Phecda: Post-Quantum Transparent zkSNARKs from Improved Polynomial Commitment and VOLE-in-the-Head with Application in Publicly Verifiable AES

Changchang Ding and Yan Huang

Indiana University, Bloomington

***Abstract*—We propose Phecda, a new framework to produce quantum-resistant transparent zkSNARKs in the Random Oracle Model. Phecda features a novel multi-linear polynomial commitment scheme and a novel VOLE-in-the-Head zero-knowledge argument, offering a versatile solution for verifying many real-world computations. In particular, we invent a novel AES verification circuit, which, combined with Phecda, allows to verify 1024 blocks of AES in the counter-mode in 10ms using a single-thread program running on a Linux PC.**

1. Introduction

Zero-Knowledge Proof (ZKP) is an important enabling primitive underpinning many critical infrastructures such as remote authentication, electronic voting, cryptocurrency and decentralized finance. ZKPs are often made *non-interactive* and *transparent* (i.e., requiring no trusted setup) to meet even more real-world needs. As a potential game-changer for ensuring security and privacy in the cyberspace, significant research effort and industrial investment have been dedicated to developing concretely faster and smaller NIZK proofs.

Many state-of-the-art zero-knowledge proof systems [1], [2], [3], [4], [5] (Table 1) were built on top of GKR [6], a seminal randomized protocol that translates the validity of a layered circuit computation into exponentially fewer (and mostly linear) polynomial constraints. The crucial differences of those protocols mainly stem from their methods to prove the resulting polynomial constraints. Hyrax [2] used Elliptic Curve Cryptography (ECC) based homomorphic commitments to commit the witnesses and prove the polynomial constraints over them; Libra [1] used random masking polynomials to commit the witnesses and prove the constraints using an ECC-based zero-knowledge verifiable polynomial delegation. Due to the inherent limitation of ECC, these protocols are computationally expensive and cumbersome to support many application circuits over fields where the discrete-log assumption does not hold. Later, Virgo [3] proposed to use Fast Reed-Solomon Interactive (FRI) proof of proximity [7] based verifiable polynomial delegation protocol, which offers faster prover and makes it transparent. Recently, Dubhe [5] combined Fully Linear PCP (FLPCP) [8] with a simplified MPC-in-the-Head (MPCitH) protocol [9] to prove all remaining polynomial constraints. Dubhe offers faster prover and high flexibility in supporting different fields required by target applications. Its verification time and proof size, however, grow linearly with the length of witnesses.

We extend this line of research by investigating new methods to more efficiently handle the polynomial constraints resulted from the GKR transform. As the GKR transform has been steadily improved [1], [4], [10], [11], the cost of finishing the rest of the proof has become increasingly dominant in these protocols. Take AES computation as an example, MPCitH's verifier time and proof size consist of 95% and 62% of the overall costs of Dubhe, which combined GKR with MPCitH. It is therefore crucial to address these new performance bottlenecks. To this end, we develop (1) a transparent Polynomial Commitment (PC) protocol specialized to succinctly prove the *multi-linear* constraint produced by GKR for the input layer, and (2) an efficient VOLE-in-the-Head (VOLEitH) based protocol to prove all linear relations resulted from GKR and FLPCP transformations. Through experiments, we demonstrate our approach produces transparent zkSNARKs that outperform their best existing counterparts. We further use our zkSNARK to realize an efficient publicly verifiable AES protocol.

1.1. Contribution

We propose Phecda, an efficient zero-knowledge proof system based on GKR, FLPCP, PC and VOLEitH-based ZKP. ZK proofs produced by Phecda are succinct, transparent (hence publicly verifiable), and plausibly quantum-resistant. Phecda matches the asymptotic efficiency of Virgo but offers more competitive concrete efficiency than state-of-the-art post-quantum proof systems. Source code of Phecda can be found at github.com/zkPrfs/phecda.

We propose a new transparent multi-linear polynomial commitment scheme that fits especially well with the need of GKR in succinctly verifying polynomial computation over potentially lengthy witnesses. It features a novel integration of ideas from FRI and Gemini [12], which allows to directly verify an n-variate multi-linear polynomial. In contrast to Virgo's polynomial commitment (called zkVPD in Virgo's terminology), ours eliminates the need of the GKR on public inputs, and is concretely more efficient. Moreover, our implementation directly supports binary extension fields.

TABLE 1: Representative GKR-based Zero-Knowledge SNARK Systems

	Libra [1]	**Hyrax** [2]	**Virgo** [3]	**Virgo++** [4]	**Dubhe** [5]	**Phecda**
$\mathcal{P}$	$O(C+w)$	$O(C\log C+w)$	$O(C+w\log w)$	$O(C+w\log w)$	$O(C+w)$	$O(C+w\log w)$
$\mathcal{V}$	$O(d\log C+\log w)$	$O(d\log C+\sqrt{w})$	$O(d\log C+\log^2 w)$	$O(d\log C+d^2+\log^2 w)$	$O(d\log C+w)$	$O(d\log C+\log^2 w)$
$\lvert\pi\rvert$	$O(d\log C+\log w)$	$O(d\log C+\sqrt{w})$	$O(d\log C+\log^2 w)$	$O(d\log C+d^2+\log^2 w)$	$O(d\log C+w)$	$O(d\log C+\log^2 w)$
	Not Transparent	Transparent				
	Not Quantum-Resistant	Plausible Quantum-Resistant				

C: circuit size by gates; d: circuit depth; w: witness length. $C \geq w$. GKR allows to reduce d to a small constant using extra witnesses.

Our new VOLE-in-the-Head-based ZKP is specialized for committing and efficiently proving linear constraints produced by GKR and FLPCP, though its generic version can directly handle any polynomial constraints, thus may be of independent interest. It combines the transparent, GGM-tree-based, VOLE generation from SoftSpokenOT [13]. As shown in Table 2 and Table 8, our VOLEitH ZKP asymptotically and concretely outperform the MPCitH-based ZKP which we aim to replace.

We propose a novel AES verification circuit that works better with GKR and PC based succinct proof systems. The new circuit allows to emulate the mix of byte field and byte ring operations in AES using a larger binary extension field. Comparing to previous AES verification circuits, the number of gates and the number of witnesses in our circuit are significantly smaller. Combined with Phecda, it exhibits unprecedented performance and scalability in proving parallel computation of many AES blocks. E.g., for 128-bit security, our zkSNARK of 1024 AES blocks has 576KB proof size and only takes 10ms verification time (Figure 7).

1.2. Related Work

Phecda shares some high-level design with Dubhe [5], but improved it in several important aspects. First, Phecda uses PC to replace MitH in verifying the evaluation of the secret multi-linear polynomial at a public point. This makes Phecda proof succinct in the witnesses while Dubhe was not. However, this change also calls for fundamentally new circuit design for some computations such as AES and SHA3 in order for GKR to efficiently handle (almost) all gates. Second, Phecda replaces Dubhe's MPCitH with VOLEitH for proving all the linear constraints. This change lends Phecda extra asymptotic and concrete performance advantages (see Section 5.2 and Section 7.3).

Like Gemini [12] and HyperPlonk [14], we commit and prove multi-linear polynomials through their corresponding univariate counterparts. However, their focus was scenarios with trusted setup, hence used bilinear maps over specific elliptic curve fields, whereas ours are based on novel batched Low Degree Test (LDT). As a result, our protocol is transparent, quantum-resistant, and supports a wider range of finite fields including binary extension fields and many more prime fields.

FAEST [15] is a recent post-quantum signature scheme that combines SoftSpokenOT's VOLE generation [13] and Quicksilver [16]'s polynomial verification. Comparing to FAEST, which emphasizes on proving small quadratic circuits, such as AES, over small fields using replication code, our construction focuses on efficiently proving linear relations over large fields using linear Maximum Distance Separable (MDS) codes.

2. Preliminaries

Notation. Given a non-negative integer n, $[n]$ denotes the set $\{0, 1, \ldots, n-1\}$. We use "$a = b$" to denote equality and use ":=" to denote deterministic assignment. We use boldface uppercase letters, such as $\mathbf{A}, \mathbf{U}$, to denote matrices, and boldface lowercase letters, such as $\mathbf{d}$, to denote vectors. We denote the degree of a polynomial by d (in slanted font), while using d (in regular math font) for other context-dependent things, e.g. depth of circuit or VOLE receiver's choice element.

2.1. Interactive Proofs and Arguments

Let $\mathcal{P}$, $\mathcal{V}$ be two interactive Turing machines, and $\langle \mathcal{P}, \mathcal{V} \rangle(\lambda, x)$ be the *transcript*, i.e., concatenation of all communications, of running $\mathcal{P}$, $\mathcal{V}$ over public input x using public security parameter λ. We use $(\mathcal{P}, \mathcal{V})(\lambda, x)$ to denote $\mathcal{V}$'s *acceptance bit* after interacting with $\mathcal{P}$ on input x and λ. We say an NP-relation R defines an NP-language L if $x \in L \Leftrightarrow \exists w, R(w, x) = 1$.

A protocol $(\mathcal{P}, \mathcal{V})$ for an NP-relation R is an ***interactive proof*** if it is both *complete* (i.e., $\forall w, x, R(w, x) = 1 \Rightarrow (\mathcal{P}(w), \mathcal{V})(\lambda, x) = 1$) and *sound* (i.e., $\forall x,\ R(w, x) \neq 1 \Rightarrow \Pr[(\mathcal{P}'(w), \mathcal{V})(\lambda, x) = 1]$ is negligible in λ for any $\mathcal{P}'$). An interactive proof is a *proof of knowledge* if for every efficient $\mathcal{P}'$ there exists an efficient extractor $\varepsilon^{\mathcal{P}'}$ such that, $\Pr[(\mathcal{P}', \mathcal{V})(\lambda, x) = 1 \wedge R(w, x) \neq 1 \mid w \leftarrow \varepsilon^{\mathcal{P}'}(\lambda, x)]$ is negligible in λ. An interactive proof is *Honest-Verifier Zero-Knowledge* (HVZK) if for any prover $\mathcal{P}'$ and the honest verifier $\mathcal{V}$, there exists an efficient simulator $\mathcal{S}$ such that $\langle \mathcal{P}', \mathcal{V} \rangle(\lambda, x)$ is indistinguishable from $\mathcal{S}(x)$, i.e., the output of running $\mathcal{S}$ over x.

A protocol $(\mathcal{P}, \mathcal{V})$ is considered n-*round* if it starts with an initial message a_0 from the $\mathcal{P}$, and continues with n challenge-response message pairs $(e_1, a_1), \ldots, (e_n, a_n)$

where e_i's are uniform challenges from $\mathcal{V}$ and a_i's are the $\mathcal{P}$'s responses, and finalizes with $\mathcal{V}$ outputs 1-bit indicating whether it accepts. The protocol is said to be *public-coin* if all e_i's are public coin tosses independent of previous responses.

An n-round HVZK proof can be compiled to an efficient NIZK in the random oracle model using the Fiat-Shamir transform [17]. In this setting, the soundness is defined only with respect to bounded probabilistic polynomial time $\mathcal{P}'$. Computationally-secure proofs are sometimes more specifically called *argument*s instead of proofs.

2.2. GKR

Multilinear Extension. The multilinear extension of any function $V : \{0,1\}^\ell \mapsto \mathbb{F}$ is denoted by $\widetilde{V} : \mathbb{F}^\ell \mapsto \mathbb{F}$, a polynomial defined as

$$\widetilde{V}(x_0, \ldots, x_{\ell-1}) \stackrel{\text{def}}{=} \sum_{b \in \{0,1\}^\ell} \left(V(b) \prod_{i \in \ell} \left((1-x_i)(1-b_i) + x_i b_i \right) \right).$$

Obviously, $\forall x \in \{0,1\}^\ell, \widetilde{V}(x) = V(x)$. Using multilinear extension, the validity of *all* evaluations of V can be reduced to checking a single uniform random point of $\widetilde{V}$, except for a soundness error of $1/|\mathbb{F}|$.

SumCheck. The goal of the sumcheck protocol [18] is to verify the summation of a polynomial $f : \{0,1\}^\ell \mapsto \mathbb{F}$ on a binary hypercube, i.e., $\sum_{b_i \in \{0,1\}} f(b_1, \ldots, b_\ell)$. It is achieved by reducing a correct summation into evaluating $\widetilde{f}$ on a random point, i.e., $\widetilde{f}(r_1, \ldots, r_\ell)$ with $r_i \in \mathbb{F}$ uniformly picked by $\mathcal{V}$ in each of the ℓ rounds. In the i^{th} round, $\mathcal{P}$ sends polynomial

$$\widetilde{f}_i(x_i) \stackrel{\text{def}}{=} \sum_{b_{i+1}, \ldots, b_\ell \in \{0,1\}} \widetilde{f}(r_1, \ldots, r_{i-1}, x_i, b_{i+1}, \ldots, b_\ell)$$

where $r_1, \ldots, r_{i-1}$ are random values picked by $\mathcal{V}$ in previous rounds. Then $\mathcal{V}$ checks

$$\widetilde{f}_{i-1}(r_{i-1}) = \widetilde{f}_i(0) + \widetilde{f}_i(1) \tag{1}$$

and sends random $r_i \in \mathbb{F}$. Overall, $\mathcal{V}$ checks ℓ linear equalities and the validity of $\widetilde{f}(\boldsymbol{r})$. The i^{th} SumCheck round introduces soundness error $\mathsf{deg}(f_i)/|\mathbb{F}|$.

GKR. Given a d-layer circuit, the GKR protocol [6] captures the correctness of the i^{th} layer computation with a multivariate polynomial $V_i : \{0,1\}^{s_i} \mapsto \mathbb{F}$ defined by

$$V_i(z) = \sum_{x,y} \left(\mathsf{add}_i(x,y,z)\big(V_{i+1}(x) + V_{i+1}(y)\big) + \mathsf{mul}_i(x,y,z) V_{i+1}(x) V_{i+1}(y) \right)$$

where $\mathsf{add}_i, \mathsf{mul}_i$ are the predicate functions for addition and multiplication gates on the i^{th} layer.

The GKR protocol starts from using multilinear extension to reduce the validity of all outputs of the circuit to $\widetilde{V}_0(\boldsymbol{t}_0)$ where $\boldsymbol{t}_0$ is a random point picked by $\mathcal{V}$. Then, for i from 1 to $d-1$, GKR uses SumCheck to verify $V_{i-1}(\boldsymbol{t}_{i-1})$, reducing it to checking the values of $\widetilde{V}_i(\boldsymbol{r}_i), \widetilde{V}_i(\boldsymbol{s}_i)$ and a equality:

$$V_{i-1}(\boldsymbol{t}_{i-1}) = \widetilde{\mathsf{add}}_i(\boldsymbol{r}_i, \boldsymbol{s}_i, \boldsymbol{t}_{i-1})\big(\widetilde{V}_i(\boldsymbol{r}_i) + \widetilde{V}_i(\boldsymbol{s}_i)\big) + \widetilde{\mathsf{mul}}_i(\boldsymbol{r}_i, \boldsymbol{s}_i, \boldsymbol{t}_{i-1}) \widetilde{V}_i(\boldsymbol{r}_i) \widetilde{V}_i(\boldsymbol{s}_i) \tag{2}$$

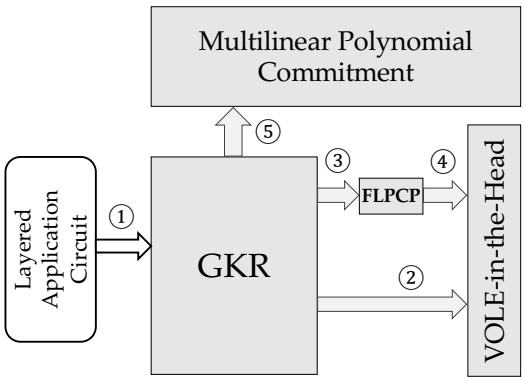

Figure 1: The Phecda System Workflow

where $\boldsymbol{r}_i, \boldsymbol{s}_i, \boldsymbol{t}_{i-1}$ are random challenges picked by $\mathcal{V}$ in previous SumCheck rounds. Finally, at the initial input layer, the validity of any evaluation of $\widetilde{V}_d$ can be established by interpolating the multilinear polynomial defined by the circuit's initial inputs. One concern of this scheme is that the number of points to check on $\widetilde{V}_i$ doubles as i increments. Chiesa-Forbes-Spooner [19] proposed to fix this issue by combining $\widetilde{V}_i(\boldsymbol{r}_i)$ and $\widetilde{V}_i(\boldsymbol{s}_i)$ at each layer using public-coin random coefficients.

Directly interpolating the multilinear polynomial at the initial input layer can take $\mathcal{V}$ linear time and bandwidth (in the input length). To make the proof succinct, Hyrax [2] used homomorphic commitments while Virgo/Virgo+ [3], [4] combined Merkle-based commitment and low-degree tests to accomplish the final check on the initial input layer.

2.3. FLPCP

Fully Linear PCP [8] verifies the correctness of a polynomial evaluation in a 3-step randomized process involving only linear relations. To verify $G(w_0, \ldots, w_{\ell-1}) = 0$ where $G : \mathbb{F}^\ell \to \mathbb{F}$ is an ℓ-variate degree-$\boldsymbol{d}$ polynomial and w_i's are the inputs, the basic idea is

(1) $\mathcal{P}$ encodes each w_i ($i \in [\ell]$) as a random degree-1 polynomial p_i, which are sent to $\mathcal{V}$ along with the polynomial $p_\ell = G(p_1, \ldots, p_\ell)$ in value representation;
(2) $\mathcal{V}$ sends a uniform random element $r \in \mathbb{F}$;
(3) $\mathcal{P}$ sends $p_i(r)$ for all $i \in [\ell+1]$, and $\mathcal{V}$ verifies $p_i(r)$'s are consistent with what can be interpolated using r from the point values received in Step (1).

The idea can be augmented to prove a batch of degree-$\boldsymbol{d}$ ℓ-variate polynomial equations more efficient than proving them individually, but we won't need this augmented variant in most applications since the circuits are generally not deep.

3. The Main Protocol

Figure 1 depicts Phecda's main components and the workflow between them. Here the GKR and FLPCP protocols are used to compile complex input constraints into

simpler but statistically-equivalent output constraints, making them better-suited for subsequent processing.

GKR takes a layered circuit (①) that verifies the target computation, and transforms the validity of the circuit computation into a set of linear equations (②) due to checking Equation (1), d multiplicative equations (③) due to checking Equation (2), and evaluating one secret multilinear polynomial over a public point (⑤) for the input layer due to Equation (3). GKR allows to establish the correctness of a computation between a $O(C+w)$-time prover and a $O(d \log C+w)$-time verifier. GKR alone does not offer zero-knowledge.

FLPCP transforms each multiplicative equation (③) $f = 0$ into $O(\mathsf{deg}(f))$ ($\mathsf{deg}(f) \leq 2$ for all circuits considered in this paper) secret inputs and $O(1)$ linear equations (④), which are fed into the VOLE-in-the-Head module. FLPCP allows linear prover and verifier but does not offer zero-knowledge by itself.

PC translates the validity of a multi-linear polynomial evaluation into $O(\kappa \log n)$ plaintext equality checks (where κ is a security parameter and n is the length of the polynomial) and a FRI-based low degree test. The PC prover runs in $O(n \log n)$ time while the verifier runs in $O(\log^2 n)$ time.

VOLE-in-the-Head receives a number of linear equations involving secrets (②, ④) from the GKR and FLPCP modules, and efficiently proves their validity using our VOLEitH-based ZKP protocol.

The PC and VOLE-in-the-Head modules combine to guarantee Phecda is zero-knowledge. Hence, the four components complement each other in terms of security and performance, allowing Phecda to make the most out of all four components: obtaining zero-knowledge from VOLEitH, FLPCP and PC, while achieving superior asymptotic and concrete efficiency through GKR, PC, and FLPCP.

Protocol Specification and Security. The Phecda protocol is given in Figure 2. Step (1) augments the original circuit C with an extra layer of identity gates so that it suffices to prove one (instead of two) evaluation of the polynomial determined by the witness. Step (2) extends the witness with sufficient dummy values to accommodate the leakage due to the PC: $2\kappa(\log n + 1)$ dummies for checks inside PC and one extra for revealing $\widetilde{V}_d(\boldsymbol{r}_d)$. Then $\mathcal{P}$ commits its witness in Step (3). Finally, Step (4) follows the GKR paradigm to compile the circuit down to probabilistically equivalent constraints amenable to be handled by VOLEitH (Step (4)a), FLPCP (Step (4)b), and PC (Step (4)c) schemes. Like GKR, we use FLPCP as a compiler to turn multiplicative constraints to probabilistically equivalent linear constraints eventually handled by VOLEitH.

We state the security of Phecda in Theorem 1, which can be easily proved based on the security of its component protocols: GKR, FLPCP, PC, and VOLEitH.

Theorem 1. The Phecda protocol given in Figure 2 is a zero-knowledge interactive proof of knowledge with soundness $(1-\mathcal{E}_{\mathrm{GKR}}) \cdot (1-\mathcal{E}_{\mathrm{FLPCP}}) \cdot (1-\mathcal{E}_{\mathrm{VOLEitH}}) \cdot (1-\mathcal{E}_{\mathrm{PC}})$.

Public Inputs: x and the circuit C verifying $R(x, \boldsymbol{w}) = 1$.
Input to $\mathcal{P}$: secret witness $\boldsymbol{w}$.
Protocol:

(1) Augment the circuit C to C' with a layer of identity gates for every input wire of C.
(2) Augment the witness $\boldsymbol{w}$ at the augmented input layer into $\boldsymbol{w}'$ with enough dummy inputs (which are not involved in the circuit evaluation) so that the length of $\boldsymbol{w}'$ is the smallest perfect power of 2 and $|\boldsymbol{w}'| \geq |\boldsymbol{w}| + 2\kappa(\log n + 1) + 1$.
(3) $\mathcal{P}$ computes and sends $\mathsf{com}_{\boldsymbol{w}'} = \mathsf{PC.Com}(\boldsymbol{w}')$ to $\mathcal{V}$.
(4) Apply the GKR transform to circuit C',
  a) In every SumCheck round, a linear equality Equation (1) (where r_{i-1} is a public-coin) is verified using VOLEitH-based ZKP, so that no f_i is leaked for any i.
  b) At the end of each layer, the Equation (2), in which $\widetilde{V}_{i-1}(\boldsymbol{t}_{i-1}), \widetilde{V}_i(\boldsymbol{r}_i), \widetilde{V}_i(\boldsymbol{s}_i)$ are the only secrets, is compiled by FLPCP into several linear constraints, which are then committed and verified using VOLEitH-based ZKP.
  c) At the initial input layer of C' which consists only of unary identity gates, $\mathcal{P}$ and $\mathcal{V}$ call PC to prove
$$\widetilde{V}_d(\boldsymbol{r}_d) = \sum_{i \in [n]} w'_i \cdot \Pi_{j \in [\log n]} \chi(i_j, r_j) \tag{3}$$
  where $\widetilde{V}_d(\boldsymbol{r}_d)$ was sent in plaintext in the last SumCheck round (assuming $\boldsymbol{r}_d$ is not a point on the binary hypercube), w'_i is the i^{th} part of the witness $\boldsymbol{w}'$, i_j is the j^{th} bit of i, r_j is the j^{th} part of the public-coin $\boldsymbol{r}_d$, and $\chi(0, x) = 1 - x$, $\chi(1, x) = x$.
(5) $\mathcal{V}$ accepts if and only if all the checks above have passed.

Figure 2: The Phecda Protocol

Proof. Completeness. Because of the perfect completeness of GKR, FLPCP, VOLEitH, and PC, it is easy to verify that for any x and a $\mathcal{P}$ knowing the witness w such that $R(x, w) = 1$, $\mathcal{V}$ will always accept.

Soundness. Phecda's four components, GKR, FLPCP, PC, and VOLEitH, are put together in a blackbox way, i.e., a component interacts with other components only through explicitly defined input and output. Each of the four components are proven sound with respective soundness error $\mathcal{E}_{\mathrm{GKR}}$, $\mathcal{E}_{\mathrm{FLPCP}}$, $\mathcal{E}_{\mathrm{VOLEitH}}$, and $\mathcal{E}_{\mathrm{PC}}$. Because Phecda is sound whenever all of the four components are sound and each public-coin component proof fails independent of the others, it is easy to infer Phecda's soundness $(1 - \mathcal{E}_{\mathrm{GKR}})(1 - \mathcal{E}_{\mathrm{FLPCP}})(1 - \mathcal{E}_{\mathrm{VOLEitH}})(1 - \mathcal{E}_{\mathrm{PC}})$ by the inclusion-exclusion principle.

Knowledge Soundness. In Phecda, GKR along with FLPCP probabilistically compiles the original NP relation R into a multilinear polynomial constraint R_1 (to be proved by PC) and a set of linear constraints R_2 (to be proved by VOLEitH). This compiler's soundness error is $\mathcal{E}_{\mathrm{Compiler}} \leq \mathcal{E}_{\mathrm{GKR}} + \mathcal{E}_{\mathrm{FLPCP}}$. The original witness is committed by PC, whose knowledge soundness error is at most ϵ_{PC}. Thus, the knowledge extractor for Phecda can simply call PC's knowledge extractor to obtain a w. Because Phecda and its sub-protocols are sound, for any efficient $\mathcal{P}'$ and a w extracted via PC, $\Pr[(\mathcal{P}', \mathcal{V})(x) = 1 \wedge R(w, x) = 0]$ must be negligible. More formally, let $(\mathcal{P}'_1, \mathcal{V}_1, x_1)$ (resp.

$(\mathcal{P}'_2, \mathcal{V}_2, x_2))$ be the prover, verifier and public input associated with the PC (resp. the VOLEitH) sub-protocol. So $\Pr[R_1(w, x_1) = 1 \wedge R_2(f(w), x_2) = 1 | R(w, x) = 0] \leq \mathcal{E}_{\text{Compiler}}$ where f is a public function fixed by GKR and FLPCP. With a w extracted from PC, Phecda's knowledge soundness error $\epsilon \leq \epsilon_{\text{PC}} + \mathcal{E}_{\text{Compiler}} + \mathcal{E}_{\text{VOLEitH}}$ because

$$
\begin{aligned}
&\epsilon = \Pr[(\mathcal{P}', \mathcal{V})(x) = 1 \wedge R(w, x) = 0] \\
&= \Pr\begin{bmatrix} (\mathcal{P}', \mathcal{V})(x) = 1 \\ R_1(w, x_1) = 0 \\ R(w, x) = 0 \end{bmatrix} + \Pr\begin{bmatrix} (\mathcal{P}', \mathcal{V})(x) = 1 \\ R_1(w, x_1) = 1 \\ R(w, x) = 0 \end{bmatrix}, \text{ while} \\
&\Pr\begin{bmatrix} (\mathcal{P}', \mathcal{V})(x) = 1 \\ R_1(w, x_1) = 0 \\ R(w, x) = 0 \end{bmatrix} \leq \Pr\begin{bmatrix} (\mathcal{P}'_1, \mathcal{V}_1)(x_1) = 1 \\ R_1(w, x_1) = 0 \\ R(w, x) = 0 \end{bmatrix} \\
&\leq \Pr\begin{bmatrix} (\mathcal{P}'_1, \mathcal{V}_1)(x_1) = 1 \\ R_1(w, x_1) = 0 \end{bmatrix} = \epsilon_{\text{PC}}, \quad \text{and} \\
&\Pr\begin{bmatrix} (\mathcal{P}', \mathcal{V})(x) = 1 \\ R_1(w, x_1) = 1 \\ R(w, x) = 0 \end{bmatrix} \leq \Pr\begin{bmatrix} (\mathcal{P}'_2, \mathcal{V}_2)(x_2) = 1 \\ R_1(w, x_1) = 1 \\ R(w, x) = 0 \end{bmatrix} \\
&\leq \Pr\left[\begin{array}{c} (\mathcal{P}'_2, \mathcal{V}_2)(x_2) = 1 \\ R_1(w, x_1) = 1 \end{array} \,\middle|\, R(w, x) = 0\right] \\
&= \Pr\begin{bmatrix} R(w, x) = 0 : \\ (\mathcal{P}'_2, \mathcal{V}_2)(x_2) = 1 \\ R_1(w, x_1) = 1 \\ R_2(f(w), x_2) = 1 \end{bmatrix} + \Pr\begin{bmatrix} R(w, x) = 0 : \\ (\mathcal{P}'_2, \mathcal{V}_2)(x_2) = 1 \\ R_1(w, x_1) = 1 \\ R_2(f(w), x_2) = 0 \end{bmatrix} \\
&\leq \Pr\begin{bmatrix} R(w, x) = 0 : \\ R_1(w, x_1) = 1 \\ R_2(f(w), x_2) = 1 \end{bmatrix} + \Pr\begin{bmatrix} (\mathcal{P}'_2, \mathcal{V}_2)(x_2) = 1 \\ R_2(f(w), x_2) = 0 \end{bmatrix} \\
&\leq \mathcal{E}_{\text{Compiler}} + \Pr[(\mathcal{P}'_2, \mathcal{V}_2)(x_2) = 1 \mid R_2(f(w), x_2) = 0] \\
&= \mathcal{E}_{\text{Compiler}} + \mathcal{E}_{\text{VOLEitH}}.
\end{aligned}
$$

Honest-Verifier Zero-Knowledge. GKR and FLPCP only transform the circuits using public randomness, leaving nothing on Phecda's transcript other than their public-coins. Phecda's transcripts are actually produced by VOLEitH and PC, both being honest-verifier zero-knowledge. Also note that the distribution of VOLEitH's transcript is independent of PC, as they are produced with independent randomness. Therefore, it is easy to prove Phecda HVZK by constructing a simulator for Phecda that calls the simulators from the HVZK proofs of VOLEitH and PC then concatenates the resulting transcripts. The indistinguishability proof also directly follows the HVZK property of VOLEitH and PC. □

Support More Fields. Unlike Virgo/Virgo++, Phecda works with both multiplicative and additive cosets of finite fields. This feature lends Phecda greater flexibility to efficiently prove many computations, including some cryptographic ciphers such as AES and SHA3, as well as emulating arithmetic operations of the infinite ring of integers.

Comparison with Dubhe [5]. Dubhe was built by combining GKR, FLPCP and MPCitH. In comparison, Phecda replaces Dubhe's MPCitH with PC and VOLEitH. As a result, Phecda allows proof verification to be *succinct* in the length of the secret witness thanks to PC, while reducing the concrete costs in proving, verifying, and communication. However, to achieve strong succinctness, we need fresh circuit designs to handle computation over the potentially lengthy witness using GKR. In contrast, Dubhe allows computation over the witness also be processed by MPCitH. As an example, Section 6 describes a redesigned AES circuit that is fully compatible with GKR.

Comparison with Virgo/Virgo++. Our approach differs from Virgo/Virgo++ in three aspects: (1) We use a new PC protocol to directly handle the check at the input layer of GKR; (2) We use VOLEitH to provide zero-knowledge for GKR, whereas they relied on random polynomials masks; (3) Our implementation directly supports operations over $\mathbb{F}_{2^k}$, which fits better to many useful computations including AES and SHA3.

4. Multilinear Polynomial Commitment

Virgo viewed the input layer equality check of Equation (3) as verifying the dot product of a secret coefficient vector and a public vector properly computed from verifier's challenges. However, in order to ensure the public vector is correctly computed from the challenges, their approach pays substantial costs to run an extra GKR. In contrast, we directly commit the multilinear polynomial $\widetilde{V}_d$, and prove/verify Equation (3) on the public point $\boldsymbol{r}_d$, hence eliminating the extra GKR.

Definition 1. Let $\boldsymbol{w} = (w_0, \ldots, w_{2^n-1})$ and $f : \mathbb{F}^n \mapsto \mathbb{F}$ be a multi-linear polynomial,

$$f(x_0, \ldots, x_n) = \sum_{i \in [2^n]} w_i \prod_{j \in [n]} \chi(i_j, x_j)$$

where $\chi(0, x) = 1 - x$, $\chi(1, x) = x$, and i_j denotes the j^{th} bit of i. A polynomial commitment scheme with security parameter λ for f consists of three algorithms $(\mathsf{Com}, \mathsf{Prove}, \mathsf{Verify})$ such that

$\mathsf{com}_{\boldsymbol{w}} \leftarrow \mathsf{Com}(1^\lambda, \boldsymbol{w})$**:** Commit the length-$2^n$ multi-linear polynomial f whose coefficients are $\boldsymbol{w}$.

$b \leftarrow (\mathsf{Prove}(\boldsymbol{w}), \mathsf{Verify}(\mathsf{com}_{\boldsymbol{w}}))\,(1^\lambda, \boldsymbol{r}, v)$**:** On input $\boldsymbol{r}, v$, $\mathcal{V}$ (with input com_f) interacts with $\mathcal{P}$ (with secret $\boldsymbol{w}$), and outputs $b = 1$ if and only if $f(\boldsymbol{r}) = v$.

for which the following properties hold:

Completeness: For all $\boldsymbol{w}$, $\boldsymbol{r}$, and $\mathsf{com}_{\boldsymbol{w}} \leftarrow \mathsf{Com}(1^\lambda, \boldsymbol{w})$,

$$(\mathsf{Prove}(\boldsymbol{w}), \mathsf{Verify}(\mathsf{com}_{\boldsymbol{w}}))\,(1^\lambda, \boldsymbol{r}, f(\boldsymbol{r})) = 1.$$

where f is the multi-linear polynomial defined by $\boldsymbol{w}$.

Soundness: For all $\boldsymbol{w}, \boldsymbol{x}, y$, and any probabilistic polynomial adversary $\mathcal{A}$, $\mathsf{com} \leftarrow \mathsf{Com}(1^\lambda, \boldsymbol{w})$, if $f(\boldsymbol{x}) \neq y$ where f is the multi-linear polynomial defined by $\boldsymbol{w}$, then

$$\Pr\left[(\mathcal{A}, \mathsf{Verify}(\mathsf{com}))\,(1^\lambda, \boldsymbol{x}, y) = 1\right] = \mathsf{negl}(\lambda)$$

Notation: Given any i-variate ($\forall 0 \leq i \leq n$) multi-linear polynomial f_i, we use $\check{f}_i(x)$ to denote the unique degree-(2^i-1) polynomial satisfying $\check{f}_i(x) = f_i\left(Z_0^{(i)}(x), Z_1^{(i)}(x), \ldots, Z_{i-1}^{(i)}(x)\right)$ where $\left\{Z_j^{(i)}\right\}_{j\in[i]}$ are defined as follows:

- On large-prime power fields, $Z_j^{(i)}(x) \stackrel{\text{def}}{=} x^{2^j}$.
- On binary extension fields, $Z_j^{(i)}(x) \stackrel{\text{def}}{=} \Pi_{a\in\mathbb{H}_j^{(i)}}(x-a)$ where $\mathbb{H}_j^{(i)}$ is the linear space of 2^i field elements from $\mathbb{F}$, spanned by basis $\left\{\gamma_k^{(i)}\right\}_{k\in[j]}$ where $\mathit{lift}^{(i)}\left(\gamma_{k+1}^{(i)}\right) = \gamma_k^{(i-1)}$.

For every $u \in \mathbb{L}^{(i)}$, let $\hat{u} \in \mathbb{L}^{(i)}$ denote the conjugate element of u in $\mathbb{L}^{(i)}$, i.e., $\mathit{lift}^{(i)}(u) = \mathit{lift}^{(i)}(\hat{u})$ where $\mathit{lift}^{(i)}(x) \stackrel{\text{def}}{=} x^2$ for large prime power fields; and for binary extension fields, $\mathit{lift}^{(i)}(x) \stackrel{\text{def}}{=} x \cdot \left(x + \gamma_0^{(i)}\right)$.

Premise: $2\kappa(n+1)$ out of the 2^n coefficients of f are dummy secret values.

- $\mathsf{com}_f \leftarrow \mathsf{PC.Com}(f; \mathbb{L}^{(0)})$

(1) $\mathcal{P}$ outputs $\mathsf{com}_f = \mathsf{MT.Com}(\check{f}; \mathbb{L}^{(0)})$, where $\mathsf{MT.Com}(\check{f}; \mathbb{L}^{(0)})$ denotes the Merkle tree commitment of $\check{f}$ over the domain $\mathbb{L}^{(0)}$.

- $b \leftarrow \mathsf{PC.Prove}(\mathsf{com}_f, f, (z_0, \ldots, z_{n-1}), y; \mathbb{L}^{(0)})$

(1) $\forall i \in \{1, \ldots, n-1\}$, $f_i(x_0, \ldots, x_{i-1}) \stackrel{\text{def}}{=} f(z_0, \ldots, z_{n-i-1}, x_0, \ldots, x_{i-1})$. $\mathcal{P}$ sends $\mathsf{com}_{f_i} = \mathsf{MT.Com}(\check{f}_i; \mathbb{L}^{(i)})$.
(2) $\mathcal{P}$ and $\mathcal{V}$ run Figure 4 protocol $\mathsf{mLDT}((\check{f}_1, \ldots, \check{f}_{n-1}), (\mathsf{com}_{f_1}, \ldots, \mathsf{com}_{f_{n-1}}), (1, \ldots, n-1))$ to ensure $\deg(\check{f}_i) < 2^i$.
(3) Let $\left\{\left(u_j^{(i)}, \check{f}_i\left(u_j^{(i)}\right)\right)\right\}_{i\in[n]/\{0\}, j\in[\kappa]}$ be the set of points on $\check{f}_i$ opened during the mLDT call above. $\mathcal{P}$ opens $\left\{\left(\hat{u}_j^{(i)}, \check{f}_i\left(\hat{u}_j^{(i)}\right)\right)\right\}_{i\in[n]/\{0\}, j\in[\kappa]}$. $\mathcal{V}$ verifies

$$\check{f}_{i-1}\left(u_j^{(i-1)}\right) = \frac{u_j^{(i)}\check{f}_i\left(\hat{u}_j^{(i)}\right) - \hat{u}_j^{(i)}\check{f}_i\left(u_j^{(i)}\right)}{u_j^{(i)} - \hat{u}_j^{(i)}} + z_{n-i}\frac{\check{f}_i\left(u_j^{(i)}\right) - \check{f}_i\left(\hat{u}_j^{(i)}\right)}{u_j^{(i)} - \hat{u}_j^{(i)}}, \qquad \forall i \in \{2, \ldots, n\}, j \in [\kappa] \tag{4}$$

$$y = \frac{u_j^{(1)}\check{f}_1\left(\hat{u}_j^{(1)}\right) - \hat{u}_j^{(1)}\check{f}_1\left(u_j^{(1)}\right)}{u_j^{(1)} - \hat{u}_j^{(1)}} + z_{n-1}\frac{\check{f}_1\left(u_j^{(1)}\right) - \check{f}_1\left(\hat{u}_j^{(1)}\right)}{u_j^{(1)} - \hat{u}_j^{(1)}}, \qquad \forall j \in [\kappa] \tag{5}$$

Figure 3: The Multilinear Polynomial Commitment Protocol

Knowledge Soundness: For all $\boldsymbol{x}, y$, and any probabilistic polynomial adversary $\mathcal{A}$, there exists $\varepsilon^{\mathcal{A}}$ such that for every $\boldsymbol{w} \leftarrow \varepsilon^{\mathcal{A}}(1^\lambda, \boldsymbol{x}, y)$ and $\mathsf{com} \leftarrow \mathsf{Com}(1^\lambda, \boldsymbol{w})$

$$\Pr\left[\begin{array}{c} f(\boldsymbol{x}) \neq y \\ (\mathcal{A}, \mathsf{Verify}(\mathsf{com}))\left(1^\lambda, \boldsymbol{x}, y\right) = 1 \end{array}\right] = \mathsf{negl}(\lambda)$$

where f is the multi-linear polynomial defined by $\boldsymbol{w}$.

Zero-Knowledge: For all $\boldsymbol{w}, \boldsymbol{x}$, $\mathsf{com} \leftarrow \mathsf{Com}(1^\lambda, \boldsymbol{w})$, and for any computationally bounded adversary $\mathcal{A}$, there exists an efficient simulator S such that

$$\langle \mathsf{Prove}(\boldsymbol{w}), \mathcal{A}(\mathsf{com})\rangle\left(1^\lambda, \boldsymbol{x}, f(\boldsymbol{x})\right) \approx S\left(1^\lambda, \boldsymbol{x}, f(\boldsymbol{x})\right)$$

where the left-hand side is the distribution of the protocol transcript and "$\approx$" indicates the distributions on the two sides are indistinguishable.

Basic Ideas. We leverage the one-to-one correspondence between an i-variate multi-linear polynomial f_i and its corresponding degree-2^i univariate polynomial $\check{f}_i$. The correspondence is defined by univariate polynomials $\left\{Z_j^{(i)}(x)\right\}_{j\in[i]}$. To commit f_i, it suffices to commit $\check{f}_i$ using a Merkle tree with leaves containing $\check{f}_i(x)$ for all $x \in \mathbb{L}^{(i)}$ with $\mathbb{L}^{(i)}$ a sufficiently large subset of $\check{f}_i$'s domain. Note that $\{\mathbb{L}^{(i)}\}_{i\in[n]}$ is a hierarchy of Fast Fourier Transform (FFT)-friendly sets, i.e., $|\mathbb{L}^{(i+1)}| = 2|\mathbb{L}^{(i)}|$ and $\forall x \in \mathbb{L}^{(i)}$, there exists $\hat{x} \in \mathbb{L}^{(i)}$ called the *conjugate* of x such that $\hat{x} \neq x$ but $\mathit{lift}^{(i)}(\hat{x}) = \mathit{lift}^{(i)}(x) \in \mathbb{L}^{(i-1)}$ where $\mathit{lift}^{(i)}$ is a function that lifts any element of $\mathbb{L}^{(i)}$ to one in $\mathbb{L}^{(i-1)}$, hence able to support protocols like FFT and FRI.

Given a public point $\boldsymbol{z} = (z_0, \ldots, z_{n-1})$ and y, to prove $f(\boldsymbol{z}) = y$, we use a series of multi-linear polynomials f_i specialized by partially evaluating f on the first $n-i$ parts of $\boldsymbol{z}$. By re-grouping the terms, for all $i \leq n$ the i-variate multi-linear polynomial f_i can be written as

$$f_i(x_0, \ldots, x_{i-1}) = h_i(x_1, \ldots, x_{i-1}) + x_0 h_i'(x_1, \ldots, x_{i-1})$$

with multi-linear polynomials h_i, h_i' uniquely defined by f_i. The prover will commit f_i through its corresponding univariate polynomial $\check{f}_i$. The validity of $f(z) = y$ can thus be verified by checking the validity of the polynomials $\{\check{f}_i\}_{i\in[n]}$. Since $\check{f}_i$'s are committed through their evaluations on the hierarchical domains $\mathbb{L}^{(i)}$, it suffices to verify FFT-like equalities of $\check{f}_i$ (i.e., Equation (4) and Equation (5) in Figure 3) on sufficiently many randomly sampled points.

Premise: $\forall i \in [n]$, f_i is a degree-2^i univariate polynomial that is committed over $\mathbb{L}^{(i)}$. $\{\mathbb{L}^{(i)}\}_{i\in[n]}$ is a hierarchy of FFT-friendly domains.

- $b \leftarrow \mathsf{mLDT}((f_1,\ldots,f_n),(\mathsf{com}_{f_1},\ldots,\mathsf{com}_{f_n}),(1,\ldots,n))$

(1) Let $g_n = f_n$. For $i \in \{n,\ldots,2\}$,

a) Use $g_i(x)$ to define $g_{i,o}(x), g_{i,e}(x)$ such that $g_{i,e}(\mathit{lift}^{(i)}(x)) + x \cdot g_{i,o}(\mathit{lift}^{(i)}(x)) = g_i(x)$.

b) $\mathcal{V}$ sends random $\alpha_i, \beta_i \in \mathbb{F}$ to $\mathcal{P}$. Let $g_{i-1}(x) \stackrel{\text{def}}{=} g_{i,e}(x) + \alpha_i g_{i,o}(x) + \beta_i f_{i-1}(x)$. $\mathcal{P}$ sends $\mathsf{com}_{g_{i-1}} = \mathsf{MT.Com}(g_{i-1}; \mathbb{L}^{(i-1)})$ to $\mathcal{V}$.

(2) $\mathcal{V}$ sends random $\alpha_1 \in \mathbb{F}$ to $\mathcal{P}$. Since $g_0(x) \stackrel{\text{def}}{=} g_{1,e}(x) + \alpha_1 g_{1,o}(x)$ is a constant polynomial, $\mathcal{P}$ opens g_0 directly.

(3) $\mathcal{V}$ sends random $\left\{u_j^{(n)} \in \mathbb{L}^{(n)} \backslash \mathbb{W}^{(n)}\right\}_{j\in[\kappa]}$ where $\mathbb{W}^{(n)} = \left\{u \,\middle|\, \left(Z_0^{(n)}(u),\ldots,Z_{n-1}^{(n)}(u)\right) \in \{0,1\}^n\right\}$. $\mathcal{P}$ opens g_n on points $\left\{u_j^{(n)}\right\}_{j\in[\kappa]}$. For i from n down to 2,

a) $\mathcal{P}$ opens $\left\{g_i\left(\hat{u}_j^{(i)}\right)\right\}_{j\in[\kappa]}$ and opens f_{i-1}, g_{i-1} on points $\left\{u_j^{(i-1)}\right\}_{j\in[\kappa]}$ where $u_j^{(i-1)} = \mathit{lift}^{(i)}\left(u_j^{(i)}\right)$.

b) $\mathcal{V}$ verifies $\forall j \in [\kappa]$, $g_{i-1}\left(u_j^{(i-1)}\right) - \beta_i \cdot f_{i-1}\left(u_j^{(i-1)}\right) = \dfrac{u_j^{(i)} g_i\left(\hat{u}_j^{(i)}\right) - \hat{u}_j^{(i)} g_i\left(u_j^{(i)}\right)}{u_j^{(i)} - \hat{u}_j^{(i)}} + \alpha_i \cdot \dfrac{g_i\left(u_j^{(i)}\right) - g_i\left(\hat{u}_j^{(i)}\right)}{u_j^{(i)} - \hat{u}_j^{(i)}}$.

$\mathcal{V}$ verifies $\forall j \in [\kappa]$, $g_0 = \dfrac{u_j^{(1)} g_1\left(\hat{u}_j^{(1)}\right) - \hat{u}_j^{(1)} g_1\left(u_j^{(1)}\right)}{u_j^{(1)} - \hat{u}_j^{(1)}} + \alpha_1 \cdot \dfrac{g_1\left(u_j^{(1)}\right) - g_1\left(\hat{u}_j^{(1)}\right)}{u_j^{(1)} - \hat{u}_j^{(1)}}$.

Figure 4: The Special Batched LDT Protocol

Conditioned on the fact that the degrees of $\left\{\check{f}_i\right\}_{i\in[n]}$ are all properly bounded (by LDT), checking Equation (4) and Equation (5) on sufficiently many points guarantees that all $\check{f}_i$ are honestly computed except for a negligible probability.

If realized naively, this would require n separate polynomial LDTs. This is especially true in case of binary extension fields where $\mathbb{L}^{(i)}$ does not necessarily contain $\mathbb{L}^{(i-1)}$, so the trick of committing and testing a random linear combination of properly lifted $\check{f}_i$ will not work. To cope with this challenge, we propose a protocol (Figure 4) to batch execute these n special LDTs at a cost comparable to a single one. The idea is to committing $\check{f}_i$ and g_i over $\mathbb{L}^{(i)}$ where g_i is a degree-2^i univariate polynomial constructed from a random combination of f_{i-1} and the "odd" and "even" parts of g_{i+1} (see Step (1)b). In the end, it is trivial to prove g_0 is a constant. Our batched LDT does not invoke FRI but uses similar ideas as FRI: the verifier checks the FFT-like relations among $g_i, f_i, g_{i-1}, f_{i-1}$ on κ random points (see Step (3)b).

Our polynomial commitment protocol, along with the batched LDT subroutine, will reveal $2\kappa(n+1)$ points during their equality checks. Each of these point will reveal a linear equation involving the original witnesses. To make sure it is zero-knowledge, we require that the input polynomial f contains at least $2\kappa(n+1)$ dummy secret coefficients, so the process leaks nothing about the rest $2^n - 2\kappa(n+1)$ witnesses.

The above ideas work for both fields with multiplicative cosets (e.g., large prime modulus fields and their extensions) and binary extension fields which require additive cosets. The differences stem from the definitions of the basis polynomials $Z_j^{(i)}(x)$ and the hierarchical sets $\mathbb{L}^{(i)}$. We give a field-agnostic presentation of our polynomial commitment protocol in Figure 3, where $Z_j^{(i)}(x)$'s are defined for the two different types of fields.

Specification and Security Proofs. The PC protocol is fully specified in Figure 3, with the special batched LDT protocol given in Figure 4. We state and prove the security of our polynomial commitment scheme as Theorem 2 below.

Theorem 2. The protocol given in Figure 3 is an honest-verifier polynomial commitment scheme.

Proof. Completeness. Since f_i is multi-linear, there exist unique multi-linear polynomials h_i, h_i' such that

$$f_i(x_0, x_1 \ldots, x_{i-1}) = h_i(x_1,\ldots,x_{i-1}) + x_0 h_i'(x_1,\ldots,x_{i-1})$$

Hence,

$$\begin{aligned}
&\check{f}_i(u)\\
=&h_i\left(Z_1^{(i)}(u),\ldots,Z_{i-1}^{(i)}(u)\right)\\
&\qquad + Z_0^{(i)}(u)\cdot h_i'\left(Z_1^{(i)}(u),\ldots,Z_{i-1}^{(i)}(u)\right)\\
=&h_i\left(Z_1^{(i)}(u),\ldots,Z_{i-1}^{(i)}(u)\right) + u\cdot h_i'\left(Z_1^{(i)}(u),\ldots,Z_{i-1}^{(i)}(u)\right)\\
=&h_i\left(Z_0^{(i-1)}(v),\ldots,Z_{i-2}^{(i-1)}(v)\right)\\
&\qquad + u\cdot h_i'\left(Z_0^{(i-1)}(v),\ldots,Z_{i-2}^{(i-1)}(v)\right)
\end{aligned}$$

where $v = \mathit{lift}^{(i)}(u)$. The 3^{rd} equality is based on Lemma 1.

Similarly, for $\hat{u}$ such that $\mathit{lift}^{(i)}(\hat{u}) = \mathit{lift}^{(i)}(u) = v$,

$$\begin{aligned}
\check{f}_i(\hat{u}) =&h_i\left(Z_0^{(i-1)}(v),\ldots,Z_{i-2}^{(i-1)}(v)\right)\\
&\qquad + \hat{u}\cdot h_i'\left(Z_0^{(i-1)}(v),\ldots,Z_{i-2}^{(i-1)}(v)\right)
\end{aligned}$$

Therefore, we can solve h and h',

$$h_i\left(Z_0^{(i-1)}(v),\ldots,Z_{i-2}^{(i-1)}(v)\right)=\frac{\hat{u}\check{f}_i(u)-u\check{f}_i(\hat{u})}{\hat{u}-u},$$
$$h'_i\left(Z_0^{(i-1)}(v),\ldots,Z_{i-2}^{(i-1)}(v)\right)=\frac{\check{f}_i(u)-\check{f}_i(\hat{u})}{u-\hat{u}}.$$

Finally, by definition,

$$\begin{aligned}&f_{i-1}(x_0,\ldots,x_{i-2})\\ =&f_i(z_i,x_0\ldots,x_{i-2})\\ =&h_i(x_0,\ldots,x_{i-2})+z_ih'_i(x_0,\ldots,x_{i-2})\end{aligned}$$
$$\begin{aligned}\therefore\ \check{f}_{i-1}(v)=&h_i\left(Z_0^{(i-1)}(v),\ldots,Z_{i-2}^{(i-1)}(v)\right)\\ &+z_ih'_i\left(Z_0^{(i-1)}(v),\ldots,Z_{i-2}^{(i-1)}(v)\right)\end{aligned}$$
$$\therefore\ \check{f}_{i-1}(v)=\frac{\hat{u}\check{f}_i(u)-u\check{f}_i(\hat{u})}{\hat{u}-u}+z_i\frac{\check{f}_i(u)-\check{f}_i(\hat{u})}{u-\hat{u}},$$

hence all the checks should pass.

Soundness. In our analysis below, we ignore collisions on the hashes since they occur with negligible probability. Let $\Delta(\cdot,\cdot)$ be the relative Hamming distance between two vectors, and $\Delta(\cdot,\mathtt{RS}[\mathbb{L},\rho])$ be the minimum distance between a vector and a codeword of the RS code. For every vector $\hat{f}_i$ committed by the prover, let F_i be the set of closest codewords to $\hat{f}_i$ in $\mathtt{RS}[\mathbb{L}_\mathtt{i},\rho]$ by Hamming distance. Let M_i be the bijective mapping from a univariate polynomial $\check{f}_i$ to its corresponding i-variate multi-linear polynomial f_i where the correspondence is as given in Figure 3. Define

$\mathtt{ChkP}_i$: Equation (4) holds on a particular i for all j;
$\mathtt{ChkP}_{n-1}$: Equation (5) holds;
$\mathtt{mLDTP}$: mLDT checks passed;
$\mathtt{LD}(\delta)$: $\max\limits_{i\in[n+1]}\Delta(\hat{f}_i,\mathtt{RS}[\mathbb{L}_\mathtt{i},\rho])=\delta$;
$\mathtt{B}_i$: $\forall h_i\in F_i,h_{i+1}\in F_{i+1}$,
$\quad h_i(x)\neq h_{i+1,e}(x)+z_{n-i-1}h_{i+1,o}(x)$
where $h_{i+1,e}(lift^{(i)}(x))+x\cdot h_{i+1,o}(lift^{(i)}(x))=h_{i+1}(x)$
$\mathtt{B}$: $\forall h_n\in F_n,y\neq(M_n(h_n))(z_0,\ldots,z_{n-1})$
$\mathtt{G}_n$: **true**;
$\mathtt{G}_i$: $\mathtt{G}_{i+1}\wedge\neg\mathtt{B}_i$.

By definition, the soundness error

$$\begin{aligned}\varepsilon=&\Pr\left[\mathtt{mLDTP}\wedge\bigwedge_{i\in[n]}\mathtt{ChkP}_i\;\middle|\;\mathtt{B}\right]\\ \leq&\max\left\{\Pr\left[\mathtt{mLDTP}\wedge\bigwedge_{i\in[n]}\mathtt{ChkP}_i\mid\mathtt{B}\wedge\mathtt{G}_{n-1}\right],\right.\\ &\left.\Pr\left[\mathtt{mLDTP}\wedge\bigwedge_{i\in[n]}\mathtt{ChkP}_i\mid\mathtt{B}\wedge\mathtt{B}_{n-1}\right]\right\}\\ \leq&\max\left\{\Pr\left[\mathtt{mLDTP}\wedge\bigwedge_{i\in[n]}\mathtt{ChkP}_i\mid\mathtt{B}\wedge\mathtt{G}_{n-2}\right],\right.\\ &\Pr\left[\mathtt{mLDTP}\wedge\bigwedge_{i\in[n]}\mathtt{ChkP}_i\mid\mathtt{B}\wedge\mathtt{G}_{n-1}\wedge\mathtt{B}_{n-2}\right],\\ &\left.\Pr\left[\mathtt{mLDTP}\wedge\bigwedge_{i\in[n]}\mathtt{ChkP}_i\mid\mathtt{B}\wedge\mathtt{B}_{n-1}\right]\right\}\\ \leq&\max_{i\in[n]}\left\{\Pr\left[\mathtt{mLDTP}\wedge\bigwedge_{j\in[n]}\mathtt{ChkP}_j\mid\mathtt{B}\wedge\mathtt{G}_0\right],\right.\\ &\left.\Pr\left[\mathtt{mLDTP}\wedge\bigwedge_{j\in[n]}\mathtt{ChkP}_j\mid\mathtt{B}\wedge\mathtt{G}_{i+1}\wedge\mathtt{B}_i\right]\right\}\\ =&\max_{i\in[n]}\left\{\Pr\left[\mathtt{mLDTP}\wedge\bigwedge_{j\in[n]}\mathtt{ChkP}_j\;\middle|\;\mathtt{B}\wedge\mathtt{G}_0\wedge\exists k,|F_k|\geq2\right],\right.\\ &\left.\Pr\left[\mathtt{mLDTP}\wedge\bigwedge_{j\in[n]}\mathtt{ChkP}_j\mid\mathtt{B}\wedge\mathtt{G}_{i+1}\wedge\mathtt{B}_i\right]\right\}\\ \leq&\max_{i\in[n]}\left\{\Pr[\mathtt{mLDTP}\wedge\mathtt{ChkP}_i\mid\mathtt{B}\wedge\mathtt{G}_0\wedge|F_i|\geq2],\right.\\ &\left.\Pr[\mathtt{mLDTP}\wedge\mathtt{ChkP}_i\mid\mathtt{B}\wedge\mathtt{G}_{i+1}\wedge\mathtt{B}_i]\right\}\\ \leq&\max_{i\in[n];\delta}\left\{\Pr[\mathtt{mLDTP}\wedge\mathtt{ChkP}_i\mid\mathtt{B}\wedge\mathtt{G}_0\wedge|F_i|\geq2\wedge\mathtt{LD}(\delta)],\right.\\ &\left.\Pr[\mathtt{mLDTP}\wedge\mathtt{ChkP}_i\mid\mathtt{B}\wedge\mathtt{G}_{i+1}\wedge\mathtt{B}_i\wedge\mathtt{LD}(\delta)]\right\}\\ =&\max_{i\in[n];\delta}\left\{\Pr[\mathtt{ChkP}_i\mid\mathtt{mLDTP}\wedge\mathtt{B}\wedge\mathtt{G}_0\wedge|F_i|\geq2\wedge\mathtt{LD}(\delta)]\right.\\ &\quad\cdot\Pr[\mathtt{mLDTP}\mid\mathtt{B}\wedge\mathtt{G}_0\wedge|F_i|\geq2\wedge\mathtt{LD}(\delta)],\\ &\quad\Pr[\mathtt{ChkP}_i\mid\mathtt{mLDTP}\wedge\mathtt{B}\wedge\mathtt{G}_{i+1}\wedge\mathtt{B}_i\wedge\mathtt{LD}(\delta)]\\ &\left.\quad\cdot\Pr[\mathtt{mLDTP}\mid\mathtt{B}\wedge\mathtt{G}_{i+1}\wedge\mathtt{B}_i\wedge\mathtt{LD}(\delta)]\right\}\\ \leq&\max_{i\in[n];\delta}\left\{\left(\frac{2^i(1+2\rho\delta)}{|\mathbb{L}_i|}\right)^\kappa,\left(\frac{2^i(1+3\rho\delta)}{|\mathbb{L}_i|}\right)^\kappa\right\}\cdot\varepsilon_{\mathsf{mLDT}}(\delta)\\ \leq&\max_\delta\left(\frac{1}{\rho}+3\delta\right)^\kappa\cdot\varepsilon_{\mathsf{mLDT}}(\delta)\end{aligned}$$

Let j be the last mLDT round where $\Delta(\hat{f}_j,\mathtt{RS}[\mathbb{L}_\mathtt{j},\rho])=\delta$. To maximize ϵ_{mLDT}, the prover will maintain $\Delta(\hat{f}_i,\mathtt{RS}[\mathbb{L}_\mathtt{i},\rho])=0$, i.e., $\hat{f}_i=\check{f}_i$ for all $i<j$. Therefore, the mLDT protocol from round j to the end is the same as the batched FRI protocol of [20] except that in each round our target polynomial has an additive term $\beta_{i+1}\check{f}_i$. Since the soundness of batched FRI only depends on the target polynomials' distances to the RS code and adding $\beta_{i+1}\check{f}_i$ has no impact on these distances, we have $\epsilon_{\mathsf{mLDT}}(\delta)\leq\epsilon_{\mathtt{FRI}}(\delta)$, which has been proven negligible in [20].

Knowledge Soundness. Thanks to the extractability of Merkle tree, given the root hash used for committing a degree-d polynomial, the knowledge extractor can rewind the prover to learn $O(d)$ distinct paths of a Merkle tree to fully reconstruct all leaves. This further allows to recover with high probability the committed secret polynomial, through decoding the RS code. Because our protocol is sound, the probability that a prover convinces the verifier while evaluating the extracted polynomial at $(z_0,\ldots,z_{n-1})$ does not equal y has to be negligible.

Honest-Verifier Zero-Knowledge. Each opened point on

f_i and g_i can be seen as a linear combination of the original 2^n coefficients of f. There are a total of 2κ points opened for each f_i ($i \in [n+1]$) and g_i $i \in [n]$ (recall that $f \stackrel{\text{def}}{=} f_n$ and $f_0 = y$), while half of the points on g_i and f_i are interdependent with each other, constrained by Equation (4). Therefore, at most $2\kappa + 2\kappa n$ independent linear combinations are actually revealed. Thus, a simulator can be constructed as follows:

(1) Pick uniform g_0, α_1.
(2) Pick linear polynomials $\check{f}_1, g_1$ such that $\check{f}_1(z_{n-1}) = y, g_1(\alpha_1) = g_0$. Pick κ uniform points in $\mathbb{L}_1$ and evaluate $\check{f}_1, g_1$ on these points and also the corresponding conjugate points.
(3) $\forall i \in \{2, \ldots, n\}$, pick uniform $\alpha_i, \beta_i \in \mathbb{F}$. And for each of the previous picked κ points $u_j^{(i-1)}$, randomly choose $u_j^{(i)}$ such that $\mathit{lift}^{(i)}(u_j^{(i)}) = u_j^{(i-1)}$.
For each distinct $u_j^{(i-1)}$, pick uniform value for $\check{f}_i(u_j^{(i)})$, and $g_i(u_j^{(i)})$.
If for some j, k, $u_j^{(i-1)} = u_k^{(i-1)}$ and $u_j^{(i)} = u_k^{(i)}$, pick the same value for both $\check{f}_i(u_j^{(i)})$ and $\check{f}_i(u_k^{(i)})$, and same value for both $g_i(u_j^{(i)})$ and $g_i(u_k^{(i)})$.
If for some j, k, $u_j^{(i-1)} = u_k^{(i-1)}$ and $u_j^{(i)} \neq u_k^{(i)}$, thus $u_k^{(i)} = \hat{u}_j^{(i)}$, pick a value for $\check{f}_i(u_j^{(i)})$ and set value of $\check{f}_i(u_k^{(i)})$ using Equation (4),
(4) For each of the κ points above, compute its conjugate point. Determine the values of $\check{f}_i$ at these conjugate points using Equation (4) and Equation (5); determine the values of g_i at these conjugate points using the two equations in Step (3)b.
(5) Output the $2\kappa(n+1)$ random points, $\{\alpha_i, \beta_i\}_{i\in[n]}$, and the Merkle tree opening messages as the transcripts.

It is easy to verify that the output of the simulator and the legitimate transcripts are identically distributed. □

Lemma 1. $Z_{j+1}^{(i+1)}(x) = Z_j^{(i)}\left(\mathit{lift}^{(i+1)}(x))\right)$

Proof. We prove this equality for both large prime power fields and binary extension fields.

Large prime power fields. In this case,

$$Z_{j+1}^{(i+1)}(x) = x^{2^{j+1}} = \left(x^2\right)^{2^j} = Z_j^{(i)}\left(x^2\right) = Z_j^{(i)}\left(\mathit{lift}^{(i+1)}(x)\right)$$

Binary extension fields. Recall that

$$Z_j^{(i)} \stackrel{\text{def}}{=} \prod_{a\in\mathbb{H}_j^{(i)}} (x-a),$$

where

$$\begin{aligned}\mathbb{H}_j^{(i)} &\stackrel{\text{def}}{=} \left\{\sum_{k=0}^{j-1} b_k\gamma_k^{(i)} \middle| (b_0,\ldots,b_{j-1}) \in \{0,1\}^j\right\} \\ &= \left\{\sum_{k=0}^{j-2} b_k\gamma_k^{(i)}\right\} \cup \left\{\gamma_{j-1}^{(i)} + \sum_{k=0}^{j-2} b_k\gamma_k^{(i)}\right\}\end{aligned}$$

with linear-independent $\left\{\gamma_k^{(i)} \in \mathbb{F}\right\}_{k\in[j]}$ over binary coefficients.

Therefore, $Z_j^{(i)}(x) = Z_{j-1}^{(i)}(x) Z_{j-1}^{(i)}\left(x + \gamma_{j-1}^{(i)}\right)$.

$$\begin{aligned}
&\mathbb{H}_{i-1}^{(i-1)} = \left\{\mathit{lift}^{(i)}(x) \mid x \in \mathbb{H}_i^{(i)}\right\} \\
=&\left\{\left(\sum_{k=0}^{i-1} b_k\gamma_k^{(i)}\right)\left(\sum_{k=0}^{i-1} b_k\gamma_k^{(i)} + \gamma_0^{(i)}\right) \middle| \boldsymbol{b} \in \{0,1\}^i\right\} \\
=&\left\{\left(\sum_{k=0}^{i-1} b_k\gamma_k^{(i)}\right)^2 + \left(\sum_{k=0}^{i-1} b_k\gamma_k^{(i)}\right)\gamma_0^{(i)} \middle| \boldsymbol{b} \in \{0,1\}^i\right\} \\
=&\left\{\left(\sum_{k=0}^{i-1} b_k^2\left(\gamma_k^{(i)}\right)^2\right) + \sum_{k=0}^{i-1} b_k\gamma_k^{(i)}\gamma_0^{(i)} \middle| \boldsymbol{b} \in \{0,1\}^i\right\} \\
=&\left\{\sum_{k=0}^{i-1} b_k\left(\gamma_k^{(i)}\right)^2 + \sum_{k=0}^{i-1} b_k\gamma_k^{(i)}\gamma_0^{(i)} \middle| \boldsymbol{b} \in \{0,1\}^i\right\} \\
=&\left\{\sum_{k=0}^{i-1} b_k\gamma_k^{(i)}(\gamma_k^{(i)} + \gamma_0^{(i)}) \middle| (b_0,\ldots,b_{i-1}) \in \{0,1\}^i\right\} \\
=&\left\{\sum_{k=1}^{i-1} b_k\gamma_k^{(i)}(\gamma_k^{(i)} + \gamma_0^{(i)}) \middle| (b_1,\ldots,b_{i-1}) \in \{0,1\}^{i-1}\right\}
\end{aligned}$$

Therefore, by defining $\gamma_k^{(i-1)} \stackrel{\text{def}}{=} \gamma_{k+1}^{(i)}(\gamma_{k+1}^{(i)} + \gamma_0^{(i)})$ for all $k \in [i-1]$, we can ensure $\mathbb{H}_{i-1}^{(i-1)}$ is a linear space spanned by $\{\gamma_k^{(i-1)}\}_{k\in[i-1]}$.

Now we can prove the lemma by induction. When $j = 0$, by definition $Z_1^{(i+1)}(x) = x(x + \gamma_0^{(i+1)}) = \mathit{lift}^{(i+1)}(x)$ and $Z_0^{(i)}(x) = x$, thus $Z_1^{(i+1)}(x) = Z_0^{(i)}\left(\mathit{lift}^{(i+1)}(x)\right)$. Hypothesizing the lemma holds for $j = k$, then for $j = k+1$,

$$\begin{aligned}
&Z_{k+1}^{(i+1)}(x) = Z_k^{(i+1)}(x) Z_k^{(i+1)}\left(x + \gamma_k^{(i+1)}\right) \\
=&Z_{k-1}^{(i)}\left(x(x + \gamma_0^{(i+1)})\right) \\
&\quad \cdot Z_{k-1}^{(i)}\left((x + \gamma_k^{(i+1)})(x + \gamma_k^{(i+1)} + \gamma_0^{(i+1)})\right) \\
=&Z_{k-1}^{(i)}\left(x(x + \gamma_0^{(i+1)})\right) \\
&\quad \cdot Z_{k-1}^{(i)}\left(x(x + \gamma_0^{(i+1)}) + \gamma_k^{(i+1)}(\gamma_k^{(i+1)} + \gamma_0^{(i+1)})\right) \\
=&Z_{k-1}^{(i)}\left(x(x + \gamma_0^{(i+1)})\right) Z_{k-1}^{(i)}\left(x(x + \gamma_0^{(i+1)}) + \gamma_{k-1}^{(i)}\right) \\
=&Z_k^{(i)}\left(x(x + \gamma_0^{(i+1)})\right) = Z_k^{(i)}\left(\mathit{lift}^{(i+1)}(x)\right)
\end{aligned}$$

□

5. Transparent VOLE-in-the-Head ZKP

In this section, we present a VOLE-in-the-Head protocol that allows a prover $\mathcal{P}$ to commit some secret $\mathbf{w} = [w_1, \ldots, w_n] \in \mathbb{F}_p^n$, and later prove in zero-knowledge that public constants $\mathbf{a} \in \mathbb{F}_p^n$ and $b \in \mathbb{F}$, linear constraints like $\mathbf{a} \cdot \mathbf{w} = b$ hold.

Goal: $\mathcal{P}$ holding a secret $\mathbf{X}$ proves in zero-knowledge to $\mathcal{V}$ that $\mathbf{A} \circ \mathbf{X} + a = 0$ for public $\mathbf{A} \in \mathbb{F}_p^{\ell \times m}$, $a \in \mathbb{F}_p$, where $\circ : \mathbb{F}_p^{\ell \times m} \times \mathbb{F}_p^{\ell \times m} \to \mathbb{F}_p$ computes the sum of element-wise products of two matrices.

Offline preparation: (before $\mathbf{X}, \mathbf{A}, a$ are known)

$\mathcal{P}$ commits n GGM trees. Each tree has N leaves. Treat the k^{th} leaf of the j^{th} tree as seed $seed_{j,k}$. $\forall i \in [2\ell], j \in [n], k \in [N]$, $\mathcal{P}$ computes $t_{i,j,k} \coloneqq \mathtt{PRG}(i, seed_{j,k}) \in \mathbb{F}_p$, and

$$u_{i,j} \coloneqq \sum_{k=0}^{N-1} t_{i,j,k},\ v_{i,j} \coloneqq -\sum_{k=0}^{N-1} s_k t_{i,j,k}$$

where $\{s_k \in \mathbb{F}_p\}_{k\in[N]}$ are distinct public values. Let $\mathbf{U} \in \mathbb{F}_p^{2\ell \times n}, \mathbf{V} \in \mathbb{F}_p^{2\ell \times n}$ be matrices whose entries are $u_{i,j}$ and $v_{i,j}$, resp. $\mathcal{P}$ sets $\begin{bmatrix}\mathbf{U}_1 \\ \mathbf{U}_2\end{bmatrix} \mathbf{C} \coloneqq \mathbf{U}\mathbf{T}^{-1}$, where $\mathbf{T} = \begin{bmatrix}\mathbf{G} \\ \mathbf{R}\end{bmatrix} \in \mathbb{F}_p^{n\times n}$ is a public full-rank matrix and $\mathbf{G} \in \mathbb{F}_p^{m\times n}$ is the encoding matrix of an MDS code, $\mathbf{R} = [\mathbf{0}\ \mathbf{I}] \in \mathbb{F}_p^{(n-m)\times n}$ ($\mathbf{0}$ is the $(n-m)\times m$ zero matrix and $\mathbf{I}$ is the $(n-m)\times(n-m)$ identity matrix) $\mathbf{U}_1, \mathbf{U}_2 \in \mathbb{F}_p^{\ell\times m}$, $\mathbf{C} \in \mathbb{F}_p^{2\ell\times(n-m)}$. $\mathcal{P}$ sends $\mathbf{C}$ to $\mathcal{V}$.

Online $\mathsf{com}(\mathbf{X}) \leftarrow \mathsf{Com}(\mathbf{X})$**:** ($\mathbf{X}$ is known, $\mathbf{A}, a$ are not)

$\mathcal{P}$ sends $\mathbf{X}' \coloneqq \mathbf{X} - \mathbf{U}_1$ to $\mathcal{V}$.

Online $b \leftarrow \mathsf{Prove}(\mathbf{X}, \mathsf{com}_{\mathbf{X}}, \mathbf{A}, a)$**:** ($\mathbf{X}, \mathbf{A}, a$ are ready)

(1) $\mathcal{P}$ sends $h_{\mathbf{A}\circ\mathbf{U}_2} \coloneqq \mathtt{Hash}(\mathbf{A} \circ \mathbf{U}_2)$ to $\mathcal{V}$.

(2) $\mathcal{V}$ samples $\alpha \leftarrow \mathbb{F}_p$ and sends it to $\mathcal{P}$.

(3) Let $\begin{bmatrix}\mathbf{V}_1 \\ \mathbf{V}_2\end{bmatrix} = \mathbf{V}$ where $\mathbf{V}_1, \mathbf{V}_2 \in \mathbb{F}_p^{\ell\times n}$. $\mathcal{P}$ sends $\mathbf{S} \coloneqq \alpha\mathbf{U}_1 + \mathbf{U}_2$ and $h_{\alpha\mathbf{V}_1+\mathbf{V}_2} \coloneqq \mathtt{Hash}(\alpha\mathbf{V}_1 + \mathbf{V}_2)$ to $\mathcal{V}$.

(4) $\mathcal{V}$ samples $\mathbf{d} = (d_0, \ldots, d_{n-1})$ where $d_i \leftarrow [N]$ and sends $\mathbf{d}$ to $\mathcal{P}$. $\mathcal{V}$ sets $\mathbf{D} \coloneqq \mathsf{diag}\left(\{s_{d_j}\}_{j\in[n]}\right)$.

(5) $\forall j \in [n]$, $\mathcal{P}$ opens the j^{th} GGM-tree except its ${d_j}^{th}$ leaf.

(6) $\forall i \in [2\ell], j \in [n], k \in [N]\backslash\{d_j\}, \mathcal{V}$ computes $t_{i,j,k} \coloneqq \mathtt{PRG}(i, \mathtt{seed}_{j,k})$ and $q_{i,j} \coloneqq \sum_{k=0}^{N-1}(s_{d_j} - s_k)\cdot t_{i,j,k}$. Let $\mathbf{Q} \in \mathbb{F}_p^{2\ell\times n}$ be the matrix whose entries are $q_{i,j}$. $\mathcal{V}$ computes $\begin{bmatrix}\mathbf{Q}_1 \\ \mathbf{Q}_2\end{bmatrix} \coloneqq \mathbf{Q} - \mathbf{CRD}$ where $\mathbf{Q}_1, \mathbf{Q}_2 \in \mathbb{F}_p^{\ell\times n}$. $\mathcal{V}$ aborts if any of the following equality checks fail:

$$h_{\alpha\mathbf{V}_1+\mathbf{V}_2} = \mathtt{Hash}(\alpha\mathbf{Q}_1 + \mathbf{Q}_2 - \mathbf{SGD}) \quad (6)$$

$$h_{\mathbf{A}\circ\mathbf{U}_2} = \mathtt{Hash}(\alpha a + \mathbf{A} \circ (\mathbf{S} + \alpha\mathbf{X}')) \quad (7)$$

$\mathcal{V}$ outputs 1 if and only if all checks above pass.

Figure 5: VitH ZKP for Linear Relations on Large Fields

"Large fields" implies: (1) $p \geq N$ so all s_k are distinct values of $\mathbb{F}_p$; (2) p is sufficiently large to offer meaningful soundness (see the proof of Theorem 3 for more details on this).

5.1. Basic Ideas

A properly distributed random VOLE $d\mathbf{u}+\mathbf{v} = \mathbf{q}$ where $\mathcal{P}$ holds $\mathbf{u}, \mathbf{v} \in \mathbb{F}^n$ and $\mathcal{V}$ holds $d \in \mathbb{F}, \mathbf{q} \in \mathbb{F}^n$ such that $\mathbf{q}$ can be viewed as a commitment of $\mathcal{P}$'s random vector $\mathbf{u}$ to $\mathcal{V}$. Given such a random VOLE and a secret vector $\mathbf{w}$, it is easy for $\mathcal{P}$ to commit $\mathbf{w}$ by sending $\mathbf{w}' \coloneqq \mathbf{w} - \mathbf{u}$ while $\mathcal{V}$ setting $\mathbf{q}' \coloneqq d\mathbf{w}' + \mathbf{q}$ and holding $d, \mathbf{q}'$ as a commitment of $\mathbf{w}$ because $d\mathbf{w} + \mathbf{v} = \mathbf{q}'$. Note that this commitment is linear-homomorphic over $\mathbb{F}$, so given the commitment of $\mathbf{w}$, it is straightforward to prove any linear function f of secret $\mathbf{w}$, which is exactly what we needed to finish proving the linear constraints in Phecda.

It is not hard to generate properly distributed random VOLE $(u, v; d, q)$. Let N be an integer security parameter. $\mathcal{P}$ picks uniform $\{t_k \in \mathbb{F}\}_{k\in[N]}$ and sets

$$u \coloneqq \sum_{k\in[N]} t_k; \qquad v \coloneqq -\sum_{k\in[N]} s_k \cdot t_k$$

where $\{s_k \in \mathbb{F}\}_{k\in[N]}$ are public distinct non-zero field elements, among which $\mathcal{V}$ randomly pick one, say s_d, as its secret. If $\mathcal{V}$ could somehow learn all $\{t_k\}_{k\neq d}$, then she can compute

$$q = \sum_{k\neq d}(s_d - s_k)\cdot t_k = \sum_{k\in[N]}(s_d - s_k)\cdot t_k = s_d \cdot u + v.$$

As $\mathcal{P}$ doesn't know d while $\mathcal{V}$ doesn't know u and v due to t_d, a VOLE tuple would be properly distributed to $\mathcal{P}$ and $\mathcal{V}$, except that a malicious $\mathcal{P}$ has $1/N$ chance to correctly guess $\mathcal{V}$'s s_d and cheat accordingly (we will come back to this with a satisfactory prevention mechanism soon).

Observe that even before $\mathcal{V}$ learns $\{t_k\}_{k\neq d}$ and q, $\mathcal{P}$ can use u, v to commit secret values by applying u as an additive mask to the value being committed. In addition, the revelation of $\{t_k\}_{k\neq d}$ can be delayed until later proving constraints about the committed values, at which point $\mathcal{V}$ simply picks a uniform $d \in [N]$ and reveals it to $\mathcal{P}$, who then sends back $\{t_k\}_{k\neq d}$ for $\mathcal{V}$ to compute q, as well as checking $\mathcal{P}$'s integrity in preparing $\{t_k\}_{k\in[N]}$.

In fact, $\mathcal{P}$ can derive $\{t_k\}_{k\in[N]}$ from a single seed using a GGM tree so that it costs only $O(\log N)$ bandwidth to reveal $\{t_k\}_{k\neq d}$ [13]. In the vectorizing form, this procedure is repeated n times to generate $s_{\mathbf{d}} * \mathbf{u} + \mathbf{v} = \mathbf{q}$ where $\mathbf{d} = \{d_i \in [N]\}_{i\in[n]}$, $s_{\mathbf{d}} = \{s_{d_i}\}_{i\in[n]} \in \mathbb{F}^n$, $\mathbf{u} = \{u_i\}_{i\in[n]}$, and "$*$" denotes element-wise product. Hence $(s_{\mathbf{d}}, \mathbf{q})$ can be held as a commitment of the vector $\mathbf{u}$. Since linear relations involving the n secrets are just dot-products of a public vector and the secret vector, it is easy to prove them given the commitment of the secret vector.

Recall that $(s_{\mathbf{d}}, \mathbf{q})$ as a commitment of $\mathbf{u}$ has $1/N$ binding error because if $\mathcal{P}$ guessed any entry of $s_{\mathbf{d}}$ correctly, the commitment can be opened to $(\mathbf{u}', \mathbf{v}') \neq (\mathbf{u}, \mathbf{v})$. To improve soundness, $\mathcal{P}$ can commit an error correction code $\mathbf{u}\cdot\mathbf{G}$ of $\mathbf{u}$ where $\mathbf{G}$ is a public encoding matrix. In particular, we use linear MDS codes, which offer maximal code distance possible given their codeword and message lengths. If the minimal distance of the MDS code is δ symbols, then now

the binding error is reduced to $1/N^\delta$ because a malicious $\mathcal{P}$ who tampers $\mathbf{u}$ has to correctly guess at least δ entries of $s_{\mathbf{d}}$ to remain undetected. However, since a uniform random $\mathbf{u}$ is not necessarily a codeword, $\mathcal{P}$ needs to send δ correction symbols in the beginning to coerce it to a valid codeword. Also, $\mathcal{V}$ will check a masked codeword in the end to make sure a valid codeword was indeed committed earlier.

While the secret witness $\mathbf{w}$ can be long, one expects the encoding matrix $\mathbf{G}$ to produce relatively-short fixed-length codewords. To this end, we treat $\mathbf{w}$ as an $\ell \times n$ matrix $\mathbf{U}$ where the number of columns (each corresponding to a GGM-tree) n can be a small constant whereas $\ell = |\mathbf{w}|/n$. For all $i \in [\ell]$, $t_{i,j,k}$ can be easily generated as $\mathtt{PRG}(i, seed_{j,k})$ where $seed_{j,k}$ is stored in the k^{th} leaf of the j^{th} tree. Now the MDS code encodes every row of $\mathbf{U}$ by computing $\mathbf{UG}$. This decouples the length of the secret from the length of the codewords.

However, two additional challenges arise: (1) since $\mathbf{d}$ is a vector, it can only form VOLE relation with columns of the witness matrix but not the whole witness matrix; (2) a linear relation that involves *all* entries of a matrix of secrets may not always be expressible as matrix multiplications. Instead, what we need is a "$\circ$" operator such that $\mathbf{A} \circ \mathbf{X}$ denotes the sum of products between corresponding entries of a public coefficients matrix $\mathbf{A}$ and the secret witness matrix $\mathbf{X}$. To cope with these challenges, we are inspired by [15] to generate a second VOLE of same-size matrices $\mathbf{U}_2, \mathbf{V}_2, \mathbf{Q}_2$ so that the correctness of both VOLEs can be verified by checking a random combination (through α) of $(\mathbf{U}_1, \mathbf{V}_1, \mathbf{Q}_1)$ and $(\mathbf{U}_2, \mathbf{V}_2, \mathbf{Q}_2)$.

Putting things in matrix form, let $\mathbf{U}, \mathbf{D}, \mathbf{V}, \mathbf{Q}$ be as defined in Figure 5, specifically $\mathbf{U} = \begin{bmatrix} \mathbf{U}_1 \\ \mathbf{U}_2 \end{bmatrix} \begin{bmatrix} \mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{G} \\ \mathbf{R} \end{bmatrix}$. So the equation $\mathbf{UD} + \mathbf{V} = \mathbf{Q}$ can be re-written as

$$\begin{bmatrix} \mathbf{U}_1 \\ \mathbf{U}_2 \end{bmatrix} \begin{bmatrix} \mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{G} \\ \mathbf{R} \end{bmatrix} \mathbf{D} + \begin{bmatrix} \mathbf{V}_1 \\ \mathbf{V}_2 \end{bmatrix} = \mathbf{Q} \tag{8}$$

where $\mathbf{C}$ is the correction matrix, $\mathbf{G} \in \mathbb{F}_p^{m \times n}$ is the MDS encoding matrix, and $\mathbf{R} \in \mathbb{F}_p^{(n-m) \times n}$ can be set to $\begin{bmatrix} \mathbf{0} & \mathbf{I} \end{bmatrix}$ where $\mathbf{0}$ is the $(n-m) \times m$ 0-matrix and $\mathbf{I}$ is the $(n-m) \times (n-m)$ identity matrix, so that $\begin{bmatrix} \mathbf{G} \\ \mathbf{R} \end{bmatrix} \in \mathbb{F}_p^{n \times n}$ is a public full-rank matrix. If Equation (8) does not hold, then it will be detected with probability $1 - 1/|\mathbb{F}|$ when $\mathcal{V}$ checks Equation (6), which is essentially the top row of Equation (8) multiplied by a uniform α chosen by $\mathcal{V}$ and added to the bottom row.

Otherwise, Equation (8) holds. Let $\mathbf{S} = \alpha \mathbf{U}_1 + \mathbf{U}_2$ be the linear combination received by $\mathcal{V}$. Then, unless $\alpha = 0$, checking linear constraint $\mathbf{A} \circ \mathbf{X} + a = 0$ is equivalent to checking

$$\begin{aligned} 0 &= \alpha(\mathbf{A} \circ \mathbf{X} + a) = \alpha \mathbf{A} \circ (\mathbf{X}' + \mathbf{U}_1) + \alpha a \\ &= \alpha \mathbf{A} \circ \mathbf{X}' + \alpha \mathbf{A} \circ \mathbf{U}_1 + \alpha a \\ &= \alpha \mathbf{A} \circ \mathbf{X}' + \mathbf{A} \circ (\mathbf{S} - \mathbf{U}_2) + \alpha a \\ &= \alpha \mathbf{A} \circ \mathbf{X}' + \mathbf{A} \circ \mathbf{S} + \alpha a - \mathbf{A} \circ \mathbf{U}_2 \end{aligned}$$

which is essentially what Equation (7) in Figure 5 verifies.

5.2. The Protocol

Specification. Our full protocol, specified in Figure 5, consists of an offline preparation and an online phase. The interfaces provided in the online phase include: (1) $\mathsf{com}_{\mathbf{X}} \leftarrow \mathsf{Commit}(\mathbf{X})$, where $\mathbf{X}$ is the secret witnesses involved in a set of linear constraints to be proved later; and (2) $b \leftarrow \mathsf{Prove}(\mathbf{X}, \mathsf{com}_{\mathbf{X}}, \mathbf{A}, a)$, which proves a committed $\mathbf{X}$ satisfy $\mathbf{A} \circ \mathbf{X} + a = 0$, where function $\circ : \mathbb{F}_p^{\ell \times m} \times \mathbb{F}_p^{\ell \times m} \to \mathbb{F}_p$ computes the sum of element-wise products of two matrices.

Unlike [15], our protocol does not rely on $\mathbb{F}_p^l$-hiding ϵ-almost universal family of hashes, and has fewer checks and uses fewer rounds, hence need a fresh proof of security. We use a $[n, m, p]$ linear MDS code that encodes m-symbol messages to n-symbol codewords ($m < n$) where symbols are $\mathbb{F}_p$ elements. In practice, this can be efficiently realized with Reed-Solomon codes.

To save bandwidth, we simply check the hashes (rather than preimage matrices) on both sides of Equation (6) and Equation (7). This strategy is especially helpful because, in practice, often many constraints over the same set of committed secrets need to be proved. That is, we need to realize $\mathsf{Prove}\left(\mathsf{com}_{\mathbf{X}}, \{\mathbf{A}_0, \ldots, \mathbf{A}_{M-1}\}, \{a_0, \ldots, a_{M-1}\}\right)$ for some large integer M, to prove $\mathbf{A}_i \circ \mathbf{X} + a_i = 0, \forall i \in [M]$. To this end, we need to replicate Equation (7) M times, one for each $\mathbf{A}_i, a_i$. However, the M equalities can be checked via sending a single hash of the concatenation of all "$\mathbf{A}_i \circ \mathbf{U}_2$"s. Finally, as Equation (6) remains unchanged, its cost can be amortized over proving M constraints.

Although our discussion was concentrated on proving linear relations, the above ideas can be easily generalized to prove arbitrary degree polynomials over committed secrets. Assume a vector $\mathbf{x}$ of secrets is committed via VOLE, i.e., $d\mathbf{x} + \mathbf{v} = \mathbf{q}$ with $\mathcal{P}$ knows $\mathbf{x}$, $\mathbf{v}$ and $\mathcal{V}$ knows d, $\mathbf{q}$. As suggested by Quicksilver [16], one can prove any equation $f(\mathbf{x}) = 0$ where f is a n-variate degree-d polynomial for an $\mathbf{x} \in \mathbb{F}^n$ committed earlier, through checking

$$\sum_{h \in [\mathsf{d}+1]} g_h(\mathbf{q}) \cdot d^{\mathsf{d}-h} = \sum_{h \in [\mathsf{d}]} a_h \cdot d^h \tag{9}$$

where $\{g_h\}_{h \in [\mathsf{d}+1]}$ are the degree-h monomials of f such that $f(\mathbf{x}) = \sum_{h \in [\mathsf{d}+1]} g_h(\mathbf{x})$, and $\{a_h\}_{h \in [\mathsf{d}]}$ are efficiently computable by $\mathcal{P}$ from the coefficients of f and $\mathbf{u}, \mathbf{v}$. After $\mathcal{P}$ sends a_h masked by random VOLEs, $\mathcal{V}$ can compute both sides of Equation (9) and verify the equality.

Security. The security of our VOLE-in-the-Head protocol is stated and proved as Theorem 3 below.

Theorem 3. The protocol given in Figure 5 is an honest-verifier zero-knowledge proof of knowledge for linear relations $\mathbf{A} \circ \mathbf{X} + a = 0$ where $\mathbf{X}$ is $\mathcal{P}$'s secret witness and $\mathbf{A}, a$ are public constants.

Proof. Completeness. It is easy to verify that if $\mathbf{A} \circ \mathbf{X} + a = 0$, then all checks in the protocol will pass hence an honest $\mathcal{V}$ will accept.

Soundness. We define the following events

$$\texttt{Check1Pass} : h_{\alpha\mathbf{V}_1+\mathbf{V}_2} = \texttt{Hash}(\alpha\mathbf{Q}_1 + \mathbf{Q}_2 - \mathbf{SGD})$$
$$\texttt{Check2Pass} : h_{\mathbf{A}\circ\mathbf{U}_2} = \texttt{Hash}(\alpha a + \mathbf{A}\circ(\mathbf{S} + \alpha\mathbf{X}'))$$
$$\texttt{B} : \mathbf{A}\circ\mathbf{X} + a \neq 0$$

To keep things simpler, in our analysis below we ignore the event of collisions on collision-resistant hashes, which only occurs with negligible probability. The soundness error of our protocol can be expressed as

$$\varepsilon = \Pr[\texttt{Check1Pass} \wedge \texttt{Check2Pass} \mid \texttt{B}]$$

Let $E_{\mathbf{C}}, E_{\mathbf{S}}, E_{\mathbf{A}\circ\mathbf{U}_2}, E_{\alpha\mathbf{V}_1+\mathbf{V}_2}$ be the additive differences between their respective "suppose-to-be" values an honest prover would have used and those corresponding to actual messages sent by a potentially dishonest prover. E.g., $E_{\mathbf{C}} = \hat{\mathbf{C}} - \mathbf{C}$ where $\mathbf{C}$ is sent by the honest prover whereas $\hat{\mathbf{C}}$ is sent by a potentially cheating prover. In addition, note that the values are fixed in our protocol in the order: $E_{\mathbf{C}} \to \mathbf{X}' \to E_{\mathbf{A}\circ\mathbf{U}_2} \to \alpha \to \{E_{\mathbf{S}}, E_{\alpha\mathbf{V}_1+\mathbf{V}_2}\} \to \mathbf{D}$.

Let $\begin{bmatrix} E_{\mathbf{C}_1} \\ E_{\mathbf{C}_2} \end{bmatrix} = E_{\mathbf{C}}$ for $E_{\mathbf{C}_1}, E_{\mathbf{C}_2} \in \mathbb{F}_p^{\ell\times(n-m)}$. By Claim 1 and Claim 2, unless a collision occurs on the hash, `Check1Pass` and `Check2Pass` are equivalent to the following two equations:

$$E_{\alpha\mathbf{V}_1+\mathbf{V}_2} = (-\alpha E_{\mathbf{C}_1}\mathbf{R} - E_{\mathbf{C}_2}\mathbf{R} - E_{\mathbf{S}}\mathbf{G}_C)\mathbf{D} \quad (10)$$
$$E_{\mathbf{A}\circ\mathbf{U}_2} = \alpha a + \alpha\mathbf{A}\circ\mathbf{X} + \mathbf{A}\circ E_{\mathbf{S}} \quad (11)$$

where values of $\mathbf{X}$, $E_{\mathbf{A}\circ\mathbf{U}_2}$, $E_{\mathbf{S}}$, $E_{\alpha\mathbf{V}_1+\mathbf{V}_2}$ are of the malicious prover's choice. By Lemma 2, the attack strategy given in Figure 6 is *optimal* for a malicious prover to find $E_{\mathbf{C}}$, $\mathbf{X}$, $E_{\mathbf{A}\circ\mathbf{U}_2}$, $E_{\mathbf{S}}$, $E_{\alpha\mathbf{V}_1+\mathbf{V}_2}$ to pass the checks above.

With the optimal strategy, since $\begin{bmatrix} E_{\mathbf{C}_1} \\ E_{\mathbf{C}_2} \end{bmatrix}\mathbf{R} = \begin{bmatrix} \mathbf{B}_1 \\ \mathbf{B}_2 \end{bmatrix}\mathbf{G}$, Equation (10) can be rewritten as

$$E_{\alpha\mathbf{V}_1+\mathbf{V}_2} = (-\alpha\mathbf{B}_1 - \mathbf{B}_2 - E_{\mathbf{S}})\mathbf{G}_C\mathbf{D} \quad (12)$$

The success rate of the optimal strategy can be derived by a case analysis:

(1) When $\alpha' = \alpha$ (which occurs with probability $1/p$), the assigned values of $E_{\alpha\mathbf{V}_1+\mathbf{V}_2}, E_{\mathbf{S}}$ ensure that Equation (12), thus Equation (10), will hold, while the assignment of $E_{\mathbf{A}\circ\mathbf{U}_2}$ ensures Equation (11) will hold.

(2) When $\alpha' \neq \alpha$, it is easy to verify that the way $E_{\mathbf{A}\circ\mathbf{U}_2}$ and $E_{\mathbf{S}}$ are picked guarantees Equation (11) will hold, and that the way $E_{\alpha\mathbf{V}_1+\mathbf{V}_2}$ is picked guarantees Equation (12), hence Equation (10), will hold if and only if relevant entries of $\mathbf{D}'$ are guessed correctly.
Note that it should be impossible for a malicious prover to find an $E_{\mathbf{S}}$ such that $\alpha\mathbf{B}_1 + \mathbf{B}_2 + E_{\mathbf{S}} = \mathbf{0}$ while satisfying both Equation (10) and Equation (11), because otherwise, a malicious prover, who does not know the witness $\mathbf{X}$ such that $\mathbf{A}\circ\mathbf{X} + a = 0$ *before* $\mathbf{A}$ and a are revealed, would be able to learn a witness through running the optimal attacking strategy above. Specifically, a malicious prover would learn $\mathbf{B} = \begin{bmatrix} \mathbf{B}_1 \\ \mathbf{B}_2 \end{bmatrix}$, $\mathbf{X}$, $E_{\mathbf{A}\circ\mathbf{U}_2} = \alpha' c + \alpha'\mathbf{A}\circ\mathbf{X} + \mathbf{A}\circ(-\alpha'\mathbf{B}_1 - \mathbf{B}_2)$, $E_{\mathbf{S}} = -\alpha\mathbf{B}_1 - \mathbf{B}_2$ satisfying Equation (11), i.e.,

$$\alpha' a + \alpha'\mathbf{A}\circ\mathbf{X} + \mathbf{A}\circ(-\alpha'\mathbf{B}_1 - \mathbf{B}_2) = \alpha a + \alpha\mathbf{A}\circ\mathbf{X} + \mathbf{A}\circ(-\alpha\mathbf{B}_1 - \mathbf{B}_2)$$

hence, $(\alpha' - \alpha)(a + \mathbf{A}\circ\mathbf{X}) = (\alpha' - \alpha)\mathbf{A}\circ\mathbf{B}_1$. Since $\alpha' - \alpha \neq 0$, therefore $\mathbf{A}\circ(\mathbf{X} - \mathbf{B}_1) + c = \mathbf{0}$. Since $\mathbf{X}$ and $\mathbf{B}_1$ were sent before ma and c are known, the malicious prover essentially knows a solution to the equation $\mathbf{A}\circ\mathbf{X} + a = 0$, which contradicts to the assumption that the malicious prover does not know a solution to $\mathbf{A}\circ\mathbf{X} + a = 0$ before learning $\mathbf{A}$ and a.
Now that $\alpha\mathbf{B}_1 + \mathbf{B}_2 + E_{\mathbf{S}} \neq 0$, $(\alpha\mathbf{B}_1 + \mathbf{B}_2 + E_{\mathbf{S}})\mathbf{G}$ must have at least δ non-zero columns. Therefore, a malicious prover has to correctly guess δ entries in $\mathbf{D}'$ to pass the check of Equation (12), an event occurring with probability at most $1/N^{\delta}$.

Overall, the success rate of an attacker's optimal strategy is at most $1/p + (1 - 1/p)/N^{\delta} = 1 - (1 - 1/p)(1 - 1/N^{\delta})$.

Knowledge Soundness. It is easy to verify that our protocol is $\left(\binom{n}{n-\delta+1} + 1, N^{n-\delta} + 1\right)$-*special sound*: Given any $\left(\binom{n}{n-\delta+1} + 1, N^{n-\delta} + 1\right)$-tree of accepting transcripts, one must be able to efficiently recover the witness $\mathbf{X}$:

Goal: Attack the Figure 5 protocol as a malicious prover. That is, without the witness $\mathbf{X}$ before knowing some $\mathbf{A}_i$, b_i, find appropriate values of $E_{\mathbf{C}}$, $\mathbf{X}$, $E_{\mathbf{A}\circ\mathbf{U}_2}$, $E_{\mathbf{S}}$, $E_{\alpha\mathbf{V}_1+\mathbf{V}_2}$ when they are needed in running the Figure 5 protocol with an honest verifier to pass all verification checks.

(1) Pick $E_{\mathbf{C}}$ so that every row of $E_{\mathbf{C}}\mathbf{R}$ is a valid codeword, i.e,. $\exists\mathbf{B} \in \mathbb{F}_p^{\ell\times m}$, such that $E_{\mathbf{C}}\mathbf{R} = \mathbf{BG}$.
(2) Pick any $\mathbf{X}$ from $\mathbb{F}_p^{\ell\times m}$.
(3) Guess α. Let the guess be α'. Set

$$E_{\alpha\mathbf{V}_1+\mathbf{V}_2} \coloneqq \mathbf{0}$$
$$E_{\mathbf{S}} \coloneqq -\alpha'\mathbf{B}_1 - \mathbf{B}_2$$
$$E_{\mathbf{A}\circ\mathbf{U}_2} \coloneqq \alpha' a + \alpha'\mathbf{A}\circ\mathbf{X} + \mathbf{A}\circ E_{\mathbf{S}} = \alpha' a + \alpha'\mathbf{A}\circ\mathbf{X} + \mathbf{A}\circ(-\alpha'\mathbf{B}_1 - \mathbf{B}_2)$$

where $\mathbf{B}_1 = E_{\mathbf{C}_1}\mathbf{R}$, $\mathbf{B}_2 = E_{\mathbf{C}_2}\mathbf{R}$.
(4) After receiving α, if $\alpha' \neq \alpha$, guess relevant entries[a] of $\mathbf{D}$ with respect to the codeword matrix $(E_{\mathbf{S}} + \alpha'\mathbf{B}_1 + \mathbf{B}_2)\mathbf{G}$. Let $\mathbf{D}'$ be the guess of $\mathbf{D}$. Pick $E_{\mathbf{S}}$ to satisfy

$$\mathbf{A}\circ E_{\mathbf{S}} = (\alpha' - \alpha)(a + \mathbf{A}\circ\mathbf{X}) - \mathbf{A}\circ(\alpha'\mathbf{B}_1 + \mathbf{B}_2)$$

then set

$$E_{\alpha\mathbf{V}_1+\mathbf{V}_2} \coloneqq (-\alpha\mathbf{B}_1 - \mathbf{B}_2 - E_{\mathbf{S}})\mathbf{GD}'$$

a. Considering multiplying a matrix $\mathbf{M}$ with a diagonal matrix $\mathbf{D}$, we call the i^{th} entry of $\mathbf{D}$ *relevant* if the i^{th} column of $\mathbf{M}$ is non-zero.

Figure 6: Optimal Attack Strategy for a Malicious Prover

(1) Recall from the proof of Lemma 2 that given $\mathbf{C}$, $\mathbf{X}'$, and $E_{\mathbf{A}\circ\mathbf{U}_2}$, there are at most $\binom{n}{n-\delta+1}$ possible values of α allowing a malicious prover to pass the checks. Since the first layer of the transcripts-tree has $\binom{n}{n-\delta+1}+1$ nodes, at least one node corresponds to the case $d \geq \delta$ (i.e., the "otherwise" case in the proof of Lemma 2).
(2) Since for every first-layer node of the transcripts-tree, there are $(N^{n-\delta}+1)$ second-layer nodes, which allow to *fully* recover at least $n-\delta+1$ GGM-trees. Thus, the $(N^{n-\delta}+1)$ second-layer nodes corresponding to a first-layer node associated with a $d \geq \delta$ case can guarantee the recovery of $\mathbf{U}_1$ (because any $n-\delta+1$ columns of $\mathbf{U}_1$ allows to uniquely decode $\mathbf{U}_1$), hence recovering $\mathbf{X}$.

Therefore, By [21, Theorem 1], the knowledge error of our protocol is at most

$$\begin{aligned}&\frac{N_1N_2-\left(N_1-\binom{n}{n-\delta+1}\right)\left(N_2-N^{n-\delta}\right)}{N_1N_2}\\ =&\frac{\binom{n}{n-\delta+1}}{N_1}+\frac{N^{n-\delta}}{N_2}-\frac{\binom{n}{n-\delta+1}N^{n-\delta}}{N_1N_2}\\ =&\frac{\binom{n}{n-\delta+1}}{p}+\frac{1}{N^\delta}-\frac{\binom{n}{n-\delta+1}}{pN^\delta}.\end{aligned}$$

Honest-Verifier Zero-Knowledge. Given an arbitrary non-witness $\mathbf{X}$, it is easy to use a simulator to generate an accepting transcript as follows:

(1) Pick a uniform α and $\mathbf{d}$.
(2) Compute $\mathbf{C}$, $\mathbf{X}'$, $\mathbf{S}$ exactly as specified in our protocol.
(3) Set $h_{\alpha\mathbf{V}_1+\mathbf{V}_2} \coloneqq \mathtt{Hash}(\alpha\mathbf{Q}_1+\mathbf{Q}_2-\mathbf{SGD})$.
(4) Set $h_{\mathbf{A}\circ\mathbf{U}_2} \coloneqq \mathtt{Hash}(\alpha a+\mathbf{A}_i\circ(\mathbf{S}+\alpha\mathbf{X}'))$.
(5) Output $(\mathbf{C},\mathbf{X}',h_{\mathbf{A}\circ\mathbf{U}_2},\alpha,\mathbf{S},h_{\alpha\mathbf{V}_1+\mathbf{V}_2},\mathbf{d})$ and the "all-but-one" opening messages of GGM-trees as transcript.

It is easy to verify that the distribution of transcripts produced by the simulator is identical to that produced by a legitimate prover interacting with an honest verifier. □

Lemma 2. The strategy given in Figure 6 offers a malicious prover the optimal success rate in convincing an honest verifier any false statement using Figure 5 protocol.

Proof. Let d be the number of non-zero columns in $(-\alpha E_{\mathbf{C}_1}\mathbf{R}-E_{\mathbf{C}_2}\mathbf{R}-E_{\mathbf{S}}\mathbf{G})$, the matrix to be multiplied with $\mathbf{D}$ in Equation (10). In other words, d is the number of different columns between $E_{\mathbf{S}}\mathbf{G}$ and $-(\alpha E_{\mathbf{C}_1}\mathbf{R}+E_{\mathbf{C}_2}\mathbf{R})$. The probability that a malicious prover $\mathcal{P}'$ finds $E_{\mathbf{S}}, E_{\alpha\mathbf{V}_1+\mathbf{V}_2}$ (before knowing $\mathbf{D}$) to pass Equation (10) check is $1/N^d$ since each entry of the diagonal matrix $\mathbf{D}$ is uniformly picked from N candidate $\mathbb{F}_p$ elements.

Regardless of how $\mathcal{P}'$ picks $E_{\mathbf{C}}$, $\mathbf{X}$, and $E_{\mathbf{A}\circ\mathbf{U}_2}$, once $\mathcal{P}'$ receives a uniform α, we can analyze $\mathcal{P}$'s attack success rate by examining two cases below:

(1) **There exists an $E_{\mathbf{S}}$ such that $d \leq \delta-1$:** To derive an upper-bound on the attacker's success rate, we will count, for a given d, the number of different α values that can pass Equation (10) check with probability $1/N^d$ while always passing Equation (11).

It is easy to check the following facts are all equivalent:

a) $\exists E_{\mathbf{S}}$: $\begin{cases} d \leq \delta-1 \\ E_{\mathbf{A}\circ\mathbf{U}_2}=\alpha a+\mathbf{A}\circ(\alpha\mathbf{X}+E_{\mathbf{S}})\end{cases}$

b) $\exists \boldsymbol{\pi} \in \{(\pi_0,\pi_1,\ldots,\pi_{n-\delta}) \mid \pi_i \in [n], \pi_i<\pi_{i+1}\}$: $\begin{cases}(E_{\mathbf{S}}\mathbf{G})[\boldsymbol{\pi}]=(-\alpha E_{\mathbf{C}_1}\mathbf{R}-E_{\mathbf{C}_2}\mathbf{R})[\boldsymbol{\pi}] \\ E_{\mathbf{A}\circ\mathbf{U}_2}=\alpha a+\mathbf{A}\circ(\alpha\mathbf{X}+E_{\mathbf{S}})\end{cases}$

c) $\exists \boldsymbol{\pi}$: $\begin{cases}E_{\mathbf{S}}(\mathbf{G}[\boldsymbol{\pi}])=-\alpha(E_{\mathbf{C}_1}\mathbf{R})[\boldsymbol{\pi}]-(E_{\mathbf{C}_2}\mathbf{R})[\boldsymbol{\pi}] \\ E_{\mathbf{A}\circ\mathbf{U}_2}=\alpha a+\mathbf{A}\circ(\alpha\mathbf{X}+E_{\mathbf{S}})\end{cases}$

d) $\exists \boldsymbol{\pi}$: $\begin{cases}E_{\mathbf{S}}=-\alpha\mathbf{M}_{1,\boldsymbol{\pi}}-\mathbf{M}_{2,\boldsymbol{\pi}} \\ E_{\mathbf{A}\circ\mathbf{U}_2}=\alpha a+\mathbf{A}\circ(\alpha\mathbf{X}+E_{\mathbf{S}})\end{cases}$
where $\mathbf{M}_{1,\boldsymbol{\pi}}=(E_{\mathbf{C}_1}\mathbf{R})[\boldsymbol{\pi}](\mathbf{G}[\boldsymbol{\pi}])^{-1}$,
and $\mathbf{M}_{2,\boldsymbol{\pi}}=(E_{\mathbf{C}_2}\mathbf{R})[\boldsymbol{\pi}](\mathbf{G}[\boldsymbol{\pi}])^{-1}$

e) $\exists \boldsymbol{\pi}$: $\begin{cases}E_{\mathbf{S}}=-\alpha\mathbf{M}_{1,\boldsymbol{\pi}}-\mathbf{M}_{2,\boldsymbol{\pi}} \\ E_{\mathbf{A}\circ\mathbf{U}_2}=\alpha a+\mathbf{A}\circ(\alpha\mathbf{X}-\alpha\mathbf{M}_{1,\boldsymbol{\pi}}-\mathbf{M}_{2,\boldsymbol{\pi}})\end{cases}$

f) $\exists \boldsymbol{\pi}$: $\begin{cases}E_{\mathbf{S}}=-\alpha\mathbf{M}_{1,\boldsymbol{\pi}}-\mathbf{M}_{2,\boldsymbol{\pi}} \\ \alpha=\dfrac{E_{\mathbf{A}\circ\mathbf{U}_2}+\mathbf{A}\circ\mathbf{M}_{2,\boldsymbol{\pi}}}{a+\mathbf{A}\circ(\mathbf{X}-\mathbf{M}_{1,\boldsymbol{\pi}})}\end{cases}$

The property of MDS code guarantees that $\mathbf{G}[\boldsymbol{\pi}]^{-1}$ exists for any $\boldsymbol{\pi}$. Also note that $a+\mathbf{A}\circ(\mathbf{X}-\mathbf{M}_{E_{\mathbf{C}_1}\mathbf{R}}) \neq 0$; otherwise, $\mathcal{P}'$ would have known a witness to $\mathbf{A}\circ\mathbf{X}+a=0$ before knowing $\mathbf{A}$ and a because in our protocol $\mathbf{X}$ and $E_{\mathbf{C}}$, hence $\mathbf{M}_{E_{\mathbf{C}}\mathbf{R}}$, are fixed before $\mathbf{A}, a$. Therefore, once $E_{\mathbf{C}}$ is sent, all satisfying $\mathbf{M}_{1,\boldsymbol{\pi}}$, $\mathbf{M}_{2,\boldsymbol{\pi}}$, and α are fixed, and each selection of the $(n-\delta+1)$ columns (represented by $\boldsymbol{\pi}$) is associated with certain values for $\mathbf{M}_{1,\boldsymbol{\pi}}$, $\mathbf{M}_{2,\boldsymbol{\pi}}$, and α. Let $k_{d,\mathbf{M}}$ (resp. $k_{d,\alpha}$) be the number of distinct $(\mathbf{M}_{1,\boldsymbol{\pi}},\mathbf{M}_{2,\boldsymbol{\pi}})$ pairs (resp. α values) for a fixed d value. Then,

$$\begin{aligned}k_{d,\alpha} &\leq k_{d,\mathbf{M}}\\ \binom{n}{n-\delta+1} &= \sum_{d=0}^{\delta-1} k_{d,\mathbf{M}}\binom{n-d}{n-\delta+1}\\ &\geq \sum_{d=0}^{\delta-1} k_{d,\alpha}\binom{n-d}{n-\delta+1}.\end{aligned}$$

Since α and $\mathbf{D}$ are selected uniform random, the probability that $\mathcal{P}'$ can succeed in this case is $\sum_{d=0}^{\delta-1}\frac{k_{d,\alpha}}{p}\frac{1}{N^d}$.

(2) **Otherwise:** There does not exist any $E_{\mathbf{S}}$ such that $d \leq \delta-1$. So the probability $\mathcal{P}'$ succeeds is at most $\left(1-\left(\sum_{d=0}^{\delta-1}k_{d,\alpha}\right)\Big/p\right)\frac{1}{N^\delta}$.

Overall, $\varepsilon(k_{0,\alpha},\ldots,k_{\delta-1,\alpha})$, the attack success rate of $\mathcal{P}'$, is at most

$$\sum_{d=0}^{\delta-1}\frac{k_{d,\alpha}}{p}\frac{1}{N^d}+\left(1-\frac{\left(\sum_{d=0}^{\delta-1}k_{d,\alpha}\right)}{p}\right)\frac{1}{N^\delta}$$

where the $k_{d,\alpha}$'s are constrained by

$$\sum_{d=0}^{\delta-1} k_{d,\alpha} \binom{n-d}{n-\delta+1} \leq \binom{n}{n-\delta+1} \tag{13}$$

and reflect the attack strategy of $\mathcal{P}'$. Claim 3 shows this $\varepsilon(k_{0,\alpha}, \ldots, k_{\delta-1,\alpha})$ is maximized when $k_{d,\alpha} = 0 \; \forall d < \delta - 1$ and $k_{\delta-1,\alpha} = 1$, which corresponds exactly to the optimal attack strategy given in Figure 6. □

Claim 1. Unless a collision is found on `Hash`,

$$\texttt{Check1Pass} \Leftrightarrow E_{\alpha\mathbf{V}_1+\mathbf{V}_2} = (-\alpha E_{\mathbf{C}_1}\mathbf{R} - E_{\mathbf{C}_2}\mathbf{R} - E_{\mathbf{S}}\mathbf{G})\mathbf{D}.$$

Proof. In order for a malicious prover to pass the `Check1Pass` check, the prover is essentially choosing $\hat{\mathbf{C}}$, $\widetilde{\alpha\mathbf{V}_1 + \mathbf{V}_2}$, and $\widetilde{\mathbf{S}}$ to simultaneously satisfy the following two equations:

$$\begin{bmatrix}\hat{\mathbf{Q}}_1\\ \hat{\mathbf{Q}}_2\end{bmatrix} = \mathbf{Q} - \hat{\mathbf{C}}\mathbf{R}\mathbf{D}$$
$$\widetilde{\alpha\mathbf{V}_1 + \mathbf{V}_2} = \alpha\hat{\mathbf{Q}}_1 + \hat{\mathbf{Q}}_2 - \widetilde{\mathbf{S}}\mathbf{G}\mathbf{D}$$

Since $\widetilde{\alpha\mathbf{V}_1 + \mathbf{V}_2} = (\alpha\mathbf{V}_1 + \mathbf{V}_2) + E_{\alpha\mathbf{V}_1+\mathbf{V}_2}$, so

$$(\alpha\mathbf{V}_1 + \mathbf{V}_2) + E_{\alpha\mathbf{V}_1+\mathbf{V}_2} = \alpha\hat{\mathbf{Q}}_1 + \hat{\mathbf{Q}}_2 - \widetilde{\mathbf{S}}\mathbf{G}\mathbf{D}$$

Recall that

$$\begin{bmatrix}\mathbf{U}_0 & \mathbf{C}\end{bmatrix}\begin{bmatrix}\mathbf{G}\\ \mathbf{R}\end{bmatrix} = \mathbf{U}$$
$$\mathbf{Q} = \mathbf{U}\mathbf{D} + \mathbf{V}$$
$$\begin{bmatrix}\mathbf{Q}_1\\ \mathbf{Q}_2\end{bmatrix} = \mathbf{Q} - \mathbf{C}\mathbf{R}\mathbf{D}$$

Hence,

$$\begin{aligned}\mathbf{Q} &= (\mathbf{U}_0\mathbf{G} + \mathbf{C}\mathbf{R})\mathbf{D} + \mathbf{V}\\ \begin{bmatrix}\mathbf{Q}_1\\ \mathbf{Q}_2\end{bmatrix} &= (\mathbf{U}_0\mathbf{G} + \mathbf{C}\mathbf{R})\mathbf{D} + \mathbf{V} - \mathbf{C}\mathbf{R}\mathbf{D}\\ &= \mathbf{U}_0\mathbf{G}\mathbf{D} + \mathbf{V}\\ &= \begin{bmatrix}\mathbf{U}_1\\ \mathbf{U}_2\end{bmatrix}\mathbf{G}\mathbf{D} + \begin{bmatrix}\mathbf{V}_1\\ \mathbf{V}_2\end{bmatrix}\\ \begin{bmatrix}\hat{\mathbf{Q}}_1\\ \hat{\mathbf{Q}}_2\end{bmatrix} &= (\mathbf{U}_0\mathbf{G} + \mathbf{C}\mathbf{R})\mathbf{D} + \mathbf{V} - \hat{\mathbf{C}}\mathbf{R}\mathbf{D}\\ &= (\mathbf{U}_0\mathbf{G} + \mathbf{C}\mathbf{R})\mathbf{D} + \mathbf{V} - (\mathbf{C} + E_{\mathbf{C}})\mathbf{R}\mathbf{D}\\ &= \mathbf{U}_0\mathbf{G}\mathbf{D} + \mathbf{V} - E_{\mathbf{C}}\mathbf{R}\mathbf{D}\\ &= \begin{bmatrix}\mathbf{U}_1\\ \mathbf{U}_2\end{bmatrix}\mathbf{G}\mathbf{D} + \begin{bmatrix}\mathbf{V}_1\\ \mathbf{V}_2\end{bmatrix} - \begin{bmatrix}E_{\mathbf{C}_1}\\ E_{\mathbf{C}_2}\end{bmatrix}\mathbf{R}\mathbf{D}\end{aligned}$$

Hence,

$$\begin{aligned}(\alpha\mathbf{V}_1 + \mathbf{V}_2) + E_{\alpha\mathbf{V}_1+\mathbf{V}_2} &= \alpha(\mathbf{U}_1\mathbf{G}\mathbf{D} + \mathbf{V}_1 - E_{\mathbf{C}_1}\mathbf{R}\mathbf{D})\\ &\quad + (\mathbf{U}_2\mathbf{G}\mathbf{D} + \mathbf{V}_2 - E_{\mathbf{C}_2}\mathbf{R}\mathbf{D}) - \widetilde{\mathbf{S}}\mathbf{G}\mathbf{D}\end{aligned}$$
$$E_{\alpha\mathbf{V}_1+\mathbf{V}_2} = \alpha(\mathbf{U}_1\mathbf{G}\mathbf{D} - E_{\mathbf{C}_1}\mathbf{R}\mathbf{D}) + (\mathbf{U}_2\mathbf{G}\mathbf{D} - E_{\mathbf{C}_2}\mathbf{R}\mathbf{D}) - \widetilde{\mathbf{S}}\mathbf{G}\mathbf{D}$$
$$E_{\alpha\mathbf{V}_1+\mathbf{V}_2} = \left(\alpha(\mathbf{U}_1\mathbf{G} - E_{\mathbf{C}_1}\mathbf{R}) + (\mathbf{U}_2\mathbf{G} - E_{\mathbf{C}_2}\mathbf{R}) - \widetilde{\mathbf{S}}\mathbf{G}\right)\mathbf{D}$$

Because $\widetilde{\mathbf{S}} = \mathbf{S} + E_{\mathbf{S}} = (\alpha\mathbf{U}_1 + \mathbf{U}_2) + E_{\mathbf{S}}$, therefore

$$E_{\alpha\mathbf{V}_1+\mathbf{V}_2} = (-\alpha\hat{\mathbf{C}}_1\mathbf{R} - \hat{\mathbf{C}}_2\mathbf{R} - E_{\mathbf{S}}\mathbf{G})\mathbf{D}$$

□

Claim 2. Unless a collision is found on `Hash`,

$$\texttt{Check2Pass} \Leftrightarrow E_{\mathbf{A}\circ\mathbf{U}_2} = \alpha a + \alpha\mathbf{A}\circ\mathbf{X} + \mathbf{A}\circ E_{\mathbf{S}}.$$

Proof. In order for the malicious prover to pass the `Check2Pass` check, the prover need to send values of $\widetilde{\mathbf{A}\circ\mathbf{U}_2}$, $\widetilde{\mathbf{S}}$, $\mathbf{X}'$ to satisfy the equation:

$$\widetilde{\mathbf{A}\circ\mathbf{U}_2} = \alpha a + \mathbf{A}\circ(\widetilde{\mathbf{S}} + \alpha\mathbf{X}')$$

Since $\widetilde{\mathbf{A}\circ\mathbf{U}_2} = \mathbf{A}\circ\mathbf{U}_2 + E_{\mathbf{A}\circ\mathbf{U}_2}$, and $\widetilde{\mathbf{S}} = \mathbf{S} + E_{\mathbf{S}}$, the equation above is equivalent to

$$\mathbf{A}\circ\mathbf{U}_2 + E_{\mathbf{A}\circ\mathbf{U}_2} = \alpha a + \mathbf{A}\circ(\mathbf{S} + E_{\mathbf{S}} + \alpha\mathbf{X}')$$

Substituting $\mathbf{S} = \alpha\mathbf{U}_1 + \mathbf{U}_2$, we obtain

$$\begin{aligned}\mathbf{A}\circ\mathbf{U}_2 + E_{\mathbf{A}\circ\mathbf{U}_2} &= \alpha a + \mathbf{A}\circ(\alpha\mathbf{U}_1 + \mathbf{U}_2 + E_{\mathbf{S}} + \alpha\mathbf{X}')\\ E_{\mathbf{A}\circ\mathbf{U}_2} &= \alpha a + \mathbf{A}\circ(\alpha\mathbf{U}_1 + E_{\mathbf{S}} + \alpha\mathbf{X}')\end{aligned}$$

Recall that $\mathbf{X} = \mathbf{U}_1 + \mathbf{X}'$. Therefore, it is equivalent to picking $E_{\mathbf{A}\circ\mathbf{U}_2}$, $E_{\mathbf{S}}$, and $\mathbf{X}$ such that

$$E_{\mathbf{A}\circ\mathbf{U}_2} = \alpha a + \mathbf{A}\circ(\alpha\mathbf{X} + E_{\mathbf{S}})$$

□

Claim 3. $\varepsilon(k_{0,\alpha}, \ldots, k_{\delta-1,\alpha})$ is maximized when $k_{0,\alpha} = k_{1,\alpha} = \cdots = k_{\delta-2,\alpha} = 0$ and $k_{\delta-1,\alpha} = 1$, i.e.

$$\varepsilon(k_{0,\alpha}, \ldots, k_{\delta-1,\alpha}) \leq \varepsilon(0, \ldots, 0, 1) = \frac{1}{p} + \left(1 - \frac{1}{p}\right)\frac{1}{N^\delta}.$$

Proof. Because $\varepsilon = \sum_{d=0}^{\delta-1} \frac{k_{d,\alpha}}{p}\frac{1}{N^d} + \left(1 - \frac{\sum_{d=0}^{\delta-1} k_{d,\alpha}}{p}\right)\frac{1}{N^\delta}$,

$$\therefore pN^\delta\varepsilon - p = \sum_{d=0}^{\delta-1} k_{d,\alpha}\left(N^{\delta-d} - 1\right)$$
$$\therefore pN^\delta\varepsilon - p = \sum_{d=0}^{\delta-1} k_{d,\alpha}\binom{n-d}{n-\delta+1} \cdot \frac{N^{\delta-d}-1}{\binom{n-d}{n-\delta+1}}$$

Since the function $f(d) = \left(N^{\delta-d} - 1\right)/\binom{n-d}{n-\delta+1}$ is strictly decreasing for all $0 \leq d < \delta < n$ (Claim 4), so $f(d) < f(0) = \left(N^\delta - 1\right)/\binom{n}{n-\delta+1}$. Combining with the constraint on $k_{d,\alpha}$, we have

$$pN^\delta\varepsilon - p < \binom{n}{n-\delta+1} \cdot \frac{N^\delta - 1}{\binom{n}{n-\delta+1}} = N^\delta - 1$$
$$\therefore \varepsilon < \frac{1}{p} + \left(1 - \frac{1}{p}\right)\frac{1}{N^\delta}.$$

□

Claim 4. $f(d) = \left(N^{\delta-d}-1\right)/\binom{n-d}{n-\delta+1}$ is strictly decreasing for all $0 \le d < \delta < n$ and $N \ge 2$.

Proof. Because $f(d) > 0$ for the ranges of d, δ, n, N that we consider, the monotonicity of $f(d)$ can be established from examining $f(d)/f(d+1)$ for all $d \le \delta-2$, i.e., $\delta-d-1 \ge 1$.

$$\begin{aligned}\frac{f(d)}{f(d+1)} &= \frac{N^{\delta-d}-1}{N^{\delta-(d+1)}-1} \cdot \frac{\binom{n-(d+1)}{n-\delta+1}}{\binom{n-d}{n-\delta+1}} \\ &= \frac{N^{\delta-d}-1}{N^{\delta-(d+1)}-1} \cdot \frac{\delta-d-1}{n-d} \\ &= \left(N + \frac{1}{N^{\delta-(d+2)}+\cdots+N+1}\right) \cdot \frac{\delta-d-1}{n-d} \\ &> N \cdot \frac{\delta-d-1}{n-d} \ge N \cdot \frac{1}{n-d} \ge N \cdot \frac{1}{n-(\delta-2)} \\ &\ge N \cdot \frac{1}{n-\delta+2} > N \cdot \frac{1}{2} \ge 1.\end{aligned}$$

□

Costs. Let I be the number of secret witness and M be the number of constraints. Note that in our protocol $I = O(\ell m) = O(\ell n)$. We analyze the costs of our VOLE-based ZKP as follows.

Prover Time. The prover needs $O(NI)$ time to compute $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{C}$, $O(MI)$ time to compute $h_{\mathbf{A}\circ\mathbf{U}_2}$, $O(I)$ time to compute $\mathbf{X}'$ and $\mathbf{S}$, and $O(nN)$ time to generate and reveal GGM-trees, thus $O(NI + MI + nN)$ overall prover time.

Verifier Time. The verifier needs $O(NI)$ time to compute $\mathbf{Q}$, $O(I)$ time to compute $\mathbf{CRD}$, $O(nN)$ time to recover the GGM-trees, $O(I)$ time to verify Equation (6), and $O(MI)$ time to verify Equation (7). The overall verifier time is also $O(NI + MI + nN)$.

Communication. Let $|h|$, σ be the lengths of a hash and a seed, resp, both of which are small constants. We choose an MDS code with small code rate $c = n/m$ (e.g., $c = 2$). So $|\pi|$ is

$$\begin{aligned}&|\mathbf{C}| + |\mathbf{X}'| + |\mathbf{S}| + 2|h| + (\text{commit and open } n \text{ GGM-trees}) \\ =&(2\ell(n-m) + 2m\ell)\log p + 2|h| + (|h| + n\sigma\log N + n\sigma) \\ =&2n\ell\log p + 3|h| + n\sigma\log N + n\sigma \\ =&2n\frac{I}{m}\log p + 3|h| + n\sigma\log N + n\sigma \\ =&2cI\log p + 3|h| + cm\sigma\log N + cm\sigma \\ =&O(I\log p + m\log N)\end{aligned}$$

Comparison with KKW-Lin. Dubhe's KKW-Lin, the best prior MPC-in-the-Head protocol specialized from [9] to prove linear constraints, requires $O(nN\cdot(I+MI+1))$-time on both the prover and the verifier, and produces a proof of length $O(n(I+1)\log p + |h| + n\sigma\log N)$, to achieve $1-1/N^n$ soundness. We summarize Phecda's VOLE-in-the-Head asymptotic advantages over KKW-Lin in Table 2. When setting $n = 2m$, Phecda has soundness error $1/N^\delta = 1/N^{n-m+1} = 1/N^{m+1}$. In this setting, Phecda's parameter m is comparable to KKW-Lin's n.

TABLE 2: Phecda's VOLE-in-the-Head vs KKW-Lin

	KKW-Lin	**Phecda's VOLEitH**
$\mathcal{P}$, $\mathcal{V}$	$O(nNI + nNMI + nN))$	$O(NI + MI + mN)$
$\lvert\pi\rvert$	$O(nI\log p + n\log N)$	$O(I\log p + m\log N)$
Soundness	$1-1/N^n$	$1-1/N^\delta$

6. Verifiable AES

AES, approved by NIST [22], is one of the most widely used standard symmetric-key cipher. Efficient zkSNARK for AES computation promise various useful applications such as post-quantum digital signatures, verifiable symmetric encryption and verifiable pseudorandom permutation. In this section, we focus on efficiently proving a large number of AES blocks, e.g., in the counter mode of operation.

To prove many blocks of AES in the interactive setting, Ding and Huang [5] proposed two AES circuits, one on byte-field and the other on bit-field. Their byte-field based scheme allowed faster prover but requires linear verifier and communication, whereas the bit-field based scheme, which involves 41 rounds of challenges, has sublinear verifier and communication but significantly slower prover, thus impractical to be turned into a non-interactive argument via parallel repetition.

In this work, we propose a novel ZK argument for AES offering the best of both previous schemes. It allows a quasi-linear prover, sublinear verifier and sublinear proof size. It is also concretely faster in prover and verifier times, uses much less bandwidth while requiring no interaction.

To avoid the big constant factor when the computation is specified on $\mathrm{GF}(2)$ (as shown in [5]), we wish to use the byte-field circuit like the witness-based AES circuit from Dubhe [5] but commit and prove all the witness using a polynomial commitment scheme. The challenge resides in the linear operations in MixColumns and SubByte, which are defined with respect to two different modulus. It was not known how to efficiently support both modulo computations in a single circuit through a succinct PC. Dubhe relied on KKW-Lin to prove these linear operations in MixColumns and the affine transform in SubByte, but resulted in linear communication and verification time.

A Novel Approach to Verify AES. To address this challenge, we propose a new AES verification circuit specifically tailored for batch proving many AES blocks. The new circuit is defined on a larger binary extension field $\mathrm{GF}(2^N)$ which allows to efficiently emulate operations of smaller embedded binary extension fields, such as $\mathrm{GF}(2^8)$ utilized by AES. While all additions in an AES circuit over $\mathrm{GF}(2^N)$ can be easily supported with GKR, the three types of gates that need special treatment are:

SubByte**'s special inverse:** i.e., "$a\bullet b = 1 \vee (a = 0 \wedge b = 0)$" for all $a, b \in \mathrm{GF}(2^8)$ with "$\bullet$" defined modulo `0x011B`.

SubByte**'s affine transform:** i.e., "$a * \texttt{0x1F} \oplus \texttt{0x63} = b$" for all $a, b \in \mathrm{GF}(2^8)$ with ring operator "$*$" defined modulo `0x0101`.

MixColumns**'s constant multiplication:** i.e., "$\texttt{0x02} \cdot a = b$" and "$\texttt{0x03} \cdot a = b$" for all $a, b \in \mathrm{GF}(2^8)$ with "$\cdot$" defined modulo $\texttt{0x011B}$.

We developed a method to translate all these operations into efficient range checks, by applying the Emulation Lemma.

We can interpret a (with binary form $a_n \dots a_0$) as representing the polynomial $a_n \cdot x^n + \cdots + a_1 \cdot x + a_0$. Therefore, we use $\mathsf{deg}(a)$ to denote the degree of the binary-coefficient polynomial represented by a. For example, $\mathsf{deg}(1001) = 3$. In cases where the exact value of a is unknown, $\mathsf{deg}(a)$ represents the maximum possible degree of the corresponding polynomial. For example, if a denotes any value of $\mathrm{GF}(2^8)$, then $\mathsf{deg}(a) = 7$.

Emulation Lemma. Let $\star$ denote the multiplication in an n-bit ring, and $\times$ denote the multiplication of an N-bit binary extension field with $N \geq 2n-1$. Let m be the modulus defined by the ring and m^{-1} be m's inverse in the N-bit field. Let $\oplus$ be bit-wise XOR. For all n-bit values a, b, c,

$$a \star b = c \iff (a \times b \oplus c) \times m^{-1} \in [2^{k+1}]$$

where $k = \mathsf{deg}(a) + \mathsf{deg}(b) - \mathsf{deg}(m)$.

Proof. Let $\circ$ denote multiplication in the infinite-degree binary-coefficient polynomial ring. Then $a \star b = c \iff c = a \circ b \mod m$. Since $\mathsf{deg}(a \circ b) = \mathsf{deg}(a) + \mathsf{deg}(b)$, by the definition of infinite-degree polynomial ring division, "$c = a \circ b \mod m$" is equivalent to "$\exists r$ such that $a \circ b = r \circ m \oplus c \ \wedge\ \mathsf{deg}(r) \leq \mathsf{deg}(a) + \mathsf{deg}(b) - \mathsf{deg}(m)$", which is equivalent to "$\exists r$ such that $a \times b = r \times m \oplus c \ \wedge\ \mathsf{deg}(r) \leq \mathsf{deg}(a) + \mathsf{deg}(b) - \mathsf{deg}(m)$ since N is sufficiently large to avoid "overflow" as a result of the ring multiplication. The latter is essentially "$\exists r$ such that $(a \times b \oplus c) \times m^{-1} = r$ and $\mathsf{deg}(r) \leq \mathsf{deg}(a) + \mathsf{deg}(b) - \mathsf{deg}(m)$" when $N > \mathsf{deg}(a) + \mathsf{deg}(b)$. That is, "$(a \times b \oplus c) \times m^{-1} \in [2^{k+1}]$ with $k = \mathsf{deg}(a) + \mathsf{deg}(b) - \mathsf{deg}(m)$. □

Thus, applying this lemma, the check in MixColumns, "$a, b \in [2^8] \wedge \texttt{0x02} \cdot a = b$" becomes "$a, b \in [2^8] \wedge (\texttt{0x02} \times a \oplus b) \times \texttt{0x011B}^{-1} \in [2]$" since $k = \mathsf{deg}(\texttt{0x02}) + \mathsf{deg}(a) - \mathsf{deg}(\texttt{0x011B}) = 1 + 7 - 8 = 0$ while $\texttt{0x03} \cdot a = b \Leftrightarrow \texttt{0x02} \cdot a \oplus a = b$; the affine transformation check in SubByte, "$a, b \in [2^8] \wedge a * \texttt{0x1F} \oplus \texttt{0x63} = b$" becomes "$a, b \in [2^8] \wedge (a \times \texttt{0x1F} \oplus b \oplus \texttt{0x63}) \times \texttt{0x0101}^{-1} \in [2^4]$" since $k = \mathsf{deg}(a) + \mathsf{deg}(\texttt{0x1F}) - \mathsf{deg}(\texttt{0x0101}) = 7 + 4 - 8 = 3$.

Checking the special inverse for SubByte is a bit more complex. The inverse is special since the output byte b for an input byte a is defined as a^{-1} modulo $\texttt{0x011B}$ if $a \neq 0$; or 0 if $a = 0$. To realize the $a \neq 0$ case, we use Emulation Lemma to translate "$a, b \in [2^8] \wedge a \cdot b = 1$" into "$a, b \in [2^8] \wedge (a \times b \oplus 1) \times \texttt{0x011B}^{-1} \in [2^7]$" (since here $k = \mathsf{deg}(a) + \mathsf{deg}(b) - \mathsf{deg}(\texttt{0x011B}) = 7 + 7 - 8 = 6$). In case $a = 0$, the check is $a = 0 \wedge b = 0$. Like Dubhe, the check combining both cases becomes "$a, b \in [2^8] \wedge (a \times b \oplus 1) \times \texttt{0x011B}^{-1} \in [2^7] \vee a = 0 \wedge b = 0$", which can be translated to a more GKR-friendly expression: "$a, b \in [2^8] \wedge x = \mathrm{RangeCheck}\left((a \times b \oplus 1) \times \texttt{0x011B}^{-1}, [2^7]\right) \wedge x \times a = 0 \wedge x \times b = 0$" where $\mathrm{RangeCheck}(a, R)$ returns 0 if $a \in R$, and a non-zero field element if $a \notin R$. We will present our range-checking method in more detail shortly.

In summary, our AES circuit is defined on large finite fields $\mathrm{GF}(2^N)$ with $N > 14$. For 128-bit computational security and computer architecture reasons, we set $N = 192$. The key schedule for AES128 involves 40 SubBytes, thus need $2 \cdot 40 = 80$ witness elements and 80 range checks: 2 for each SubByte as it has 1 special inverse and 1 affine transform. To prove the rest of the AES128, 464 witnesses and 464 range checks are needed: 320 for the 160 SubBytes and $16 \cdot 9 = 144$ for the MixColumnss. All other parts of the circuit are linear operations that are easily handled by GKR. The circuit for m AES blocks can be done with 10 layers, with $80 + 464m$ witnesses and $80 + 464m$ range checks.

TABLE 3: Prover Time (ms) of Range Checks

$\log \lvert Range \rvert$	4	8	12	16	20	24
Traditional	5.92	39.0	47.2	102	509	6390
Proposed	3.50	12.9	16.1	22.3	28.4	35.9

Efficient Range Checking using Algebra. Our new AES verification circuit requires many range checks to ensure some secret values are within a specific range. The traditional way to realize this is through checking $\Pi_{i \in Range}(x - i) = 0$, which can be implemented as a tree of multiplications and proved in ZK. However, observing the similarity between the left-side of the equality above and the vanishing polynomial $\Pi_{i \in \mathbb{H}}(x - i)$, we notice that the ranges to prove is exactly those upon which vanishing polynomials are defined [23]. Therefore, we can leverage the structure of the underlying cosets over binary extension fields to compute the series of products recursively, using only $\log |Range|$ multiplications [24]. Let $\mathbb{H}_i = [2^i]$ and $z_i = \Pi_{a \in \mathbb{H}_i}(x - a)$, then $z_0(x) = x$ and $z_{i+1}(x) = z_i(x) \cdot (z_i(x) - z_i(\beta_i))$ where β_i are the standard basis of the subspace $\mathbb{H}_i$. When $|Range|$ is small, like in AES, it is easy to prove these multiplications sequentially. In case $|Range|$ is large, zkGKR can be applied to further reduce the verifier's time and bandwidth to $O(\log \log |Range|)$. Table 3 compares the proving time of range-checking per element, assuming $\mathrm{GF}(2^{192})$ field is used. For a witness of 20-bit or more, the proposed optimization is more than an order-of-magnitude faster.

7. Evaluation

We implemented Phecda in Rust and run experiments in the transparent zkSNARK setting, on a Linux PC with Intel i9-11900H CPU and 64GB DDR4 Memory.

7.1. Verifiable AES

Figure 7 gives the costs of Phecda on realizing verifiable AES128 in the counter-mode and compares them with several baseline implementations. The closest prior work is from Dubhe, which proposed two methods to verify

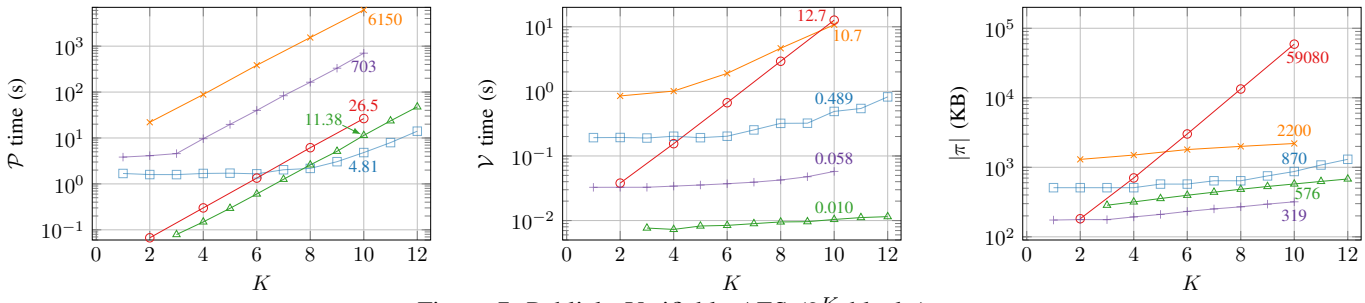

Figure 7: Publicly Verifiable AES (2^K blocks)

Phecda GF(2^{256}) 128-bit security, Dubhe (no extra-witness, scalable, 128-bit security), Dubhe (extra-witness, 128-bit security), Jolt/Lasso (Hyrax, ∼90-bit security), Virgo (∼90-bit security)

TABLE 4: Cost Breakdown of Phecda (1024 AES blocks)

%	GKR	FLPCP	VOLEitH		PC	
			commit	prove	commit	prove
$\mathcal{P}$ time	39.27	0.027	0.042	0.054	20.99	39.62
$\mathcal{V}$ time	2.21	29.04	–	56.15	–	12.72
$\|\pi\|$	–	–	22.75	1.14	0.005	76.10

AES computation: one using linear extra-witness (200 bytes per AES block, i.e., 1 byte per SubByte call) and the other without extra-witness but proving each SubByte as 8 bit-level table-lookups. Our new protocol exhibits obvious advantage in verification time. For 1024 AES blocks, Phecda offers 2.33× faster prover, 1270× faster verifier, and 102× smaller proof size, than Dubhe's extra-witness based scheme; while offering 540× faster prover, 1070× faster verifier, and 3.8× smaller proof size, than Dubhe's more scalable scheme without using extra-witnesses. The cost breakdown of the sub-protocols are given in Table 4. Note that current implementations of Lasso [25] and Virgo only offer about 90-bit computational security and the Lasso implementation is not quantum-resistant due to its use of elliptic curves. So comparisons with these two schemes are not perfectly aligned.

7.2. Polynomial Commitment

Table 5 shows the concrete improvements of our batched LDT protocol in the context of proving computations over binary extension fields. Without our batched LDT, to make sure all f_i's ($\forall i \in [n]$) are of low degrees where n is the number of variables in the multi-linear polynomial, the best existing method will need to make n independent LDT calls. The asymptotic improvement is reflected by the experiments, with proof size and verifier cost improved by 65–70% and prover cost improved by 14–21%.

Compare with Virgo. We only compare our PC with Virgo's on a squared Mersenne prime field (Table 6) since Virgo does not support binary fields. On polynomials of

TABLE 5: Improvement Ratios of mLDT

log |Input|	13	15	17	19	21
$\mathcal{P}$ faster	1.21×	1.19×	1.17×	1.16×	1.14×
$\mathcal{V}$ faster	1.65×	1.70×	1.68×	1.68×	1.71×
$\|\pi\|$ smaller	1.66×	1.70×	1.72×	1.75×	1.77×

TABLE 6: Our PC vs Virgo's on GF($(2^{127}-1)^2$)

log |Input|		13	15	17	19	21
$\mathcal{P}$ (s)	Virgo	0.15	0.63	2.92	12.90	55.73
	Phecda	0.12	0.48	2.16	9.88	43.16
$\mathcal{V}$ (ms)	Virgo	0.89	1.31	1.36	1.85	1.97
	Phecda	0.82	1.21	1.24	1.71	1.80
$\|\pi\|$ (KB)	Virgo	235	304	385	472	568
	Phecda	217	283	356	438	528

2^{13} to 2^{21} secret coefficients, our PC prover (resp. verifier) runs 25–35% (resp. 8.3–9.7%) faster than Virgo's, while our PC's proof sizes are only 92–93% of Virgo's. An improved algorithm for proving FFT can further reduce Virgo's prover time by roughly 4%–6% but won't help with verification time and proof size [26]. It is unclear if that improved algorithm works for fields over additive cosets.

Compare with Orion. Table 7 compares the costs of our PC with Orion [27]. Experiments were only run on GF $\left((2^{61}-1)^2\right)$ for 90–100-bit conjectured security (roughly 60-bit provable security) because this is the only field available in Orion's implementation. On polynomials of 2^{15} to 2^{21} secret coefficients, our prover runs 6.5–15.7× slower, while our verifier runs 25.8–44.2× faster and our proof size is 11.6–18.5× smaller than Orion's.

TABLE 7: Our PC vs Orion's on GF($(2^{61}-1)^2$)

log |Input|		15	17	19	21
$\mathcal{P}$ (ms)	Orion	31.8	81.0	286.2	1089
	Phecda	205.8	917.6	3992	17139
$\mathcal{V}$ (ms)	Orion	12.8	15.2	21.8	42.5
	Phecda	0.497	0.615	0.763	0.961
$\|\pi\|$ (KB)	Orion	2576	2699	2850	3023
	Phecda	139.4	175.7	216.1	260.7

TABLE 8: VOLEitH vs MPCitH in Verifying 2^K AES

K		4	6	8	10	12
$\mathcal{P}$ (ms)	MPCitH	37.0	40.1	52.4	57.0	64.8
	VOLEitH	7.3	8.5	9.6	10.2	12.0
$\mathcal{V}$ (ms)	MPCitH	24.2	25.4	29.9	33.6	35.5
	VOLEitH	7.0	8.1	9.2	10.1	11.1
$\lvert\pi\rvert$ (KB)	MPCitH	361	406	452	497	543
	VOLEitH	100	114	127	138	151

Compare with Brakedown. Brakedown [28] is another multi-linear polynomial commitment scheme with concrete costs similar to those of Orion (see [27, Fig. 3] for detailed data points). Brakedown is field-agnostic and features linear (and concretely faster) prover, though its verification time and proof size are asymptotically worse than Orion and ours.

7.3. VOLE-in-the-Head

Compare with MPCitH. Table 8 shows the performance advantages of VOLEitH over MPCitH. In the context of proving linear relations for verifiable AES, VOLEitH is able to beat MPCitH in all three respects of performance measures by a significant gap. Specifically, VOLEitH's prover, verifier, and proof size are roughly 5.5×, 3.3×, and 3.6× better than MPCitH's.

8. Conclusion

Phecda improves best existing GKR-based zero-knowledge proof systems with a new polynomial commitment scheme and a new linear constraints proof scheme. Using a novel verification circuit for AES, Phecda is able to offer unprecedented performance for publicly verifiable AES-based encryption or pseudorandom permutation.

Acknowledgments

We thank the anonymous reviewers of IEEE S&P for their helpful comments.

References

[1] T. Xie, J. Zhang, Y. Zhang, C. Papamanthou, and D. Song, "Libra: Succinct zero-knowledge proofs with optimal prover computation," in *CRYPTO*, 2019.

[2] R. S. Wahby, I. Tzialla, A. Shelat, J. Thaler, and M. Walfish, "Doubly-efficient zkSNARKs without trusted setup," in *IEEE Symposium on Security and Privacy*, 2018.

[3] J. Zhang, T. Xie, Y. Zhang, and D. Song, "Transparent polynomial delegation and its applications to zero knowledge proof," in *IEEE Symposium on Security and Privacy*, 2020.

[4] J. Zhang, T. Liu, W. Wang, Y. Zhang, D. Song, X. Xie, and Y. Zhang, "Doubly efficient interactive proofs for general arithmetic circuits with linear prover time," in *ACM CCS*, 2021.

[5] C. Ding and Y. Huang, "Dubhe: Succinct zero-knowledge proofs for standard aes and related applications," in *USENIX Security*, 2023.

[6] S. Goldwasser, Y. T. Kalai, and G. N. Rothblum, "Delegating computation: interactive proofs for muggles," in *STOC*, 2008.

[7] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev, "Fast reed-solomon interactive oracle proofs of proximity," in *ICALP*, 2018.

[8] D. Boneh, E. Boyle, H. Corrigan-Gibbs, N. Gilboa, and Y. Ishai, "Zero-knowledge proofs on secret-shared data via fully linear pcps," in *CRYPTO*, 2019.

[9] J. Katz, V. Kolesnikov, and X. Wang, "Improved non-interactive zero knowledge with applications to post-quantum signatures," in *ACM CCS*, 2018.

[10] G. Cormode, M. Mitzenmacher, and J. Thaler, "Practical verified computation with streaming interactive proofs," in *Proceedings of Innovations in Theoretical Computer Science Conference*, 2012.

[11] J. Thaler, "Time-optimal interactive proofs for circuit evaluation," in *CRYPTO*, 2013.

[12] J. Bootle, A. Chiesa, Y. Hu, and M. Orru, "Gemini: Elastic snarks for diverse environments," in *CRYPTO*, 2022.

[13] L. Roy, "SoftSpokenOT: Quieter OT extension from small-field silent vole in the minicrypt model," in *CRYPTO*. Springer, 2022.

[14] B. Chen, B. Bünz, D. Boneh, and Z. Zhang, "Hyperplonk: Plonk with linear-time prover and high-degree custom gates," in *EUROCRYPT*, 2023.

[15] C. Baum, L. Braun, C. D. de Saint Guilhem, M. Klooß, E. Orsini, L. Roy, and P. Scholl, "Publicly verifiable zero-knowledge and post-quantum signatures from vole-in-the-head," in *CRYPTO*, 2023.

[16] K. Yang, P. Sarkar, C. Weng, and X. Wang, "Quicksilver: Efficient and affordable zero-knowledge proofs for circuits and polynomials over any field," in *ACM CCS*, 2021.

[17] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *CRYPTO*, 1986.

[18] C. Lund, L. Fortnow, H. Karloff, and N. Nisan, "Algebraic methods for interactive proof systems," *Journal of the ACM (JACM)*, vol. 39, no. 4, pp. 859–868, 1992.

[19] A. Chiesa, M. A. Forbes, and N. Spooner, "A zero knowledge sumcheck and its applications," *arXiv:1704.02086*, 2017.

[20] E. Ben-Sasson, D. Carmon, Y. Ishai, S. Kopparty, and S. Saraf, "Proximity gaps for reed–solomon codes," in *FOCS*, 2020.

[21] T. Attema, R. Cramer, and L. Kohl, "A compressed Σ-protocol theory for lattices," in *CRYPTO*, 2021.

[22] NIST, "Federal information processing standards publication FIPS-197, advanced encryption standard," 1999.

[23] N. P. Byott and R. J. Chapman, "Power sums over finite subspaces of a field," *Finite Fields and Their Applications*, 1999.

[24] O. Ore, "On a special class of polynomials," *Transactions of the American Mathematical Society*, 1933.

[25] S. Setty, J. Thaler, and R. Wahby, "Unlocking the lookup singularity with lasso," in *EUROCRYPT*, 2024.

[26] T. Liu, X. Xie, and Y. Zhang, "Zkcnn: Zero knowledge proofs for convolutional neural network predictions and accuracy," in *ACM CCS*, 2021.

[27] T. Xie, Y. Zhang, and D. Song, "Orion: Zero knowledge proof with linear prover time," in *CRYPTO*, 2022.

[28] A. Golovnev, J. Lee, S. T. Setty, J. Thaler, and R. S. Wahby, "Brakedown: Linear-time and post-quantum snarks for r1cs." *CRYPTO*, 2024.

Appendix A.
Meta-Review

The following meta-review was prepared by the program committee for the 2025 IEEE Symposium on Security and Privacy (S&P) as part of the review process as detailed in the call for papers.

A.1. Summary

This paper introduces a new framework for constructing transparent (zk-)SNARKs with plausible post-quantum security. The construction is based on a combination of the GKR protocol, VOLE-in-the-head protocols, fully linear PCPs, and a novel multi-linear polynomial commitment scheme. The authors consider verifiable encryption (AES) as a concrete application and provide detailed experimental evaluation, comparison with prior works, and publicly-available source code, empirically demonstrating significant concrete performance improvement in terms of prover time, verifier time, and proof size.

A.2. Scientific Contributions

- Provides a Valuable Step Forward in an Established Field

A.3. Reasons for Acceptance

(1) This paper provides a valuable step forward in an established field. It gives a zkSNARK that is transparent and plausibly post-quantum secure and has competitive concrete efficiency in terms of prover and verifier time, along with proof sizes, built from known and new novel technical building blocks.