker than the standard notion of the proof of knowledge of secret keys. In history, past experiences have shown that weakening the security requirements of the proof of KOSK in PKI systems may lead to serious real-world attacks [8], [49].

## 6. Implementation and Evaluation

In this section, we show the performance of Diet designed for the LWE problem which is used in the somewhat homomorphic encryption schemes and the lattice-based KEM Kyber and Frodo. We implement our protocols in C++ 14 language experimentally. We use NTL 11.5.1[2] and GNU Multiple Precision Arithmetic 6.2.1[3]libraries to implement the polynomial and vector operations in our protocols and choose SHA256 from the OpenSSL library as a hash function. Our performance benchmarks are conducted on a 14-inch Apple MacBook Pro laptop which is powered by the Apple Silicon M1 Pro (3.2GHz) with 10-cores and 16GB of RAM.

### 6.1. Benchmark for Proving Different Gates

We conducted experiments to evaluate the performance of the components for proving each $l$-fold multiplication gate and linear transformation gate. Since the addition gate is almost free in our scheme, the performance evaluation is omitted here. In Table 4, we report the performance of our system with $q = 2^{61}$ and statistical security parameter $\lambda = 128$ with different multiplication gate batch sizes. Our protocol can evaluate 128 multiplication gates by a packed secret sharing in about 21ms with a proof size of 8.32KB and 2048 multiplication gates by a packed secret sharing in

2. https://libntl.org/doc/tour.html
3. https://gmplib.org/

about 1394ms with a proof size of 70.61 KB. In Table 3, we report the performance of our transformation gate with $q = 2^{61}$ and statistical security parameter $\lambda = 128$. Our protocol can evaluate a transformation gate by a packed secret sharing of $l = 128$ in about 14ms with a preprocessing time of 66ms and a packed secret sharing of $l = 2048$ in about 733ms with a preprocessing time of 3398ms.

As shown in Table 3 and 4, there is a trade-off between parameters $t$ and $N$. When the number of opened players $t$ increases, the size of the proof will decrease; conversely, the efficiency of the proof will be faster.

TABLE 3: Batch Linear Transformation Gates

| $t$ | $l$ | $N$ | Pre-Time(ms) | Tran-Time (ms) |
|---|---|---|---|---|
| $t = 100$ | 128 | 1118 | 66 | 14 |
| | 256 | 1737 | 120 | 26 |
| | 512 | 2990 | 228 | 63 |
| | 1024 | 5488 | 910 | 206 |
| | 2048 | 10483 | 3398 | 733 |
| $t = 200$ | 128 | 1028 | 76 | 19 |
| | 256 | 1429 | 133 | 33 |
| | 512 | 2229 | 302 | 75 |
| | 1024 | 3828 | 875 | 229 |
| | 2048 | 7374 | 3116 | 763 |

TABLE 4: Batch Multiplication Gates

| $t$ | $l$ | $N$ | Time(ms) | Size(KB) |
|---|---|---|---|---|
| $t = 100$ | 128 | 1118 | 21 | 8.32 |
| | 256 | 1737 | 40 | 12.19 |
| | 512 | 2990 | 111 | 21.52 |
| | 1024 | 5488 | 361 | 40.12 |
| | 2048 | 10483 | 1394 | 70.61 |
| $t = 200$ | 128 | 1028 | 34 | 7.65 |
| | 256 | 1429 | 60 | 10.64 |
| | 512 | 2229 | 139 | 16.60 |
| | 1024 | 3828 | 434 | 28.50 |
| | 2048 | 7374 | 1647 | 54.91 |

## 6.2. NIZKAoK for LWE

We conducted experiments to evaluate the performance for proving LWE instances of the form $(\mathbf{A}, \mathbf{As} + \mathbf{e})$. Here $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ where $q$ is the modulus. For a better comparison, we choose the same parameter set considered by Baum and Nof [13] which could potentially be used for the validation proof of somewhat homomorphic encryption ciphertexts [26]. Hence we choose $\mathbf{s}$ and $\mathbf{e}$ as binary vectors where $m = n = 2048$ and $q = 2^{61}$.

We provided a comparison of different NIZK techniques [13], [23], [24], [27], [29], [50] for proof size and proving time in the third and fourth columns in Table 1. The parameters of the packed secret sharing we chose are set as $(N, t, l) = (2140, 200, 512)$ which implies packing $s$ and $e$ into eight packed secret sharing polynomials. We set $(k, v) = (23, 4)$ to make the soundness error of preprocessing less than $2^{-128}$ . The size of our online proof is 245.88KB, while the size of a preprocessing is 350KB. Our online proof time is 0.528s, with a preprocessing time of 5.596s. Compared to other schemes, our solution offers significant advantages in terms of both proof size and proof time, when considering both factors in combination.

Interestingly, if one simultaneously proves multiple instances of the LWE problem, the efficiency can be further improved. This observation makes our scheme potentially useful for the validation proof of a large number of somewhat homomorphic encryption ciphertexts, which may be required for the preprocessing phase of MPC protocols such as SPDZ [51]. This observation comes from that the cut-and-choose proof component in the preprocessing phase could be set up for an arbitrary number of same type of linear transformation gates. Note that the polynomial of vectors (3.1) set during the preprocessing phase is $k + v$ degree, while the size of the cut-and-choose component is only proportion to $k$ but this polynomial can be used to generate random vector pairs for $v$ linear transformation gates. Hence the amortization efficiency will get better if $v$ is increased. This observation is demonstrated by experiments shown in Figure. 3. Figure 3(a) shows that the total proof size varies as we prove $\{5, 10, 100, 1000\}$ instances at a time. The proof sizes of the other four schemes are taken from [9]. Figure 3(c) shows the amortized proof size per proof as we prove different numbers of instances at a time, while Figure 3(b) shows the amortized proof time per proof for the same number of instances. Specifically, we run our protocol for $\{5, 10, 100, 1000\}$ instances and calculate the average proof size and time per proof.

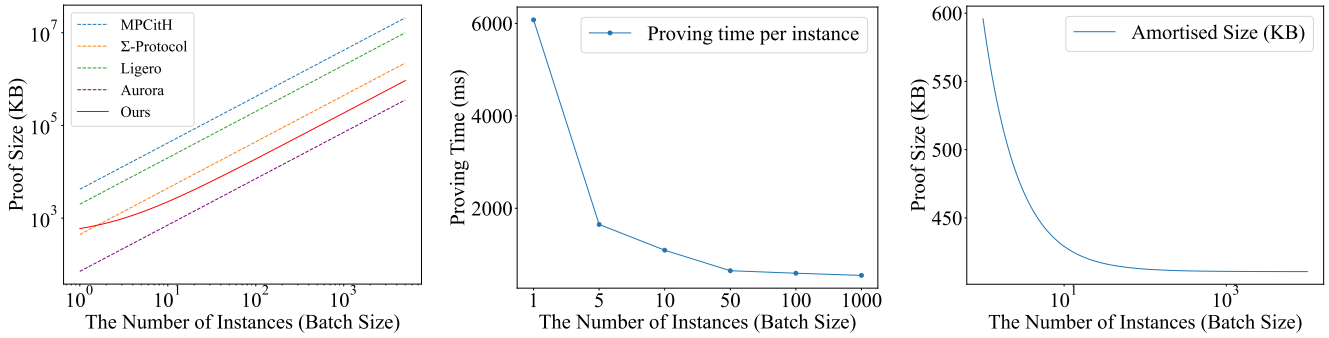
## 6.3. KOSK for Lattice Based KEMs

We conducted experiments to evaluate the performance for proving the knowledge of secret keys for two lattice-based KEMs Frodo640 and Kyber512. The proof size and proof time are shown in Table 1.

*Frodo640*: Given the parameters $(N, t, l)$ for packed secret sharing used in MPCitH paradigm, we have set $(k, v) = (90, 8)$ to make the soundness error of preprocessing less than $2^{-128}$. For both Frodo640 and our NIZKAOK system, we have chosen $n = 640$, $\overline{n} = 8$, $q = 2^{15}$, and $(N, t, l) = (2500, 250, 640)$.

In the study of proof sizes in existing constructions, Baum [13] illustrated the conventional MPCitH methodology's implementation for Frodo key generation with a proof size of 8.42 megabytes (MB). By employing the packed secret sharing technique, our work reduces the proof size from over 8.42 MB to 473.51 kilobytes (KB), with a preprocessing overhead of 123.36 KB. Thus, we achieve at least 80% reduction in proof size compared to the previous approach. Additionally, we have developed an efficient implementation. Our proof time is 12.81s, with a pre-processing time of 6.5s and a verification time of 7.3s. Specifically, the time taken to prove the linear relationship is 0.2s, and the time taken to prove the secret vector range is 12.61s. As Table 1 illustrates, we are the first to provide a concrete implementation of a NIZKAoK system for Frodo640.

*Kyber512*: Given the parameters $(N, t, l)$ for packed secret sharing used in MPCitH paradigm, we have set $(k, v) =$




(a) The proof size varies with the number of instances

(b) The amortized time varies with the number of instances

(c) The amortized proof size varies with the number of instances

Figure 3: Amortized Efficiency

$(70, 4)$ to make the soundness error of preprocessing less than $2^{-128}$. For both Kyber512 and our NIZKAoK system, we have chosen modular rank $k = 2$ and $\eta = 2, q = 3329$, and $(N, t, l) = (1454, 150, 256)$.

Our proposed approach achieves a proof size of 83.65 KB for online computation, with an additional preprocessing overhead of 152.02 KB. To demonstrate the practicality and efficiency of our approach, we provide a concrete implementation that achieves high performance, with a proof time of only 0.68 seconds, a preprocessing time of 0.81 seconds, and a verification time of 0.84 seconds. Specifically, the time required to prove the linear relationship is 0.07 seconds, while the time required to prove the range is 0.56 seconds.

While previous works [20], [52] have achieved better proof sizes, the concrete efficiency of their approaches has not been tested via implementation. As pointed out by Feneuil et al. [15], their rejection sampling rate is 0.85 under their parameters for the lattice-based zero-knowledge (ZK) scheme. This suggests that the repetition time for proof generation is likely to exceed 6, which may affect the efficiency of their approach.

## Acknowledgments

This work was supported by the National Key R&D Program of China (No.2021YFB3100100) and the National Natural Science Foundation of China Key Program (No.92270204).

## References

[1] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof systems," *SIAM Journal on computing*, vol. 18, no. 1, pp. 186–208, 1989.

[2] J. Kilian, "A note on efficient zero-knowledge proofs and arguments," in *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, 1992, pp. 723–732.

[3] A. De Santis and G. Persiano, "Zero-knowledge proofs of knowledge without interaction," in *Proceedings., 33rd Annual Symposium on Foundations of Computer Science*. IEEE Computer Society, 1992, pp. 427–436.

[4] M. Blum, P. Feldman, and S. Micali, "Non-interactive zero-knowledge and its applications," in *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, 2019, pp. 329–349.

[5] U. Feige, D. Lapidot, and A. Shamir, "Multiple non-interactive zero knowledge proofs based on a single random string," in *Proceedings [1990] 31st Annual Symposium on Foundations of Computer Science*. IEEE, 1990, pp. 308–317.

[6] A. D. Santis, S. Micali, and G. Persiano, "Non-interactive zero-knowledge proof systems," in *Conference on the Theory and Application of Cryptographic Techniques*. Springer, 1987, pp. 52–72.

[7] T. Ristenpart and S. Yilek, "The power of proofs-of-possession: Securing multiparty signatures against rogue-key attacks," in *Advances in Cryptology-EUROCRYPT 2007: 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Barcelona, Spain, May 20-24, 2007. Proceedings 26*. Springer, 2007, pp. 228–245.

[8] M. Bellare and G. Neven, "Multi-signatures in the plain public-key model and a general forking lemma," in *Proceedings of the 13th ACM conference on Computer and communications security*, 2006, pp. 390–399.

[9] T. Güneysu, P. Hodges, G. Land, M. Ounsworth, D. Stebila, and G. Zaverucha, "Proof-of-possession for kem certificates using verifiable generation," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, pp. 1337–1351.

[10] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai, "Zero-knowledge from secure multiparty computation," in *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, 2007, pp. 21–30.

[11] J. Katz, V. Kolesnikov, and X. Wang, "Improved non-interactive zero knowledge with applications to post-quantum signatures," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 525–537.

[12] C. D. de Saint Guilhem, L. De Meyer, E. Orsini, and N. P. Smart, "Bbq: Using aes in picnic signatures," in *Selected Areas in Cryptography–SAC 2019: 26th International Conference, Waterloo, ON, Canada, August 12–16, 2019, Revised Selected Papers 26*. Springer, 2020, pp. 669–692.

[13] C. Baum and A. Nof, "Concretely-efficient zero-knowledge arguments for arithmetic circuits and their application to lattice-based cryptography." *Public Key Cryptography (1)*, vol. 12110, pp. 495–526, 2020.

[14] C. Delpech de Saint Guilhem, E. Orsini, and T. Tanguy, "Limbo: Efficient zero-knowledge mpcith-based arguments," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 3022–3036.

[15] T. Feneuil, J. Maire, M. Rivain, and D. Vergnaud, "Zero-knowledge protocols for the subset sum problem from mpc-in-the-head with rejection," in *Advances in Cryptology–ASIACRYPT 2022: 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5–9, 2022, Proceedings, Part II*. Springer, 2023, pp. 371–402.

[16] T. Feneuil and M. Rivain, "Threshold linear secret sharing to the rescue of mpc-in-the-head," *Cryptology ePrint Archive*, 2022.

[17] I. Damgård, Y. Ishai, M. Krøigaard, J. B. Nielsen, and A. Smith, "Scalable multiparty computation with nearly optimal work and resilience," in *Annual International Cryptology Conference*. Springer, 2008, pp. 241–261.

[18] I. Damgård, Y. Ishai, and M. Krøigaard, "Perfectly secure multiparty computation and the computational overhead of cryptography," in *Annual international conference on the theory and applications of cryptographic techniques*. Springer, 2010, pp. 445–465.

[19] V. Lyubashevsky, N. K. Nguyen, and G. Seiler, "Practical lattice-based zero-knowledge proofs for integer relations," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 1051–1070.

[20] V. Lyubashevsky, N. K. Nguyen, and G. Seiler, "Shorter lattice-based zero-knowledge proofs via one-time commitments," in *IACR International Conference on Public-Key Cryptography*. Springer, 2021, pp. 215–241.

[21] J. Bootle, V. Lyubashevsky, N. K. Nguyen, and G. Seiler, "More efficient amortization of exact zero-knowledge proofs for lwe," in *European Symposium on Research in Computer Security*. Springer, 2021, pp. 608–627.

[22] N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer, "From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again," in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, 2012, pp. 326–349.

[23] E. Ben-Sasson, A. Chiesa, M. Riabzev, N. Spooner, M. Virza, and N. P. Ward, "Aurora: Transparent succinct arguments for r1cs," in *Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part I 38*. Springer, 2019, pp. 103–128.

[24] W. Beullens, "Sigma protocols for mq, pkp and sis, and fishy signature schemes," in *Advances in Cryptology–EUROCRYPT 2020: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10–14, 2020, Proceedings, Part III*. Springer, 2020, pp. 183–211.

[25] C. Baum, I. Damgård, V. Lyubashevsky, S. Oechsner, and C. Peikert, "More efficient commitments from structured lattice assumptions," in *International Conference on Security and Cryptography for Networks*. Springer, 2018, pp. 368–385.

[26] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(leveled) fully homomorphic encryption without bootstrapping," *ACM Transactions on Computation Theory (TOCT)*, vol. 6, no. 3, pp. 1–36, 2014.

[27] B. Libert, S. Ling, K. Nguyen, and H. Wang, "Lattice-based zero-knowledge arguments for integer relations," in *Advances in Cryptology–CRYPTO 2018: 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19–23, 2018, Proceedings, Part II*. Springer, 2018, pp. 700–732.

[28] S. Ames, C. Hazay, Y. Ishai, and M. Venkitasubramaniam, "Ligero: Lightweight sublinear arguments without a trusted setup," in *Proceedings of the 2017 acm sigsac conference on computer and communications security*, 2017, pp. 2087–2104.

[29] V. Lyubashevsky, N. K. Nguyen, and M. Plancon, "Efficient lattice-based blind signatures via gaussian one-time signatures," in *Public-Key Cryptography–PKC 2022: 25th IACR International Conference on Practice and Theory of Public-Key Cryptography, Virtual Event, March 8–11, 2022, Proceedings, Part II*. Springer, 2022, pp. 498–527.

[30] P. Schwabe, D. Stebila, and T. Wiggers, "Post-quantum tls without handshake signatures," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 1461–1480.

[31] E. Alkim, J. W. Bos, L. Ducas, P. Longa, I. Mironov, M. Naehrig, V. Nikolaenko, C. Peikert, A. Raghunathan, D. Stebila *et al.*, "Frodokem learning with errors key encapsulation," *NIST PQC standardization: Round*, vol. 3, 2020.

[32] R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, "Crystals-kyber algorithm specifications and supporting documentation," *NIST PQC Round*, vol. 2, no. 4, pp. 1–43, 2019.

[33] A. Wigderson, M. Or, and S. Goldwasser, "Completeness theorems for noncryptographic fault-tolerant distributed computations," in *Proceedings of the 20th Annual Symposium on the Theory of Computing (STOC'88)*, 1988, pp. 1–10.

[34] I. Damgård and J. B. Nielsen, "Scalable and unconditionally secure multiparty computation," in *Advances in Cryptology-CRYPTO 2007: 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007. Proceedings 27*. Springer, 2007, pp. 572–590.

[35] D. Beaver, "Efficient multiparty protocols using circuit randomization," in *Annual International Cryptology Conference*. Springer, 1991, pp. 420–432.

[36] M. Franklin and M. Yung, "Communication complexity of secure computation," in *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, 1992, pp. 699–710.

[37] L. Braun, C. D. de Saint Guilhem, R. Jadoul, E. Orsini, N. P. Smart, and T. Tanguy, "Zk-for-z2k: Mpc-in-the-head zero-knowledge proofs for $\mathbb{Z}_{2^k}$," Cryptology ePrint Archive, Paper 2023/1057, 2023, https://eprint.iacr.org/2023/1057. [Online]. Available: https://eprint.iacr.org/2023/1057

[38] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[39] J. T. Schwartz, "Fast probabilistic algorithms for verification of polynomial identities," *Journal of the ACM (JACM)*, vol. 27, no. 4, pp. 701–717, 1980.

[40] V. E. Beneš, "Optimal rearrangeable multistage connecting networks," *Bell system technical journal*, vol. 43, no. 4, pp. 1641–1656, 1964.

[41] F. Zhang, D. Maram, H. Malvai, S. Goldfeder, and A. Juels, "Deco: Liberating web data using decentralized oracles for tls," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 1919–1938.

[42] C. Adams and S. Lloyd, *Understanding public-key infrastructure: concepts, standards, and deployment considerations*. Sams Publishing, 1999.

[43] S. Chokhani, W. Ford, R. Sabett, C. Merrill, and S. Wu, "Rfc3647: Internet x. 509 public key infrastructure certificate policy and certification practices framework," 2003.

[44] M. Nystrom and B. Kaliski, "Pkcs# 10: Certification request syntax specification version 1.7," Tech. Rep., 2000.

[45] C. Adams, S. Farrell, T. Kause, and T. Mononen, "Rfc 4210: Internet x. 509 public key infrastructure certificate management protocol (cmp)," 2005.

[46] R. Barnes, J. Hoffman-Andrews, D. McCarney, and J. Kasten, "Rfc 8555: Automatic certificate management environment (acme)," 2019.

[47] M. Pritikin, P. Yee, and D. Harkins, "Rfc 7030: Enrollment over secure transport," 2013.

[48] P. Gutmann, "Rfc 8894: Simple certificate enrolment protocol," 2020.

[49] N. Asokan, V. Niemi, and P. Laitinen, "On the usefulness of proof-of-possession," in *Proceedings of the 2nd Annual PKI Research Workshop*, 2003, pp. 122–127.

[50] F. Araujo, K. W. Hamlen, S. Biedermann, and S. Katzenbeisser, "From patches to honey-patches: Lightweight attacker misdirection, deception, and disinformation," in *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, 2014, pp. 942–953.

[51] I. Damgård, V. Pastro, N. Smart, and S. Zakarias, "Multiparty computation from somewhat homomorphic encryption," in *Annual Cryptology Conference*. Springer, 2012, pp. 643–662.

[52] V. Lyubashevsky, N. K. Nguyen, and M. Plançon, "Lattice-based zero-knowledge proofs and applications: shorter, simpler, and more general," in *Advances in Cryptology–CRYPTO 2022: 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15–18, 2022, Proceedings, Part II*. Springer, 2022, pp. 71–101.

# Appendix A.
# Security Proof of Theorem 1

*Proof.* We will address completeness, soundness, and zero knowledge individually. Our proving technique is inspired by Feneuil and Rivain [16].

**Completeness:** The completeness holds from the completeness property of the underlying MPC protocol. A prover $\mathcal{P}$ who knows a witness $\omega$ such that $C(x,\omega) = 1$ and who follows the steps of the protocol always succeeds in convincing the verifier $\mathcal{V}$.

**Knowledge Soundness:** Suppose that there is an efficient prover $\mathcal{P}^*$ that, on input $x$, convinces the honest verifier $\mathcal{V}$ to accept, then, there exists an efficient probabilistic extraction algorithm $E_0$ that, given rewindable black-box access to $\mathcal{P}^*$, outputs a witness $\omega$ satisfying $C(x,\omega) = 1$.

**When restraining to only bad witnesses**: Given all the transcripts, we have a unique hash commitment $h_{com}$ in the transcript. This hash commitment uniquely defines the shares of the witness $\omega$ (by assumption on the absence of hash/commitment collisions). In the following, we shall denote $\omega^J$ the witness corresponding to the shares $(\omega_i)_{i\in[J]}$. Since the multiplication gate requires at most $2d+1$ values to determine a secret, $|J| = 2(t+l)+1$. We have a total of $\binom{N}{2(t+l)+1}$ possibly distinct witnesses $\omega^J$. We shall say that $\omega^J$ is a good witness whenever $(x,\omega^J) \in R$, otherwise we call $\omega^J$ a bad witness.

For any malicious prover $P^*$, the hash commitments $H_{com}$ uniquely define the transcript $V_i$. The hash commitments $H_C$ also uniquely define the challenge value $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_{k+v})$. we shall denote by $H$ the set of honest parties, that is, the set of the parties for which the committed transcript $V_i$ is consistent with the committed input shares $\omega_i$. More formally, $H = \{i : V_i = \phi(\omega_i, \alpha)\}$ where $\phi$ is the function corresponding to the circuit C on which the MPC protocol operates.

We further denote $Y$ the random variable that corresponds to the number of honest parties, that is, $Y = |H|$. We stress that $\{V_i\}_{i\in[N]}, H, Y$ depend on the randomness of the (malicious) prover generated random vector $(\vec{f_j})_{j\in[0,k+v]}$ and the hash function $H_C$. For every $i \in [N]$, we shall further denote $\hat{V}_i$ the transcript obtained through an honest computation from the committed input shares, that is: $\hat{V}_i = \phi(\omega_i, \alpha)$. We stress that $\hat{V}_i$ may not be equal to $V_i$. We actually have $i \in H$ if and only if $\hat{V}_i = V_i$.

In the following, we shall say that witness $\omega^J$ gives rise to a wrong result(for a bad witness $\omega$ outputs $C(x,\omega) = 1$) in the MPC protocol, and denote the probability event $E_J$ whenever $C(\hat{V}_i) = 1$ where $\hat{V}_i$ are the plain values which contain the witness share and randomness and broadcast values used in the MPC protocol. By definition of the MPC protocol and the Schwartz-Zippel lemma, we have: $\forall J$ s.t. $|J| = 2(l+t)+1, \Pr[E_J] \leq \left(\frac{k+v}{||F||}\right)^k$.

For the first step of the proof, we shall consider a subset D of parties, i.e. $D \subset \{1,2,...,N\}$, and we denote $N' = |D|$ *and* $\mathcal{J} = \{J \subset D : |J| = 2(l+t)+1\}$. We will show that, if $\{\omega^J\}_{J\in\mathcal{J}}$ are all bad witnesses, then the probability $\Pr[succ_{P^*}]$ is upper bounded by $\Pr[succ_{P^*}] \leq \frac{\binom{N}{d}}{\binom{N'}{d}} \cdot \epsilon$ where $\epsilon$ is the soundness error defined in the theorem statement, which is $\epsilon := \frac{\binom{2(t+l)}{t}}{\binom{N}{t}} + \left(\frac{k+v}{||F||}\right)^k \cdot \frac{t}{2(t+l)+1} \cdot \binom{N-t}{t+2l+1}$.

For $J \in \mathcal{J}$ and $b \in \{0,1\}$, let us introduce the notation:

$$A_J^b = \begin{cases} E_J, & if\ b = 1, \\ \overline{E}_J, & if\ b = 0. \end{cases}$$

Let $x = (x_J)_{J\in\mathcal{J}}$ and let $y \in \{0,\ldots,N\}$. Let us assume that $succ_{P^*}$, $Y = y$ and $\{A_J^{x_J}\}_{J\in\mathcal{J}}$ jointly occur. Since $succ_{P^*}$ occurs, we have $C(v_i) = 1$. Then for each set $J \in \mathcal{J}$ ($w^J$ is a bad witness) such that $J \subset H$ (the parties in J are honest), we have $V_i = \hat{V}_i$ for every $i \in J$, which implies $C(\hat{v}_i) = C(v_i) = 1$. Namely, a bad witness $\omega^J$ gives rise to a wrong result(for a bad witness $\omega$ outputs $C(x,\omega) = 1$) in the MPC protocol that necessarily occurs for $\omega^J$,i.e. $x_J = 1$, whenever $J \in \mathcal{J}$ with $J \subset H$. Thus $wt_H(x) \geq \sum_{J\in\mathcal{J}:J\subset H} x_J = \binom{y}{2(t+l)+1}$ where $wt_H(x)$ is the number of a bad witness $\omega^J$ gives rise to a proof that can be verified. By defining $y_{max} := max\{y : wt_H(x) \geq \binom{y}{2(t+l)+1}\}$ we get that $\Pr[succ_{P^*}, Y = y | \{A_J^{x_J}\}_{J\in\mathcal{J}}] = 0$, if $y > y_{max}$ and $\Pr[succ_{P^*} | Y = y] = \sum_{y=0}^{y_{max}} \Pr[succ_{P^*}, Y = y | \{A_J^{x_J}\}_{J\in\mathcal{J}}]$.

The only way for the transcript to be successful is that the set $\mathcal{I}$ of challenged opened parties only contains honest parties, i.e. $\mathcal{I} \subset H$. Thus, $\Pr'[succ_{P^*} | \{A_J^{x_J}\}_{J\in\mathcal{J}}, Y = y] \leq \Pr[\mathcal{I} \in H | \mathcal{I} \in D, Y = y] = \frac{\binom{y}{t}}{\binom{N'}{t}}$. We deduce $\Pr'[succ_{P^*} | \{A_J^{x_J}\}_{J\in\mathcal{J}}, Y = y] \leq \sum_{y=0}^{y_{max}} \frac{\binom{y}{t}}{\binom{N'}{t}} \cdot \Pr'[Y = y | \{A_J^{x_J}\}_{J\in\mathcal{J}}] \leq \frac{\binom{y_{max}}{t}}{\binom{N'}{t}}$.

Since $wt_H(x)$ is a non-negative integer, let us consider three cases:

**Case1:** $y_{max} = 2(t+l)$, it means that $wt_H(x) = 0$, then

$$\Pr[succ_{P^*} | \{A_J^{x_J}\}_{J\in\mathcal{J}}] \leq \frac{\binom{y_{max}}{t}}{\binom{N'}{t}} = \frac{\binom{2(t+l)}{t}}{\binom{N'}{t}} = \frac{wt_H(x) \cdot \binom{2(t+l)}{t-1} + \binom{2(t+l)}{t}}{\binom{N'}{t}}$$

**Case2:** $y_{max} = 2(t+l)+1$, it means $wt_H(x) \geq 1$, then

$$\Pr[succ_{P^*}|\{A_J^{x_J}\}_{J\in\mathcal{J}}] \leq \frac{\binom{y_{max}}{t}}{\binom{N'}{t}} = \frac{\binom{2(t+l)+1}{t}}{\binom{N'}{t}} \leq \frac{wt_H(x)\cdot\binom{2(t+l)}{t-1}+\binom{2(t+l)}{t}}{\binom{N'}{t}}$$

Since $\binom{2(t+l)+1}{t} = \binom{2(t+l)}{t-1}+\binom{2(t+l)}{t}$.
**Case3:** $y_{max} \geq 2(t+l)+2$, then

$$\Pr[succ_{P^*}|\{A_J^{x_J}\}_{J\in\mathcal{J}}] \leq \frac{\binom{y_{max}}{t}}{\binom{N'}{t}} = \frac{\binom{2(t+l)+1}{t}}{\binom{y_{max}-t}{t+2l+1}}\cdot\frac{\binom{y_{max}}{2(t+l)+1}}{\binom{N'}{t}} \leq \frac{\binom{2(t+l)+1}{t}}{\binom{y_{max}-t}{t+2l+1}}\cdot\frac{wt_H(x)}{\binom{N'}{t}} \leq \frac{\binom{2(t+l)+1}{t}}{t+2l+2}\cdot\frac{wt_H(x)}{\binom{N'}{t}} \leq \frac{wt_H(x)\cdot\binom{2(t+l)}{t-1}+\binom{2(t+l)}{t}}{\binom{N'}{t}}$$

The last inequality can be prove using Lemma 5 of [16] since $\binom{2(l+t)+1}{t} = \binom{2(t+l)}{t-1}+\binom{2(t+l)}{t}$.

In any case, we have $\Pr[succ_{P^*}|\{A_J^{x_J}\}_{J\in\mathcal{J}}] \leq \frac{wt_H(x)\cdot\binom{2(t+l)}{t-1}+\binom{2(t+l)}{t}}{\binom{N'}{t}}$ for any $x \in \{0,1\}^{|\mathcal{J}|}$. According to Lemma 4 of [16], we have

$$\Pr[succ_{P^*}] = \sum_{x\in\{0,1\}^J}\Pr[succ_{P^*}|\{A_J^{x_J}\}_{J\in\mathcal{J}}]\cdot\Pr[\{A_J^{x_J}\}_{J\in\mathcal{J}}]$$

$$\leq \sum_{x\in\{0,1\}^J}\frac{wt_H(x)\cdot\binom{2(t+l)}{t-1}+\binom{2(t+l)}{t}}{\binom{N'}{t}}\cdot\Pr[\{A_J^{x_J}\}_{J\in\mathcal{J}}]$$

$$= \frac{\binom{2(t+l)}{t}}{\binom{N'}{t}} + \frac{\binom{2(t+l)}{t-1}}{\binom{N'}{t}}\cdot\sum_{x\in\{0,1\}^J} wt_H(x)\cdot\Pr[\{A_J^{x_J}\}_{J\in\mathcal{J}}]$$

$$\leq \frac{\binom{2(t+l)}{t}}{\binom{N'}{t}} + \frac{\binom{2(t+l)}{t-1}}{\binom{N'}{t}}\cdot\sum_{J\in\mathcal{J}}\Pr[E_J]$$

$$\leq \frac{\binom{2(t+l)}{t}}{\binom{N'}{t}} + \frac{\binom{2(t+l)}{t-1}}{\binom{N'}{t}}\cdot|\mathcal{J}|\left(\frac{k+v}{||F||}\right)^k = \frac{\binom{N}{t}}{\binom{N'}{t}}\cdot$$

$$\left(\frac{\binom{2(t+l)}{t}}{\binom{N}{t}} + \frac{\binom{2(t+l)}{t-1}}{\binom{N}{t}}\cdot\binom{N'}{2(t+l)+1}\left(\frac{k+v}{||F||}\right)^k\right) \leq \frac{\binom{N}{t}}{\binom{N'}{t}}\cdot\epsilon$$

Since

$$\frac{\binom{2(l+t)}{t-1}\binom{N}{2(t+l)+1}}{\binom{N}{t}} = \frac{N!\cdot(2(l+t))!\cdot t!\cdot(N-t)!}{N!\cdot(2(l+t)+1)!\cdot(t-1)!(N-2t-2l-1)!(t+2l+1)!} = \frac{t}{2(t+l)+1}\binom{N-t}{t+2l+1}$$

**Building of the extractor** We now show how to build an extractor which outputs a witness $\omega$ satisfying $(x,\omega) \in R$ (if not a hash or commitment collision) when giving rewindable black-box access to a malicious prover $P^*$ which produces successful transcripts with a probability $\hat{\epsilon} > \epsilon$.

Let us fix an arbitrary value $\alpha \in \{0,1\}$ such that $(1-\alpha)\hat{\epsilon} > \epsilon$ (such $\alpha$ exists since $\hat{\epsilon} > \epsilon$). Below we denote by $R_h$ the randomness of $P^*$ which is used to generate the initial randomness $\{f_i\}_{i\in[k+v+1]}$ and the secret share of $\omega$, and we denote $r_h$ a possible realization of $R_h$. We will say that $r_h$ is good if it is such that $\Pr[succ_{P^*}|R_h = r_h] \geq (1-\alpha)\hat{\epsilon}$. By the Splitting Lemma 3 of [16] we have $\Pr[R_h\ good|succ_{P^*}] \geq \alpha$.

Our extractor first runs the $P^*$ with honest verifier requests until obtaining a successful transcript $T_0$ by running. If this $T_0$ corresponds to a good $r_h$, then we can obtain further successful transcripts with "high" probability (i.e. probability greater than $(1-\alpha)\cdot\hat{\epsilon}$) by rewinding the protocol just after the commitment of $H_C$. Based on the assumption that $r_h$ is good, a sub-extractor $E_0$ will build a list of successful transcripts $\mathcal{T}$, all with same initial commitment. We denote $P(T)$ the set of the parties which have been open in at least one transcript of $\mathcal{T}$ , i.e. $P(T) := \cup_{T\in\mathcal{T}} I_T$ where $I_T$ is the set of opened parties of the transcript T.

For a certain number $N_1$ of iterations, the sub-extractor $E_0$ tries to feed the list $\mathcal{T}$ until there exist a good witness among the open input shares. We formally describe the sub-extractor routine in the following pseudocode:

- $\mathcal{T} = T_0$.
- Do $N_1$ times:
- Run $p^*$ with honest V and same $r_h$ as $T_0$ to get transcript $T$.
- If T is a successful transcript,
- $\mathcal{T} \leftarrow \mathcal{T} \cup \{T\}$.
- If $\mathcal{T}$ contains a good witness $\omega$, Return $\omega$.
- Return $\phi$.

Let us evaluate the probability that the stop condition is reached in a given number of iteration $N_1$. Consider a loop iteration in $E_0$ at the beginning of which we have a list $\mathcal{T}$ of successful transcripts (which does not contain a good witness since the stop condition has not been reached) and a transcript T sampled at Step 3. We denote $Z$ the event that a new party is open (a party which is not in $P(\mathcal{T})$) in the transcript T. This event is defined with respect to the randomness of the challenges in T.

Let us lower bound the probability to have a successful transcript T and the event $Z$ occurring in the presence of a good $R_h$: $P_G := \Pr[succ_{p^*} \cap Z|R_h\ good]$. We have:

$$\begin{aligned} P_G &= \Pr[succ_{P^*}|R_h\ good] - \Pr[succ_{P^*}\cap\overline{Z}|R_h\ good] \\ &= \Pr[succ_{P^*}|R_h\ good] - \\ &\quad \Pr[succ_{P^*}|R_h good,\overline{Z}]\cdot\Pr[\overline{Z}|R_h\ good] \\ &\geq (1-\alpha)\hat{\epsilon} - \Pr[succ_{P^*}|R_h good,\overline{Z}]\cdot\Pr[\overline{Z}|R_h\ good] \end{aligned}$$

where the last inequality holds by $\Pr[succ_{P^*}|R_h = r_h] \geq (1-\alpha)\hat{\epsilon}$. The probability that a new party is not opened corresponds to the probability that the set $I$ of opened parties is a subset of $P(\mathcal{T})$, i.e. $\Pr[\overline{Z}|R_h\ good] = \Pr[\overline{Z}] = \frac{\binom{|P(\mathcal{T})|}{t}}{\binom{N}{t}}$.

The success probability knowing that no new party is open corresponds to the success probability when restricting to the $|P(\mathcal{T})|$ parties which have been already open. By

assumption ($\mathcal{T}$ does not contain a good witness), the shares of those parties only correspond to bad witnesses. Thus, this probability can be upper bounded using $\Pr[succ_{P^*}] \leq \frac{\binom{N}{t}}{\binom{N'}{t}} \cdot \epsilon$ with $N' = |P(\mathcal{T})|$ and $\Pr[succ_{P^*}|R_h \ good, \overline{Z}] \leq \frac{\binom{N}{t}}{\binom{|P(\mathcal{T})|}{t}} \cdot \epsilon$. Thus, we get $P_G \geq (1-\alpha) \cdot \hat{\epsilon} - \epsilon$.

In the presence of a good $R_h$, the probability of the event $succ_{P^*} \cap Z$ (i.e. getting a successful transcript T which opens a new party) is lower bounded by $(1-\alpha) \cdot \hat{\epsilon} - \epsilon > 0$. Moreover, the event $succ_{P^*} \cap Z$ can occur at most $N-t$ times, because $T_0$ already opens $t$ parties and there are $N$ parties in total. We deduce that after $N-t$ occurrences of $succ_{P^*} \cap Z$, the list $\mathcal{T}$ contains a good witness.

Let us now define $N_1 = \frac{4(N-t)}{p_0}$ $with$ $p_0 := (1-\alpha)\hat{\epsilon} - \epsilon$ and let $X \sim \mathcal{B}(N_1, p_0)$ a binomial distributed random variable with parameters $(N_1, p_0)$. The probability that $E_0$ reaches the stop condition and returns a (good) witness for a successful transcript $T_0$ with good $R_h$ satisfies:

$$
\begin{aligned}
&\Pr[E_0(T_0) \neq \phi | succ_{P^*}^{T_0} \cap R_h \ good] \geq \Pr[X > N-t] \\
&= \Pr[\frac{X}{N_1} - p_0 > \frac{N-t}{N_1} - p_0] \\
&= 1 - \Pr[\frac{X}{N_1} - p_0 \leq \frac{N-t}{N_1} - p_0] \\
&= 1 - \Pr[\frac{X}{N_1} - p_0 \leq -\frac{3}{4}p_0] \\
&\geq 1 - \Pr[|\frac{X}{N_1} - p_0| \geq \frac{3}{4}p_0] \geq 1 - \frac{p_0 \cdot (1-p_0)}{N_1 \cdot P_0^2 \cdot (\frac{3}{4})^2} \\
&= 1 - \frac{16}{9} \cdot \frac{1-p_0}{4(N-t)} = 1 - \frac{4}{9} \cdot \frac{1-p_0}{N-t} \geq 1 - \frac{4}{9} \geq \frac{1}{2}
\end{aligned}
$$

The inequality holds from the Bienayḿe-Techbychev inequality. Thus, using $N_1 = \frac{4(N-t)}{p_0}$ , the probability to reach stop condition assuming a good $R_h$ is at least 1/2. Without assumption on $R_h$, the probability to reach stop condition satisfies:$\Pr[E_0(T_0) \neq \phi | succ_{P^*}^{T_0}] \geq \Pr[R_h \ good | succ_{P^*}^{T_0}] \cdot \Pr[E_0(T_0) \neq \phi | succ_{P^*}^{T_0} \cap R_h \ good] \geq \frac{\alpha}{2}$.

Let us now describe the complete extractor procedure:

- Repeat $+\infty$ times:
- Run $p^*$ with honest V to get transcript $T_0$.
- If $T_0$ is not a successful transcript, go to next iteration.
- Call $E_0$ on $T_0$ to get list of transcripts $\mathcal{T}$.
- If $\mathcal{T} \neq \phi$, return $\mathcal{T}$.

Let $N_C$ denotes the number of calls to $P^*$ made by the extractor before ending. While entering a new iteration:

- The extractor makes one call to $P^*$ to obtain $T_0$.
- If $T_0$ is not successful, which occurs with probability $(1 - \Pr[succ_{P^*}^{T_0}])$
- The extractor continues to the next iteration and makes an average of $E[N_C]$ calls to $P^*$,
- If $T_0$ is successful, which occurs with probability $\Pr[succ_{P^*}^{T_0}]$
- The extractor makes at most $N_1$ calls to $P^*$ in the loop of $E_0$,
- Then $E_0$ returns an empty list, which occurs with probability $\Pr[E_0(T_0) = \phi | succ_{P^*}^{T_0}]$ , the extractor continues to the next iteration and makes an average of $E[N_C]$ calls to $P^*$,
- Otherwise, if $E_0(T_0)$ returns a non-empty list, the extractor stops and no more calls to $P^*$.

The mean number of calls to $P^*$ hence satisfies: $E[N_C] = 1 + (1 - \Pr[succ_{P^*}^{T_0}]) \cdot E[N_C] + \Pr[succ_{P^*}^{T_0}] \cdot (N_1 + \Pr[E_0(T_0) = \phi | succ_{P^*}^{T_0}] \cdot E[N_C])$ Which gives:

$$
\begin{aligned}
&E[N_C] \leq 1 + (1-\hat{\epsilon}) \cdot E[N_C] + \hat{\epsilon}(N_1 + (1 - \frac{\alpha}{2}) \cdot E[N_C]) \\
&\leq 1 + \hat{\epsilon} \cdot N_1 + E[N_C(1 - \frac{\hat{\epsilon} \cdot \alpha}{2})) \leq \frac{2}{\alpha \cdot \hat{\epsilon}} \cdot (1 + \hat{\epsilon} \cdot N - 1) \\
&= \frac{2}{\alpha \cdot \hat{\epsilon}} \cdot (1 + \hat{\epsilon} \cdot \frac{4 \cdot (N-t)}{(1-\alpha) \cdot \hat{\epsilon} - \epsilon})
\end{aligned}
$$

To obtain an $\alpha$-free formula, let us take $\alpha$ such that $(1-\alpha) \cdot \hat{\epsilon} = \frac{1}{2}(\hat{\epsilon} + \epsilon)$. We have $\alpha = \frac{1}{2}(1 - \frac{\epsilon}{\hat{\epsilon}})$ and the average number of calls to $P^*$ is upper bounded as $\frac{4}{\hat{\epsilon} - \epsilon} \cdot \left(1 + \hat{\epsilon} \cdot \frac{8 \cdot (N-t)}{\hat{\epsilon} - \epsilon}\right)$.

**Zero-Knowledge** There exists an efficient simulator $\mathcal{S}$ that outputs a transcript that is indistinguishable from a real transcript of our NIZK Protocol.

Firstly, $\mathcal{S}$ executes the preprocessing honestly. The simulator $\mathcal{S}$ randomly selects $\left(\vec{\omega_i}, \{[\vec{f_j}]_d(i), [\mathcal{T}(\vec{f_j})]_d(i)\}\right)$ for each $i \in [N], j = 0, \ldots, k+v$ and calculates $\mathcal{T}com_i = H_{com}\left(\vec{\omega_i}, \{[\vec{f_j}]_d(i), [\mathcal{T}(\vec{f_j})]_d(i)\}\right), i \in [N], j = 0, \ldots, k+v$. Meanwhile, $\mathcal{S}$ computes $k+v$ random elements $H_C(\mathcal{T}com_1, \ldots, \mathcal{T}com_N) = \alpha_1, \ldots, \alpha_{k+v} \in \mathbb{F}^k$ and computes $[\vec{\beta_j}]_d = [\vec{f_0}]_d + \cdots + [\vec{f_{k+v}}]_d \alpha_j^{k+v}$ and $[\vec{\gamma_j}]_d = [\mathcal{T}(\vec{f_0})]_d + \cdots + [\mathcal{T}(\vec{f_{k+v}})]_d \alpha_j^{k+v}$ for $j = 1, \ldots, k$. The simulator $\mathcal{S}$ records all shares of $[\vec{\beta_i}]_d$ and $[\vec{\gamma_i}]_d$ for $i = 1, \ldots, k$ in each $\mathcal{V}_i$.

Then, $\mathcal{S}$ chooses a random index set $\mathcal{I}^*$ which is the opened views' index. Additionally, $\mathcal{S}$ simulates $\{[\mathcal{T}(\vec{x}^* + \vec{r}^*)(i)], [\vec{u}^*]_{2d}(i)\}_{i \notin \mathcal{I}^*}$ by utilizing interpolation polynomials, in order to ensure that each $\{\mathcal{V}_i^*\}_{i \in \mathcal{I}^*}$ outputs 1 for $C(x, \omega)$. This procedure by faking the check of each multiplication gate and faking $v$ linear transformation pairs $(r_i, \mathcal{T}(r_i^*))$ in $i \notin \mathcal{I}^*$. This simulation is simple because $\mathcal{S}$ have already determined the set of indexes $\mathcal{I}^*$ and each view $\{\mathcal{V}_i\}_{i \notin \mathcal{I}^*}$ is never opened. Finally, the simulator $\mathcal{S}$ computes the commitment of the view $com_i = H_{com}(\mathcal{V}_i^*)$ for $i \in [\mathcal{I}^*]$ and randomly chooses $com_j$ for $j \notin [\mathcal{I}^*]$. Then $\mathcal{S}$ program the random oracle $H_C$ such that $I^* = H_C(\{com_i^*\}_{i \in \mathcal{I}^*}, \{com_j\}_{j \notin \mathcal{I}^*})$.

As we can see, $\{\mathcal{V}_i^*\}_{i \in \mathcal{I}^*}$ is the same as the output of a real proof. $\{com_i, \mathcal{T}com_i\}_{i \notin \mathcal{I}^*}$ are chosen randomly from the distribution of $H_{com}$ and $H_{com}$ satisfies the property of hiding. Because the preprocessing phase is performed honestly, $\{[\vec{\beta_j}]_d(i), [\mathcal{T}(\vec{\beta_j})]_d\}_{j \in [k]}^{i \notin \mathcal{I}}$, $\{[\mathcal{T}(\vec{x} + \vec{r})(i)]\}_{i \notin [\mathcal{I}]}$ is indistinguishable from the real proof distribution. $\mathcal{I}^*$ is chosen randomly. Therefore, the above simulation is computationally indistinguishable from the real proof.

□

## Appendix B.
## Meta-Review

The following meta-review was prepared by the program committee for the 2024 IEEE Symposium on Security and Privacy (S&P) as part of the review process as detailed in the call for papers.

### B.1. Summary

This paper introduces a new MPC in the Head (MPCitH) NIZK Argument of Knowledge (NIZKAoK) called Diet. The key contribution of Diet is improvement to the concrete efficiency of MPCitH protocols by observing that one can suitably modify the MPCitH protocol to allow for packed Shamir secret sharing to (1) increase efficiency; and (2) reduce the soundness error to negligible without multiple repetitions (as in the case of other MPCitH protocols). The paper demonstrates that these theoretical results lead to practical improvements with applications to proving NP statements about lattice-based problems, obtaining state-of-the-art proof-of-secret-key-knowledge for popular lattice-based KEMs.

### B.2. Scientific Contributions

- The paper provides a valuable step forward in an established field. Namely, it constructs more efficient MPC-in-the-head-based zero knowledge protocols, and shows how to use these for highly-efficient proofs of lattice-theoretic computations.

### B.3. Reasons for Acceptance

1) The paper proposes interesting new techniques to optimize MPC-in-the-head protocols that can serve as the foundation for new directions in such protocols. For example, MPCith protocols underlie many proposed post-quantum signature schemes, and it is plausible that the techniques from this paper can benefit those schemes.
2) The paper is reasonably well-written, and explains its contributions well.