Revisiting Virgo: A Study of Vulnerabilities, Limitations, and Optimizations

Abstract

This paper revisits Virgo, a well-known transparent zero-knowledge proof system that has been used in many subsequent studies. Through our analysis, we uncover previously overlooked limitations and several exploitable security vulnerabilities within Virgo's zkVPD protocol design and implementation. We subsequently address these issues and improve Virgo's zkVPD protocol. Our improvements feature simplified but more efficient VPD and zkVPD algorithms, offering enhanced support for computations over binary fields and their extension fields.

Keywords: ZK proof, transparent zkSNARK, polynomial commitments (zkVPD)

1 Introduction

Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (zkSNARK) is a useful cryptographic primitive that finds many real-world applications. A *transparent* zkSNARK further offers *public verifiability*, allowing anyone to verify the validity of a statement without any *a priori* setup, thus making it most attractive. In this paper, we revisit Virgo [1], a state-of-the-art zkSNARKs system, identifying a number of issues in its design and implementations. We provide improved versions of their Zero-Knowledge Verifiable Polynomial Delegation (zkVPD) and prove the security of the improved protocols. The improved zkVPD schemes are simpler, faster, and support broader range of finite fields than Virgo, thus more flexible to be used for real-world applications.

Virgo advocated for using fields modulo squared large Mersenne primes due to favorable performance. This inherent field constraint within Virgo has historically evaded comprehensive scrutiny, resulting in several overlooked limitations:

(1) Virgo cannot directly support arithmetic computations over most fields in practice because those fields are not modulo squared Mersenne primes. Even if the goal is to emulate an arithmetic computation defined over the infinite field so a sufficiently large squared Mersenne prime field $\mathbb{F}_{p^2}$ could be used, it is crucial to add necessary range-checks to ensure all input values are actually elements in $\mathbb{F}_p$ rather than $\mathbb{F}_{p^2}$. Otherwise, as we will see in a later example in Section 4.1, the prover can exploit this negligence to easily prove any false statements. Unfortunately, such

range-checks are expensive: the best-known generic method would require breaking $\mathbb{F}_p$ elements into bits and use an extra circuit per input element to assemble the bits back into $\mathbb{F}_p$ element.[1] This can cause a 192× blowup in both witness size and input gates for the 192-bit Mersenne prime p. Unfortunately, Virgo's existing performance evaluation did not account for these costs.

(2) Virgo's protocols design and implementation are tied to fields with *multiplicative* cosets (more on this in Section 2.4), thus facing significant challenges in efficiently supporting computations over binary extension fields, which are extensively used in various cryptographic applications like AES. The support for Boolean circuits within Virgo proves unnecessarily intricate: each XOR gate necessitates emulation via a single multiplication in $\mathbb{F}_{p^2}$, further demanding augmentation of the target circuit with extra range-checks to validate the input witnesses as genuine bit values.

(3) The size of the field crucially impacts the computational security of Virgo proof system. Virgo and its current derivative protocols utilize the 61-bit Mersenne prime, which only provides 90 to 108 bits security guarantee depending on the application circuit size. Moreover, this security assurance remains *conjectural* rather than *provable*. Due to the scarcity of suitable Mersenne primes[2], it was difficult to tune up Virgo's security guarantee without significantly penalizing its performance.

Security Issues. We identified an exploitable vulnerability in Virgo's zkVPD that breaks the soundness of Virgo's proof system. Additionally, we found several implementation flaws in Virgo's C++ source code: contrary to the protocol outlined in their paper, certain costly procedures intended to mask sensitive polynomials were absent. Left unattended, these implementation flaws pose a genuine threat that will undermine the zero knowledge property of Virgo proofs.

Potential Impact. Virgo's zkVPD and its implementation serve as foundational components in numerous subsequent protocols, including Virgo++ [3], zkCNN [4], a one-to-many proof system [5], Orion [6], zkBridge [7], Polaris [8] and Pianist [9], etc. These derivative protocols are susceptible to the identified security vulnerabilities and underestimated costs.

In this paper, we propose rigorously proven techniques to address the aforementioned limitations and security concerns. We will delve into a detailed exposition of these security issues, presenting our comprehensive solutions. Then, we conduct an experimental evaluation to validate the efficacy of our proposed techniques.

1.1 Contributions

An Improved zkVPD. We identified and fixed several security flaws in both the design and implementation of Virgo's zkVPD. We've notified Virgo's authors of our findings. Building upon the Virgo framework, we have introduced improved VPD and zkVPD protocols that are simpler and concretely more efficient, Furthermore, our protocols offer support for not only fields with multiplicative cosets but also those

[1] Although range-changes could also be done using lattice-based proofs [2], the lattice assumptions typically impose extra constraints on the fields, reducing flexibility in supporting the target computation and the rest of the proof system.

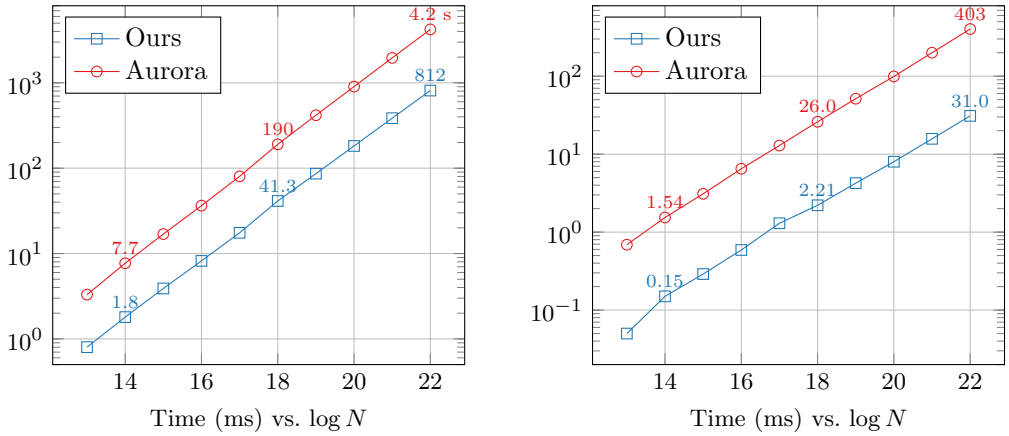
[2] Only five Mersenne primes lie between 2^{32} and 2^{606}. These are $2^{61}-1, 2^{89}-1, 2^{107}-1, 2^{127}-1, 2^{521}-1$.

with additive cosets, hence lending more flexibility in efficiently verifying real-world computations.

Implementation and Experiments. We have implemented our ideas and substantiate our arguments with experiments configured to achieve standard *provable* 128-bit security. The source code of our implementation is openly accessible for anonymous review at https://gitlab.com/zk1252304/zk.

1.2 Related Work

Comparison with Aurora. Aurora [10] also supports fields with additive cosets. However, they used a different VPD variant that does not work with the Goldwasser-Kalai-Rothblum (GKR) [11] technique as efficient as ours. In contrast, we devised several nontrivial improvements (see Section 3.2 for the details): (1) Our zkVPD is done using a single call to a Low-Degree Test (LDT) on a degree-n polynomial, rather than a degree-$2n$ polynomial obtained from a random linear combination of several polynomials as Aurora and Virgo required. (2) We investigated relevant mathematical tools from [12] and [13] to develop a specialized $O(n)$ polynomial division algorithm suited for the GKR context. As a result, the FFT and polynomial division (in point-value representation) used in our protocols are significantly faster than Aurora's [14], as we show with experiments in Figure 1 below.



(a) FFT

(b) Polylnomial Division

Fig. 1: Compare with Aurora ($\mathrm{GF}(2^{192})$).

Comparison with Binius. In a concurrent independent work, Diamond and Posen proposed Binius [15, 16] for efficiently committing and proving multilinear polynomials over binary extension fields. Their approach, based on tensor products and tower of binary extension fields, focuses on committing binary witnesses in a packed form, while allowing to prove polynomial evaluations over the original unpacked witnesses, hence reducing embedding overhead. However, Binius still needs zkVPD as a blackbox, for

which they use either Brakedown, hence resulting in square-root complexity in proof size and verifier time, or FRI [17], which offers poly-logarithmic proof size and verifier time. In this regard, our findings can be integrated with Binius' packing technique to build zero-knowledge protocols that are simpler, more efficient and more flexible to adjust for security requirements.

2 Preliminaries

Notation. We use $\mathcal{P}$, $\mathcal{V}$ to denote prover and verifier, resp., which are modeled as interactive Turing machines. We use $\langle \mathcal{P}, \mathcal{V} \rangle(\lambda, x)$ to denote 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 $x \in L \Leftrightarrow \exists w, R(w, x) = 1$.

We use d to denote the depth of a circuit and denote the degree of a polynomial by d. Given a positive integer n, we define $[n]$ to be the set $\{0, 1, \ldots, n-1\}$.

2.1 Interactive Proofs

Definition 1 (Public-coin IP)**.** *$(\mathcal{P}, \mathcal{V})$ is an n-round* public-coin *interactive proof (IP) for an NP language $L_R = \{x | \exists w, R(x, w) = 1\}$ if it has*

- ***Completeness**: $\forall x,\ x \in L_R \Rightarrow \exists w, (\mathcal{P}(w), \mathcal{V})(\lambda, x) = 1$.*
- ***Soundness**: $\forall x, x \notin L_R \Rightarrow \Pr[(\mathcal{P}', \mathcal{V})(\lambda, x) = 1]$ is negligible in λ for any $\mathcal{P}'$. If $\mathcal{P}'$ is restricted to be probabilistic polynomial-time algorithms, the soundness is said to be* computational*, as opposed to* statistical*.*
- *n **rounds**: After $\mathcal{P}$'s initial message a_0, each time $\mathcal{V}$ sends a uniform random message e_i independent of $\mathcal{P}$'s messages, $\mathcal{P}$ replies with a_i, $\forall i \in \{1, \ldots, n\}$. $\mathcal{V}$ accesses $\{a_i : i \in \{1, \ldots, n\}\}$ through an oracle and outputs one bit to indicate if the proof is accepted.*

Some IPs may have additional properties:

- **Proof of Knowledge** (or **knowledge soundness**)**.** *For every efficient $\mathcal{P}'$, there exists an efficient extractor $\varepsilon^{\mathcal{P}'}$ such that the* knowledge error *is negligible in λ, i.e.,*
$$\Pr[(\mathcal{P}', \mathcal{V})(\lambda, x) = 1 \wedge R(w, x) \neq 1 \mid w \leftarrow \varepsilon^{\mathcal{P}'}(\lambda, x)] = \mathsf{negl}(\lambda).$$
- **Honest-verifier ZK.** *There exists an efficient simulator $\S$ which has oracle access to an honest $\mathcal{V}$, such that $R(w, x) = 1$ implies the transcript $\langle \mathcal{P}(w), \mathcal{V} \rangle(\lambda, x)$ is indistinguishable from $\S^{\mathcal{V}}(x)$.*

Definition 2 (Tree of Transcripts)**.** *A $(k_1, \ldots, k_n)$-**tree of transcripts** for a n-round public-coin interactive oracle protocol, is a set of $\prod_{i=1}^{n} k_i$ transcripts organized in a tree structure such that,*

- *Each edge in the tree represents a challenge sent by $\mathcal{V}$.*
- *Each node at depth i ($1 \leq i \leq n$) in the tree represents a response from $\mathcal{P}$ to answer the i^{th} challenge from $\mathcal{V}$. This node has exactly k_{i+1} children, corresponding to k_{i+1} distinct challenges from $\mathcal{V}$.*
- *Each full transcript corresponds to a path of challenge-response messages from the root to a leaf.*

Definition 3 ($(k_1, \ldots, k_n)$-Special Soundness)**.** *An n-round protocol is said to have $(k_1, \ldots, k_n)$-**special soundness** if there exists an efficient algorithm that, given any $(k_1, \ldots, k_n)$-tree of accepting transcripts, outputs a witness w for $R(w, x) = 1$.*

Theorem 1. *([18, Theorem 1]) Let $n, k_1, \ldots, k_n \in \mathbb{N}$ be such that $K = \prod_{i=1}^{n} k_i$ can be upper bounded by a polynomial. Let $(\mathcal{P}, \mathcal{V})$ be a $(k_1, \ldots, k_n)$-special sound $(2n+1)$-move interactive protocol for relation R, where $\mathcal{V}$ samples the i^{th} challenge uniformly at random from a challenge set of size N_i. Then $(\mathcal{P}, \mathcal{V})$ is knowledge sound with knowledge error*

$$\frac{\prod_{i=1}^{n} N_i - \prod_{i=1}^{n} (N_i - k_i + 1)}{\prod_{i=1}^{n} N_i} \leq \sum_{i=1}^{n} \frac{k_i - 1}{N_i}.$$

2.2 Merkle Tree

Merkle Tree (MT) is a (typically binary) tree data structure in which every leaf node stores a cryptographic hash of a data block, whereas every internal node stores a cryptographic hash of the node's child nodes [19]. Merkle trees are widely used for efficiently assuring data integrity. To demonstrate the integrity of any data block to someone who holds the root of the tree, it suffices to send and verify the pre-images of hashes on the path from the data block up to the root of the tree. In the random oracle model, Merkle trees can be used as cryptographic commitments that allow efficiently de-committing any individual data block. Like Aurora [10] and Virgo [1], our protocol used Merkle trees to commit polynomials, by storing the hash of a polynomial evaluation in each leaf node. Modeling the cryptographic hash as random oracles, Merkle tree offers computational hiding and binding.

In this work, we assume Merkle tree offers the following calls:

$\mathsf{com}_p \leftarrow \mathsf{Com}(p)$*:* Given an input polynomial p, it evaluates p over all points in a domain $\mathbb{D}$ ($|\mathbb{D}| \gg \mathsf{deg}(p)$), whose hashes are stored as leaves of a Merkle tree. The Merkle tree's root is returned as the commitment of p.

$y \leftarrow \mathsf{Decom}(\mathsf{com}_p, x)$*:* Given com_p (a commitment of a polynomial p) and an element x within the range of p's committing domain, it reveals $y = p(x)$ and proves that y is consistent with com_p.

2.3 Reed-Solomon Proof of Proximity

Given only oracle access to a function $f : \mathbb{D} \mapsto \mathbb{F}$, a Reed-Solomon (RS) proximity test [20] is a bounded-error probabilistic polynomial-time algorithm that decides

whether f corresponds to a codeword of $\mathsf{RS}[\mathbb{F}, \mathbb{D}, N, \rho]$, i.e., RS code with code length N and code rate ρ. In other words, it gives good confidence that f is a polynomial of degree no more than ρN. In this work, we used FRI [17], an efficient interactive oracle proof of RS proximity inspired by the Fast Fourier Transform (FFT). Let $\mathbb{D}$ be a coset whose size is a power of 2, FRI reduces the proximity test of $f^{(i)} : \mathbb{D}^{(i)} \mapsto \mathbb{F}$ (where $f^{(0)} = f$, $\mathbb{D}^{(0)} = \mathbb{D}$) to that of $f^{(i+1)} : \mathbb{D}^{(i+1)} \mapsto \mathbb{F}$ where the domain $\mathbb{D}^{(i)}$ shrinks by a factor of $l^{(i)}$ to $\mathbb{D}^{(i+1)}$, like that of FFT. FRI offers a linear prover and a logarithmic verifier, both in $|\mathbb{D}|$. If f is a codeword of $\mathsf{RS}[\mathbb{F}, \mathbb{D}, N, \rho]$, FRI verifier accepts with probability 1. If f is δ-far (by relative Hamming distance) from any codeword of $\mathsf{RS}[\mathbb{F}, \mathbb{D}, N, \rho]$, it is *conjectured* in [17] that the FRI verifier, using a set $\mathbb{X}$ of query points, will accept f with probability at most

$$\frac{2\log^2|\mathbb{D}|}{\epsilon|\mathbb{F}|} + (1 - \delta \cdot (1 - \epsilon))^{|\mathbb{X}|}, \text{ for any } \epsilon > 0.$$

However, as of today, the best *provable* soundness error [21] of an r-round FRI is

$$\frac{(m+1/2)^7|\mathbb{D}|^2}{2\rho^{1.5}|\mathbb{F}|} + \frac{(2m+1)(|\mathbb{D}|+1)}{\sqrt{\rho}} \frac{\sum_{i=0}^{r-1} l^{(i)}}{|\mathbb{F}|} + \left(\sqrt{\rho}\left(1 + \frac{1}{2m}\right)\right)^{|\mathbb{X}|}$$

where m can be any integer no less than 3. So the provable soundness is significantly worse than the conjectured soundness.

Given a polynomial f, FRI's LDT will reveal $\kappa \log(\rho N)$ evaluations of the committed polynomials. However, it is easy to make it zero-knowledge by committing a degree-$(\rho N - 1)$ random polynomial g and proving that $g + f$, instead of f, has degree no more than ρN [1]. Thus, FRI is able to provide the following interfaces:

$\mathsf{com}_f \leftarrow \mathsf{Com}(f)$*:* Given a secret polynomial f, output the commitment of f. Specifically, com_f is simply $\mathsf{MT.Com}(f)$.

$b \leftarrow \mathsf{zkLDT}(f, \mathsf{com}_f, d)$*:* Given a polynomial f, its commitment com_f, and an integer d, outputs 1 if f matches com_f and f's degree is $\leq d$; or 0 otherwise.

2.4 Multiplicative and Additive Cosets

Let $\mathbb{F}$ be a finite field and $\mathbb{G}$ be a cyclic multiplicative subgroup of $\mathbb{F}$ with order a perfect power of 2. We also assume $\mathbb{G}$ can be recursively split into even and odd powers, a structure enabling quasilinear time evaluation of a degree-n polynomial f on all points in $\mathbb{G}$. Specifically, because

$$f(x) = f_{\mathrm{e}}(x^2) + x \cdot f_{\mathrm{o}}(x^2)$$
$$f(-x) = f_{\mathrm{e}}(x^2) - x \cdot f_{\mathrm{o}}(x^2)$$

where f_{e} and f_{o} are of degree $n/2$, made of the even and odd power terms, respectively, the n evaluations of f can be obtained in $O(n \log n)$ time. Let $a \in \mathbb{F}$, the *multiplicative*

coset with respect to a and $\mathbb{G}$ is

$$a\mathbb{G} = \{a \cdot g \mid g \in \mathbb{G}\}$$

In FRI, a polynomial can be committed using a Merkle tree whose leaves are evaluations of the committed polynomial at points in multiplicative cosets.

However, the idea will not work for binary extension fields since they are of characteristic 2, i.e., $x + x = 0$ for all x. For fast evaluation of polynomials, one can use a "lift" function other than squaring to recursively shrink the domain. In particular, let $(\beta_1, \ldots, \beta_{\log n})$ be a basis spanning the n elements vector space $\mathbb{G}_n$ over binary coefficients, two conjugate elements x and $x + \beta_1$ in $\mathbb{G}$ are mapped to the same element $\mathit{lift}(x) = x \cdot (x + \beta_1)$ in the next subgroup, called $\mathbb{G}_{n/2}$. This domain shrinking idea is then recursively applied until the domain size becomes a constant. Consequently, the notion of *additive cosets* with respect to $a \in \mathbb{F}$ and $\mathbb{G}$ is defined as

$$a + \mathbb{G} = \{a + g \mid g \in \mathbb{G}\}$$

To conveniently deal with computations over binary extension fields, not only a variant of FRI suitable for additive cosets should be used, the Virgo protocol itself needs also to be adapted at a higher level, which we will describe in Section 3.2.

2.5 Zero-Knowledge VPD

A zero-Knowledge verifiable polynomial delegation (zkVPD) protocol allows a prover to commit to the secret coefficients of a (possibly multivariate) polynomial so that the polynomial can be later opened at any point chosen by a verifier.

Definition 4. *A zero-knowledge Verifiable Polynomial Delegation (zkVPD) scheme with security parameter n consists of three algorithms* $(\mathsf{Com}, \mathsf{Prove}, \mathsf{Verify})$*:*

$\mathsf{com}_f \leftarrow \mathsf{Com}(1^n, f)$*: Given an input polynomial f, computes a commitment com_f that binds the polynomial f.*

$b \leftarrow (\mathsf{Prove}(f), \mathsf{Verify}(\mathsf{com}_f))\,(1^n, x, y)$*: On input x, y,* Verify *(with input com_f) interacts with* Prove *(with secret input f), and outputs $b = 1$ if and only if $f(x) = y$.*

such that the following properties hold:

(Completeness) For all f, x, and $\mathsf{com}_f \leftarrow \mathsf{Com}(1^n, f)$,

$$\Pr\left[(\mathsf{Prove}(f), \mathsf{Verify}(\mathsf{com}_f))\,(1^n, x, f(x)) = 1\right] = 1.$$

(Soundness) For all f, x, y, and any computationally bounded adversary $\mathcal{A}$, $\mathsf{com} \leftarrow \mathcal{A}(1^n, f)$,

$$y \neq f(x) \Rightarrow \Pr\left[(\mathcal{A}, \mathsf{Verify}(\mathsf{com}))\,(1^n, x, y) = 1\right] = \mathsf{negl}(n)$$

(Knowledge Soundness) For all x, y, and any computationally bounded adversary $\mathcal{A}$, there exists an efficient extractor $\mathcal{E}^{\mathcal{A}}$ such that

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

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

$$\langle \mathsf{Prove}(f), \mathcal{A}(\mathsf{com}_f)\rangle\,(1^n, x, f(x)) \approx S(1^n, x, f(x))$$

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

As a useful building block with important applications, many zkVPD protocols have been proposed and used in zero-knowledge proof systems. These include Bulletproofs [22], Hyrax [23], Libra [24], Ligero++ [3], Virgo [1], Brakedown [25], and Orion [6]. In this paper, we restrict our attention to Virgo, an important transparent succinct zkVPD protocol that have been used in many subsequent ZK protocols. Virgo's zkVPD allows $O(n \log n)$-time prover, $O(\log^2 n)$-time verifier and $O(\log^2 n)$ communication. Although Brakedown and Orion support linear proving time, Orion's offers asymptotically better verification time and proof size than Brakedown. Orion, however, needs to invoke Virgo as a subroutine.

2.6 GKR

Multilinear Extension. Given a function $V : \{0,1\}^\ell \mapsto \mathbb{F}$, its *multilinear extension* $\widetilde{V} : \mathbb{F}^\ell \mapsto \mathbb{F}$ is defined by,

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

It is easy to verify that $\forall x \in \{0,1\}^\ell, \widetilde{V}(x) = V(x)$.

The SumCheck Protocol [26]. The goal of sumcheck protocol 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

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

where $r_1, \ldots, r_{i-1}$ are random values in $\mathbb{F}$ sampled by $\mathcal{V}$ in previous rounds, then $\mathcal{V}$ checks

$$f_{i-1}(r_{i-1}) = f_i(0) + f_i(1) \tag{2}$$

and sends random value $r_i \in \mathbb{F}$.

The GKR Scheme. GKR is a seminal method to verify layered circuit computation [11]. It encodes each layer of circuit computation as a polynomial $V_i(x)$ that maps wire indices to wire values on the i^{th} layer:

$$V_i(x) = \sum_{a,b \in \{0,1\}^{s_{i+1}}} \Big(\mathsf{add}_i(a,b,x)\big(V_{i+1}(a) + \widetilde{V}_{i+1}(b)\big) + \mathsf{mul}_i(a,b,x) V_{i+1}(a) \cdot V_{i+1}(b) \Big), \tag{3}$$

where (a, b) represents hypercube points and $\mathsf{add}_i, \mathsf{mul}_i$ are the predicate functions for addition and multiplication gates. That is, $\mathsf{add}_i(a, b, x) = 1$ if and only if indices a, b, x are the left, right, and output indices of an addition gate on the i^{th} layer, so is $\mathsf{mul}_i(a, b, x)$ similarly defined. Starting from the output layer upwards, GKR invokes one instance of Lund's SumCheck protocol [26] per layer, reducing the validity of $\widetilde{V}_i$ for a previously random point $\boldsymbol{t}_i$ on the hypercube to that of $\widetilde{V}_{i+1}(\boldsymbol{r}_{i+1})$, $\widetilde{V}_{i+1}(\boldsymbol{s}_{i+1})$, which can then be reduced to checking the value of $\widetilde{V}_{i+1}$ at a single point $\boldsymbol{t}_{i+1}$ through random linear combination [27].

On the initial input layer, i.e., the d^{th} layer, to verify the validity of $\widetilde{V}_d(\boldsymbol{r}_d)$, $\widetilde{V}_d(\boldsymbol{s}_d)$, $\mathcal{V}$ picks and sends uniform α_d, β_d and verifies $\alpha_d \widetilde{V}_d(\boldsymbol{r}_d) + \beta_d \widetilde{V}_d(\boldsymbol{s}_d) = \sum_{k \in [2^{s_d}]} c_k w_k$ where w_k's are (potentially secret) inputs to the circuit, and c_k's are public constants efficiently computable from $\boldsymbol{r}_d, \boldsymbol{s}_d, \alpha_d, \beta_d$:

$$c_k = \alpha_d \prod_{i \in [s_d]} \chi(k_i, \boldsymbol{r}_d[i]) + \beta_d \prod_{i \in [s_d]} \chi(k_i, \boldsymbol{s}_d[i])$$

where k_i is the i^{th} bit of k; $\chi(0, x) = 1 - x$ and $\chi(1, x) = x$; $\boldsymbol{r}_d[i], \boldsymbol{s}_d[i]$ are the i^{th} part of $\boldsymbol{r}_d, \boldsymbol{s}_d$, resp. This final check can be accomplished by a variant of verifiable polynomial delegation protocol that we will describe in Section 3.

- $\mathsf{com}_{\boldsymbol{w}} \leftarrow \mathsf{Com}(\boldsymbol{w})$: Generate a commitment of the vector $\boldsymbol{w}$.
- $b \leftarrow \mathsf{Prove}(\boldsymbol{w}, \mathsf{com}_{\boldsymbol{w}}, \boldsymbol{r}, \boldsymbol{s}, u, v, \alpha, \beta)$: Let k_i be the i^{th} bit of k and $\boldsymbol{w} = \{w_k\}_{k \in [n]}$. $\mathcal{P}$, holding all inputs to Prove, interacts with $\mathcal{V}$, holding all but $\boldsymbol{w}$. $\mathcal{V}$ outputs a bit $b = 1$ if and only if

$$\alpha u + \beta v = \sum_k c_k w_k \tag{4}$$

$$\text{where} \quad c_k = \alpha \Pi_i \chi(k_i, \boldsymbol{r}[i]) + \beta \Pi_i \chi(k_i, \boldsymbol{s}[i]) \tag{5}$$
$$\chi(0, x) = 1 - x, \quad \chi(1, x) = x$$

• $\mathsf{com}_{\boldsymbol{w}} \leftarrow \mathsf{Com}(\boldsymbol{w})$

(1) Let $l : \mathbb{F} \mapsto \mathbb{F}$ be the degree-$(n-1)$ polynomial satisfying $l(h_i) = w_i$ for all i where $h_i \in \mathbb{H} \subset \mathbb{F}$, with $\mathbb{H}$ a multiplicative coset (of size n) suitable for FRI.
(2) $\mathcal{P}$ computes and outputs $\mathsf{com}_l = \mathsf{MT.Com}(l)$.

• $b \leftarrow \mathsf{Prove}(\mathsf{com}_{\boldsymbol{w}}, \boldsymbol{w}, \boldsymbol{r}, \boldsymbol{s}, u, v, \alpha, \beta)$

(1) Let $q : \mathbb{F} \mapsto \mathbb{F}$ be the degree-$(n-1)$ polynomial with $q(h_i) = c_i$ for all i where $h_i \in \mathbb{H}$. $\mathcal{P}$ divides $l(x)q(x)$ by $Z_{\mathbb{H}}(x) = \Pi_{a \in \mathbb{H}}(x - a)$ to obtain the unique degree-$(n-2)$ polynomials f, g, and the unique constant γ such that

$$l(x) \cdot q(x) = \gamma + x \cdot f(x) + g(x) \cdot Z_{\mathbb{H}}(x) \tag{6}$$

(2) $\mathcal{P}$ sends $\mathsf{com}_f := \mathsf{MT.Com}(f)$ and $\mathsf{com}_g := \mathsf{MT.Com}(g)$.
(3) $\mathcal{P}$ and $\mathcal{V}$ call $\mathsf{FRI.LDT}(f, \mathsf{com}_f, n-2)$ to ensure $\mathsf{deg}(f) \leq n-2$. Let $\mathbb{X}$ be a set of κ non-conjugate random check-points used in the first recursion of FRI.LDT.
(4) For every $x_i \in \mathbb{X}$, $\mathcal{P}$ sends $q(x_i)$. They call MT.Decom to reveal $l(x_i), f(x_i), g(x_i)$ and verify that $\forall x_i \in \mathbb{X}$,

$$x_i \cdot f(x_i) = l(x_i) \cdot q(x_i) - (\alpha u + \beta v) \cdot n^{-1} - g(x_i) \cdot Z_{\mathbb{H}}(x_i). \tag{7}$$

(5) $\mathcal{P}$ and $\mathcal{V}$ call the non-ZK GKR to ensure that all $q(x_i)$'s used in the check above are honestly computed from $\alpha, \beta, \boldsymbol{r}, \boldsymbol{s}, x_i$ according to Equation (5), the definition and the interpolation of q.

Fig. 2: Improved VPD protocol for multiplicative cosets

3 zkVPD

In Virgo's context, a custom zkVPD is defined to accomplish the final check of the GKR protocol. The specific task of Virgo's zkVPD consists of a plaintext evaluation of n coefficients (Equation (5)) and a dot-product between this vector of coefficients and the vector of secret witnesses (Equation (4)).

3.1 For Fields with Multiplicative Cosets

Virgo's zkVPD follows Aurora's polynomial commitment scheme for multiplicative cosets. Their scheme mainly targets at computations over finite fields of large prime modulus or their extension fields with a sufficiently large subset suitable for FRI. To support computations on the binary field, binary additions (XORs) are emulated by multiplications in large prime characteristic fields. Upon closer scrutiny, we identified a number of issues in the design and implementation of their zkVPD.

• $\mathsf{com}_{\boldsymbol{w}} \leftarrow \mathsf{Com}(\boldsymbol{w})$

(1) Let $l : \mathbb{F} \mapsto \mathbb{F}$ be the degree-$(n-1)$ polynomial satisfying $l(h_i) = w_i$ for all i where $h_i \in \mathbb{H} \subset \mathbb{F}$, with $\mathbb{H}$ an multiplicative coset (of size n) suitable for FRI.
(2) $\mathcal{P}$ masks $l(x)$ by sampling a uniform polynomial $l' : \mathbb{F} \mapsto \mathbb{F}$ of degree $\kappa - 1$. Let $\hat{l} = l + Z_{\mathbb{H}} \cdot l'$. $\mathcal{P}$ computes and outputs $\mathsf{com}_{\boldsymbol{w}} \coloneqq \mathsf{MT.Com}(\hat{l})$.

• $b \leftarrow \mathsf{Prove}(\mathsf{com}_{\boldsymbol{w}}, \boldsymbol{w}, \boldsymbol{r}, \boldsymbol{s}, u, v, \alpha, \beta)$

(1) Let $q : \mathbb{F} \mapsto \mathbb{F}$ be the degree-$(n-1)$ polynomial with $q(h_i) = c_i$ for all i where $h_i \in \mathbb{H}$. $\mathcal{P}$ divides $\hat{l}(x)q(x)$ by $Z_{\mathbb{H}}(x) = \Pi_{a \in \mathbb{H}}(x - a)$ to obtain the unique degree-$(n-2)$ polynomial $f(x)$, the unique degree-$(n+\kappa-2)$ polynomial $g(x)$, and the unique constant γ such that

$$\hat{l}(x) \cdot q(x) = \gamma + x \cdot f(x) + g(x) Z_{\mathbb{H}}(x) \tag{8}$$

(2) $\mathcal{P}$ sends $\mathsf{com}_f \coloneqq \mathsf{MT.Com}(f)$ and $\mathsf{com}_g \coloneqq \mathsf{MT.Com}(g)$.
(3) $\mathcal{P}$ and $\mathcal{V}$ call $\mathsf{FRI.zkLDT}(f, \mathsf{com}_f, n-2)$ to ensure $\mathsf{deg}(f) \leq n-2$. Let $\mathbb{X}$ be a set of κ non-conjugate random check-points used by the first recursion of FRI.zkLDT. We require $\mathbb{X} \cap \mathbb{H} = \emptyset$.
(4) For every $x_i \in \mathbb{X}$, $\mathcal{P}$ sends $q(x_i)$. They use MT.Decom to reveal $\hat{l}(x_i), f(x_i), g(x_i)$ and verify that for all i,

$$x_i f(x_i) = \hat{l}(x_i) \cdot q(x_i) - (\alpha u + \beta v) \cdot n^{-1} - g(x_i) Z_{\mathbb{H}}(x_i). \tag{9}$$

(5) $\mathcal{P}$ and $\mathcal{V}$ call the non-ZK GKR to ensure that all $q(x_i)$'s used in the check above are honestly computed from $\alpha, \beta, \boldsymbol{r}, \boldsymbol{s}, x_i$ according to Equation (5), the definition and the interpolation of q.

Fig. 3: Improved zkVPD protocol for multiplicative cosets

A Soundness Issue in Virgo's VPD. Given a degree-n univariate polynomial f and a multiplicative coset $\mathbb{H}$, to prove that $\sum_{x \in \mathbb{H}} f(x) = \mu$, they defined

$$p(x) = \frac{|\mathbb{H}| \cdot f(x) - |\mathbb{H}| \cdot Z_{\mathbb{H}}(x) \cdot h(x) - \mu}{|\mathbb{H}| \cdot x}$$

where h is the quotient polynomial obtained from dividing f by $Z_{\mathbb{H}}(x) = \prod_{a \in \mathbb{H}} (x - a)$ [1, Page 864, Formula (5)]. Leveraging Lemma 4, the goal of proving $\sum_{x \in \mathbb{H}} f(x) = \mu$ can be reduced to proving polynomials f, h, p are of degree less than $n, n - |\mathbb{H}|$, and $|\mathbb{H}| - 1$, respectively. However, when invoking the Low Degree Test (LDT), they suggested combining the three individual LDT proofs into one with the maximum degree bound n by "multiplying h and p with appropriate monomials" [1, page 864, left column, last paragraph, second sentence], i.e., $x^{|\mathbb{H}|}$ and $x^{n-|\mathbb{H}|+1}$, respectively.

We note that this introduces an easily exploitable vulnerability since the term x in the denominator of p can be canceled with the "x" in the monomial, hence cannot guarantee the divisibility of p's numerator by its denominator, directly compromising the soundness of their VPD protocol.

ZK Issues in Virgo's Implementation. their implementation did not match with their paper specification: (1) Their implementation did not generate and commit the degree-$(2n-1+\kappa)$ masking polynomial s and apply it to hide polynomial $l' \cdot q$, as described by their Protocol 3, step *3)* and *5)*; (2) Their implementation did not combine polynomials $l' \cdot q, h, p$ into a degree-$(2n-1+\kappa)$ polynomial and feed it to LDT to ensure $l' \cdot q, h, p$ are all within their respective degree thresholds, as specified in step *6)* of their Protocol 3. Instead, their LDT was only run on a degree-$(n-2)$ polynomial. It is not hard to fix these security issues, but according to our experiments, the costs of their zkVPD prover will increase by more than 90% due to the inverse FFT and other missing polynomial operations.

Improved VPD and zkVPD. Our improved VPD and zkVPD protocols for fields with appropriate multiplicative cosets are described in Figure 2 and Figure 3, respectively. Similar to Virgo, the high-level idea is to convert the validity of polynomial evaluation to low degree tests and equality check of a polynomial equation at κ random points. However, our key observation is that the equational constraint to check can be written without using divisions and it suffices to invoke LDT on a single polynomial, f, to establish the soundness of the secret polynomial evaluation. Moreover, the polynomial on which we invoke LDT is of degree $\leq (n-2)$, whereas Virgo tries to bound the degrees of three polynomials $l' \cdot q, h, p$ whose degrees are up to $(2n+\kappa-1)$, hence more than $2\times$ slower.

The changes to upgrade our VPD to a zkVPD protocol is highlighted in red in Figure 3: (1) l is masked by a degree-$(\kappa-1)$ polynomial l' so that the κ revealed evaluations of l in Step (4) leak no information about the witness. (2) The degree-$(2n+\kappa-2)$ polynomial $\hat{l} \cdot q$ (instead of $l \cdot q$) is divided by $Z_{\mathbb{H}}$ to yield f, g since it is the masked polynomial $\hat{l}$ that was committed. For the same reason, the κ points equality check is also on $\hat{l}$ and freshly defined f, g. (3) In Step (3), zkLDT, instead of LDT, is used with $\mathbb{X} \cap \mathbb{H} = \emptyset$, so that all the polynomial evaluations revealed during LDT leak no information about the witness. We stress that even if the LDT was not carried directly on $l(x)$, $l(x)$ is fully fixed by $f(x)$, whose degree is upper bounded by the LDT, because $l(x)q(x) = \gamma + xf(x)$ for all $x \in \mathbb{H}$. Therefore, once $f(x)$ is extracted, evaluations of $l(x)$ on $\mathbb{H}$ (i.e., the original witnesses) can be fully recovered.

This optimization allows us to simplify Virgo's protocols, e.g., reducing their 10-step zkVPD to our 5-step zkVPD. The run-time cost due to combining polynomials $l' \cdot q, h, p$ and low degree testing of a higher degree polynomial can be cut down substantially. Since our LDT is only applied to the single polynomial f which is not defined using polynomial divisibility, the ground of Virgo's security vulnerability (described earlier in this section)—due to multiplying a monomial with a fractional item whose denominator could be canceled out—is eliminated altogether.

Next, we state the security of Figure 3 protocol as Theorem 2 and prove it based on Lemma 4.

Theorem 2. *The protocol of Figure 3 is a zkVPD.*

Proof **Completeness.** For an honest prover who knows $\boldsymbol{w}$ such that Equation (4) holds, $\mathsf{deg}(f)$ by definition must be $\leq n-2$, so do $\mathsf{deg}(\hat{f})$ (since $\mathsf{deg}(p') = \kappa \ll n$). Therefore, the FRI low-degree test in Step (3) will pass.

Since $V = \sum_k c_k w_k = \sum_{a\in\mathbb{H}} l(a)q(a) = \sum_{a\in\mathbb{H}} \hat{l}(a)q(a)$, along with Equation (8) and Lemma 4, we have

$$\begin{aligned} V &= \sum_{a\in\mathbb{H}} \hat{l}(a)q(a) = \sum_{a\in\mathbb{H}} \gamma + \sum_{a\in\mathbb{H}} a\cdot f(a) + \sum_{a\in\mathbb{H}} g(a)Z_{\mathbb{H}}(a) \\ &= \gamma|\mathbb{H}| + 0\cdot f(0)\cdot|\mathbb{H}| + \sum_{a\in\mathbb{H}} g(a)\cdot 0 = \gamma|\mathbb{H}| = \gamma n \end{aligned}$$

Therefore,

$$\begin{aligned} x\cdot f(x) &= \hat{l}(x)q(x) - \gamma - g(x)Z_{\mathbb{H}}(x) \\ &= \hat{l}(x)q(x) - V\cdot n^{-1} - g(x)Z_{\mathbb{H}}(x) \end{aligned}$$

Hence, we know that Equation (9) must hold. So the checks in Step (4) will pass. Following the definition of q and Equation (5), it is easy to see that the GKR proof of Step (5) will pass. This completes the proof of completeness.

Soundness. We define the following events:

Eq: $x\cdot f(x) = \hat{l}(x)\cdot q(x) - V\cdot n^{-1} - g(x)\cdot Z_{\mathbb{H}}(x)$.

LD: $\mathsf{deg}(f) \leq n-2$.

LdT: $\mathsf{FRI.zkLDT}(\mathsf{com}_f, f, n-2)$ passes.

EqT: The equality check in Step (4) passes on all κ points.

Pass: $\mathsf{LdT}\wedge\mathsf{EqT}$.

Bad: $V \neq \sum_k c_k w_k$.

Awry: $\neg\mathsf{LD}\vee\neg\mathsf{Eq}$.

Next, we will prove $\mathsf{Bad}\Rightarrow\neg\mathsf{LD}\vee\neg\mathsf{Eq}$, that is, $\mathsf{Bad}\Rightarrow\mathsf{Awry}$. This is because

$$\begin{aligned} &\neg(\neg\mathsf{LD}\vee\neg\mathsf{Eq}) \Leftrightarrow \mathsf{LD}\wedge\mathsf{Eq} \\ \Leftrightarrow\ &\mathsf{LD}\wedge x\cdot f(x) = l(x)\cdot q(x) - V\cdot n^{-1} - g(x)\cdot Z_{\mathbb{H}}(x) \\ \Rightarrow\ &\mathsf{LD}\wedge \sum_{x\in\mathbb{H}} x\cdot f(x) = \sum_{x\in\mathbb{H}} \Big(l(x)\cdot q(x) - V\cdot n^{-1} - g(x)\cdot Z_{\mathbb{H}}(x)\Big) \\ \Rightarrow\ &0 = \sum_{x\in\mathbb{H}} l(x)\cdot q(x) - V - 0 \\ \Rightarrow\ &V = \sum_{x\in\mathbb{H}} l(x)\cdot q(x) \Leftrightarrow \neg\mathsf{Bad} \end{aligned}$$

Note the second implication uses Lemma 4 on $x\cdot f(x)$ and the fact that $Z_{\mathbb{H}}(x) = 0$ for all $x\in\mathbb{H}$.

Second, let $\mathbb{D}$ be the subset of points where the polynomials $\hat{f}, \hat{l}, \hat{g}$ were committed. Because LD implies Equation (7) has at most $n-2$ roots, if the equation does not hold, then the κ equality checks in Step (4) will pass with probability at most $\left(\frac{n-2}{|\mathbb{D}|}\right)^{\kappa}$. That is, $\Pr[\mathsf{LdT}\mid\mathsf{LD}\wedge\neg\mathsf{Eq}] \leq \left(\frac{n-2}{|\mathbb{D}|}\right)^{\kappa}$, which is negligible in κ as $n \ll |\mathbb{D}|$.

Now that $\mathsf{Bad} \Rightarrow \mathsf{Awry}$ and the completeness implies $\neg\mathsf{Bad} \Rightarrow \mathsf{Pass}$, we can apply Lemma 1 to infer

$$\begin{aligned}\Pr[\mathsf{Pass}|\mathsf{Bad}] &\leq \max(\Pr[\mathsf{LdT}|\neg\mathsf{LD}], \Pr[\mathsf{EqT}|\mathsf{LD} \wedge \neg\mathsf{Eq}]) \\ &= \max\left(\mathcal{E}_{\mathsf{FRI}}, \left(\frac{n-2}{|\mathbb{D}|}\right)^{\kappa}\right)\end{aligned}$$

where $\mathcal{E}_{\mathsf{FRI}}$ is the negligible soundness error of the FRI protocol. This completes the proof of soundness.

Knowledge Soundness. Ignoring the negligible knowledge error of FRI and GKR (i.e., assuming FRI and GKR realize ideal proofs of knowledge, which were proven in their respective works in the literature), our zkVPD is $\left(\binom{n-2}{\kappa}+1\right)$-special sound (Definition 3). This is because, due to the pigeonhole principle, a $\left(\binom{n-2}{\kappa}+1\right)$ transcript tree must reveal $n-1$ evaluations of the secret polynomials f, unless a collision is found on the hashes. This allows to fully recover f. Because $\forall x \in \mathbb{H}, l(x) \cdot q(x) = \gamma + x \cdot f(x)$ where $q(x), \gamma$ are constants known to $\mathcal{V}$, so all the witnesses encoded in l can be recovered through extracting corresponding points on f. Applying Theorem 1 with $\nu = 1, k_1 = \binom{n-2}{\kappa}+1, N_1 = \binom{n/\rho}{\kappa}$, we know that the knowledge error of our zkVPD is upper-bounded because

$$\frac{k_1 - 1}{N_1} = \frac{\binom{n-2}{\kappa}}{\binom{n/\rho}{\kappa}} < \frac{\left(\frac{n-2}{\kappa}\right)^{\kappa}}{\left(\frac{en/\rho}{\kappa}\right)^{\kappa}} = \left(\frac{n-2}{en/\rho}\right)^{\kappa} < \left(\frac{\rho}{e}\right)^{\kappa}$$

where ρ is the code rate, e.g., $\rho = 1/16$. The first inequality is a result of applying the bounds of binomial coefficient, $\left(\frac{n}{k}\right)^k \leq \binom{n}{k} < \left(\frac{en}{k}\right)^k$. This completes the proof of knowledge soundness.

Zero-Knowledge. Our zkVPD is zero-knowledge because all messages sent by $\mathcal{P}$ in every step are uniform random:

Step (1), Step (2) The property of random oracle guarantees the messages (Merkle tree root hashes) $\mathcal{P}$ sent are uniform random.

Step (3), Step (4) $\mathcal{P}$ reveals κ points of polynomials $\hat{l}, f, g$, which are masked by a uniform random secret polynomial of degree $\kappa - 1$, thus the revealed κ points are uniform random, independent of the witness.

Step (5) Messages in this step only involves public data.

Therefore, the transcript of our zkVPD can be generated by an efficient simulator without knowing the secret witness. This completes the proof of zero-knowledge.

□

Lemma 1. *If* $\mathsf{Bad} \Rightarrow \mathsf{Awry}$ *and* $\neg\mathsf{Bad} \Rightarrow \mathsf{Pass}$*, then*

$$\Pr[\mathsf{Pass}|\mathsf{Bad}] \leq \max(\Pr[\mathsf{LdT}|\neg\mathsf{LD}], \Pr[\mathsf{EqT}|\mathsf{LD} \wedge \neg\mathsf{Eq}]).$$

Proof

$$\begin{aligned}&\Pr[\mathsf{Pass} \mid \mathsf{Bad}] = \Pr[\mathsf{Pass} \wedge \mathsf{Bad}] / \Pr[\mathsf{Bad}] \\ \leq &\frac{\Pr[\mathsf{Pass} \wedge \mathsf{Bad}] + \Pr[\neg\mathsf{Bad} \wedge \mathsf{Awry}]}{\Pr[\mathsf{Bad}] + \Pr[\neg\mathsf{Bad} \wedge \mathsf{Awry}]}\end{aligned}$$

$$
\begin{aligned}
&= \frac{\Pr[\mathsf{Pass} \wedge \mathsf{Bad}] + \Pr[\neg\mathsf{Bad} \wedge \mathsf{Awry}]}{\Pr[\mathsf{Bad} \wedge \mathsf{Awry}] + \Pr[\neg\mathsf{Bad} \wedge \mathsf{Awry}]} && (10)\\
&= \frac{\Pr[\mathsf{Pass} \wedge \mathsf{Bad}] + \Pr[\neg\mathsf{Bad} \wedge \mathsf{Awry}]}{\Pr[\mathsf{Awry}]}\\
&= \frac{\Pr[(\mathsf{Pass} \wedge \mathsf{Bad}) \vee (\neg\mathsf{Bad} \wedge \mathsf{Awry})]}{\Pr[\mathsf{Awry}]}\\
&= \frac{\Pr[(\mathsf{Pass} \wedge \mathsf{Bad} \wedge \mathsf{Awry}) \vee (\neg\mathsf{Bad} \wedge \mathsf{Awry})]}{\Pr[\mathsf{Awry}]} && (11)\\
&= \frac{\Pr[(\mathsf{Pass} \wedge \mathsf{Bad} \vee \neg\mathsf{Bad}) \wedge \mathsf{Awry}]}{\Pr[\mathsf{Awry}]}\\
&= \frac{\Pr[((\mathsf{Pass} \wedge \mathsf{Bad}) \vee (\mathsf{Pass} \wedge \neg\mathsf{Bad})) \wedge \mathsf{Awry}]}{\Pr[\mathsf{Awry}]} && (12)\\
&= \frac{\Pr[(\mathsf{Pass} \wedge (\mathsf{Bad} \vee \neg\mathsf{Bad})) \wedge \mathsf{Awry}]}{\Pr[\mathsf{Awry}]}\\
&= \frac{\Pr[\mathsf{Pass} \wedge \mathsf{Awry}]}{\Pr[\mathsf{Awry}]}\\
&= \Pr[\mathsf{Pass} \mid \mathsf{Awry}]\\
&= \Pr[\mathsf{Pass} \mid (\neg\mathsf{LD} \vee \neg\mathsf{Eq})]\\
&= \Pr[\mathsf{Pass} \mid (\neg\mathsf{LD} \vee (\mathsf{LD} \wedge \neg\mathsf{Eq}))]\\
&\leq \max\left(\Pr[\mathsf{Pass} \mid \neg\mathsf{LD}], \Pr[\mathsf{Pass} \mid (\mathsf{LD} \wedge \neg\mathsf{Eq})]\right)\\
&= \max\left(\Pr[\mathsf{LdT} \wedge \mathsf{EqT} \mid \neg\mathsf{LD}], \Pr[\mathsf{LdT} \wedge \mathsf{EqT} \mid (\mathsf{LD} \wedge \neg\mathsf{Eq}]\right)\\
&\leq \max\left(\Pr[\mathsf{LdT} \mid \neg\mathsf{LD}], \Pr[\mathsf{EqT} \mid \mathsf{LD} \wedge \neg\mathsf{Eq}]\right)\\
&\leq \max\left(\Pr[\mathsf{LdT} \mid \neg\mathsf{LD}], \Pr[\mathsf{EqT} \mid \mathsf{LD} \wedge \neg\mathsf{Eq}]\right)
\end{aligned}
$$

Note that Equation (10) and Equation (11) hold because $\mathsf{Bad} \Rightarrow \mathsf{Awry}$, hence $\mathsf{Bad} = \mathsf{Bad} \wedge \mathsf{Awry}$; Equation (12) holds because $\neg\mathsf{Bad} \Rightarrow \mathsf{Pass}$, hence $\neg\mathsf{Bad} = \mathsf{Pass} \wedge \neg\mathsf{Bad}$. □

Lemma 2. $\forall a, b, c, d > 0$,

$$a/b > c/d \implies a/b > (a+c)/(b+d).$$

Proof $a/b > c/d \implies ad > bc \implies ab + ad > ab + bc$
$\implies (b+d) \cdot a > (a+c) \cdot b \implies a/b > (a+c)/(b+d)$. □

Lemma 3. $\forall a, b, c, d > 0$,

$$a/b < c/d,\ a > c,\ b > d \implies a/b > (a-c)/(b-d).$$

Proof $a/b < c/d \implies ad > bc \implies ab - ad > ab - bc \implies (b-d)a > (a-c)b \implies a/b > (a-c)/(b-d)$. □

• $\mathsf{com}_{\boldsymbol{w}} \leftarrow \mathsf{Com}(\boldsymbol{w})$

(1) Let $l : \mathbb{F} \mapsto \mathbb{F}$ be the degree-$(n-1)$ polynomial satisfying $l(h_i) = w_i$ for all i where $h_i \in \mathbb{H} \subset \mathbb{F}$, with $\mathbb{H}$ an additive coset (of size n) suitable for FRI.
(2) $\mathcal{P}$ computes and outputs $\mathsf{com}_l = \mathsf{MT.Com}(l)$.

• $b \leftarrow \mathsf{Prove}(\mathsf{com}_{\boldsymbol{w}}, \boldsymbol{w}, \boldsymbol{r}, \boldsymbol{s}, u, v, \alpha, \beta)$

(1) Let $q : \mathbb{F} \mapsto \mathbb{F}$ be the degree-$(n-1)$ polynomial with $q(h_i) = c_i$ for all i where $h_i \in \mathbb{H}$. $\mathcal{P}$ divides $l(x)q(x)$ by $Z_{\mathbb{H}}(x) = \Pi_{a\in\mathbb{H}}(x-a)$ to obtain the unique degree-$(n-2)$ polynomials f, g, and the unique constant γ such that

$$l(x) \cdot q(x) = f(x) + \gamma \cdot x^{n-1} + g(x) \cdot Z_{\mathbb{H}}(x). \tag{14}$$

(2) $\mathcal{P}$ sends $\mathsf{com}_f := \mathsf{MT.Com}(f)$ and $\mathsf{com}_g := \mathsf{MT.Com}(g)$.
(3) $\mathcal{P}$ and $\mathcal{V}$ call $\mathsf{FRI.LDT}(f, \mathsf{com}_f, n-2)$ to ensure $\mathsf{deg}(f) \leq n-2$. Let $\mathbb{X}$ be a set of κ non-conjugate random check-points used in the first recursion of FRI.LDT.
(4) $\forall x_i \in \mathbb{X}$, $\mathcal{P}$ sends $q(x_i)$ and call MT.Decom to reveal $l(x_i), f(x_i), g(x_i)$. With $\xi = \sum_{a\in\mathbb{H}} a^{n-1}$, $\mathcal{V}$ verifies $\forall x_i \in \mathbb{X}$,

$$\xi f(x_i) = \xi l(x_i) q(x_i) - (\alpha u + \beta v) \cdot x_i^{n-1} - \xi g(x_i) Z_{\mathbb{H}}(x_i). \tag{15}$$

(5) $\mathcal{P}$ and $\mathcal{V}$ call the non-ZK GKR to ensure that all $q(x_i)$'s are honestly computed from $\alpha, \beta, \boldsymbol{r}, \boldsymbol{s}, x_i$ according to Equation (5), the definition and the interpolation of q.

Fig. 4: Improved VPD protocol for additive cosets

Lemma 4 ([28]). *Let $\mathbb{H}$ be a multiplicative coset in $\mathbb{F}$, and $f : \mathbb{F} \mapsto \mathbb{F}$ be a polynomial of degree less than $|\mathbb{H}|$.*

$$\sum_{a\in\mathbb{H}} f(a) = f(0) \cdot |\mathbb{H}|. \tag{13}$$

3.2 For Fields with Additive Cosets

Many computations, such as block ciphers and binary codes, can be more efficiently represented by circuits over extension fields of the binary field, which was not directly supported by Virgo. To better support computations over binary extension fields, we extended Virgo's zkVPD to work with fields over additive cosets. Our VPD and zkVPD protocols are given in Figure 4 and Figure 5, respectively. The very high level idea of these protocols resemble those of the multiplicative coset case. However, the protocols for additive cosets, which use a very different set of algebraic rules, are distinctive in many details.

• $\mathsf{com}_{\boldsymbol{w}} \leftarrow \mathsf{Com}(\boldsymbol{w})$

(1) Let $l : \mathbb{F} \mapsto \mathbb{F}$ be the degree-$(n-1)$ polynomial satisfying $l(h_i) = w_i$ for all i where $h_i \in \mathbb{H} \subset \mathbb{F}$, with $\mathbb{H}$ an additive coset (of size n) suitable for FRI.
(2) $\mathcal{P}$ masks $l(x)$ by sampling a uniform polynomial $l' : \mathbb{F} \mapsto \mathbb{F}$ of degree $\kappa - 1$. Let $\hat{l} = l + Z_{\mathbb{H}} \cdot l'$. $\mathcal{P}$ computes and outputs $\mathsf{com}_{\boldsymbol{w}} \coloneqq \mathsf{MT.Com}(\hat{l})$.

• $b \leftarrow \mathsf{Prove}(\mathsf{com}_{\boldsymbol{w}}, \boldsymbol{w}, \boldsymbol{r}, \boldsymbol{s}, u, v, \alpha, \beta)$

(1) Let $q : \mathbb{F} \mapsto \mathbb{F}$ be the degree-$(n-1)$ polynomial with $q(h_i) = c_i$ for all i where $h_i \in \mathbb{H}$. $\mathcal{P}$ divides $\hat{l}(x)q(x)$ by $Z_{\mathbb{H}}(x) = \Pi_{a \in \mathbb{H}}(x - a)$ to obtain the unique degree-$(n-2)$ polynomials f, the unique degree-$(n+\kappa-2)$ polynomial g, and the unique constant γ such that

$$\hat{l}(x) \cdot q(x) = f(x) + \gamma \cdot x^{n-1} + g(x) \cdot Z_{\mathbb{H}}(x). \tag{16}$$

(2) $\mathcal{P}$ sends $\mathsf{com}_f \coloneqq \mathsf{MT.Com}(f)$ and $\mathsf{com}_g \coloneqq \mathsf{MT.Com}(g)$.
(3) $\mathcal{P}$ and $\mathcal{V}$ call $\mathsf{FRI.zkLDT}(f, \mathsf{com}_f, n-2)$ to ensure $\mathsf{deg}(f) \leq n-2$. Let $\mathbb{X}$ be a set of κ non-conjugate random check-points opened in the first recursion of FRI.zkLDT. We require $\mathbb{X} \cap \mathbb{H} = \emptyset$.
(4) $\forall x_i \in \mathbb{X}$, $\mathcal{P}$ sends $q(x_i)$ and calls MT.Decom to reveal $\hat{l}(x_i), f(x_i), g(x_i)$. With $\xi = \sum_{a \in \mathbb{H}} a^{n-1}$, $\mathcal{V}$ verifies $\forall x_i \in \mathbb{X}$,

$$\xi f(x_i) = \xi \hat{l}(x_i) q(x_i) - (\alpha u + \beta v) \cdot x_i^{n-1} - \xi \cdot g(x_i) Z_{\mathbb{H}}(x_i). \tag{17}$$

(5) $\mathcal{P}$ and $\mathcal{V}$ call the non-ZK GKR to ensure that all $q(x_i)$'s are honestly computed from $\alpha, \beta, \boldsymbol{r}, \boldsymbol{s}, x_i$ according to Equation (5), the definition and the interpolation of q.

Fig. 5: Improved zkVPD protocol for additive cosets

First, the support for additive cosets leverages Lemma 5, which is very different from Lemma 4.

Lemma 5 ([28]). *Let $\mathbb{H}$ be an affine subspace (i.e., additive coset) in $\mathbb{F}$, and $f : \mathbb{F} \mapsto \mathbb{F}$ be a polynomial of degree less than $|\mathbb{H}| - 1$.*

$$\sum_{a \in \mathbb{H}} f(a) = 0.$$

To support additive cosets, we defined the polynomial f differently, focusing on the $(n-2)$ lower-degree terms of the remainder (cf. Equation (14) vs. Equation (6), and Equation (16) vs. Equation (8)). Also, the Equation (17) to be verified involves a domain $\mathbb{H}$-dependent constant ξ, whereas Equation (9) for multiplicative cosets does

not involve any constant like this. Second, the FFT and FRI algorithms require a completely different definition of $\mathbb{H}$ than the one used in the multiplicative case. For instance, efficiently computing f, g from $\hat{l}$ requires the *novel basis* and some additional optimization, which we will elaborate next.

Polynomial division over novel basis. To compute the polynomial division as required in Equation (14), one can first apply FFT to convert the point representations of polynomials l, q and $Z_{\mathbb{H}}$ into their coefficient representations, then compute the polynomial division. However, in order to support fields over additive cosets, if those steps are carried out over the monomial basis, it is both concretely and asymptotically slow: FFT takes $O(n \log n \log\log n)$ time and polynomial division takes $O(n \log n)$ time [12]. In our implementation, we adapted the idea from [13] to compute the FFT and polynomial division steps over the novel basis, which is concretely much more efficient while allowing $O(n \log n)$-time FFT and $O(n)$-time special division by $Z_{\mathbb{H}}(x)$.

Next, we explain how [13]'s $O(n \log n)$ division algorithm can be specialized for our VPD scenario to reduce the cost of division to $O(n)$ time. Let $\mathbb{H}$ be an additive coset of 2^m elements, which can be viewed as a linear space spanned by a basis $\{\beta_i\}_{i\in[m]}$. Let $\mathbb{H}_k$ be the linear space spanned by $\{\beta_i\}_{i\in[k]}$. The vanishing polynomial $z_i(x)$ over $\mathbb{H}_i$ is defined as $z_i(x) = \prod_{a\in\mathbb{H}_i}(x-a)$. The *novel* polynomial basis $\{X_i\}_{i\in[2^k]}$ is defined as

$$X_i(x) = \frac{\prod_{j=0}^{k-1} z_j^{i_j}(x)}{\prod_{j=0}^{k-1} z_j^{i_j}(\beta_j)}$$

where $i_j \in \{0,1\}$ represents the j^{th} bit of i.

Let the degree-$(2n-2)$ dividend polynomial $l(x)\cdot q(x)$ has coefficient representation $l(x)\cdot q(x) = \sum_{j=0}^{2n-3} a_j X_j(x)$. Note our degree-$n$ divisor polynomial $Z_{\mathbb{H}}(x)$ in its coefficient representation over the novel basis is exactly $z_\ell(x)$ where $\ell = \log n$. Since

$$\begin{aligned}
l(x)\cdot q(x) &= \sum_{j=0}^{2^\ell-1} a_j X_j(x) + \sum_{j=2^\ell}^{2n-3} a_j X_j(x) \\
&= \sum_{j=0}^{2^\ell-1} a_j X_j(x) + \frac{z_\ell(x)}{z_\ell(\beta_\ell)} \sum_{j=2^\ell}^{2n-3} a_j X_{j-2^\ell}(x) \\
&= \sum_{j=0}^{2^\ell-1} a_j X_j(x) + z_\ell(x) \left(\sum_{j=0}^{2n-3-2^\ell} \left(a_{j+2^\ell} \cdot p_\ell^{-1} \right) X_j(x) \right)
\end{aligned}$$

where $p_\ell = z_\ell(\beta_\ell)$. That is, the first 2^ℓ coefficients are precisely the coefficients of the remainder polynomial, while the coefficients of the quotient polynomial can be easily computed using $2n-3-2^\ell+1 = n-2$ field multiplications.

The security of our zkVPD for additive cosets is stated as Theorem 3 and proved based on Lemma 5 as follows.

Theorem 3. *The protocol of Figure 5 is a zkVPD.*

Proof **Completeness.** In an honest execution of the protocol when Equation (4) holds, $\mathsf{deg}(f)$ by definition must be $\leq n-2$. Hence, the $\mathsf{FRI.LDT}$ call in Step (3) will pass.

Since $V = \sum_k c_k w_k = \sum_{a\in\mathbb{H}} l(a)q(a)$, along with Equation (14) and Lemma 5, we have

$$\begin{aligned} V &= \sum_{a\in\mathbb{H}} l(a)q(a) = \sum_{a\in\mathbb{H}} \hat{l}(a)q(a) \\ &= \sum_{a\in\mathbb{H}} f(a) + \sum_{a\in\mathbb{H}} \gamma a^{n-1} + \sum_{a\in\mathbb{H}} g(a)Z_{\mathbb{H}}(a) = \sum_{a\in\mathbb{H}} a^{n-1}\cdot\gamma = \xi\gamma \end{aligned}$$

Therefore,

$$\begin{aligned} \xi f(x) &= \xi\cdot l(x)\cdot q(x) - \xi\gamma\cdot x^{n-1} - \xi\cdot g(x)\cdot Z_{\mathbb{H}}(x) \\ &= \xi\cdot l(x)\cdot q(x) - V\cdot x^{n-1} - \xi\cdot g(x)\cdot Z_{\mathbb{H}}(x) \end{aligned}$$

Hence Equation (17) must hold. So the checks in Step (4) will pass.

Following the definition of q and Equation (5), it is easy to see that the GKR proof of Step (5) will pass. This completes the proof of completeness.

Soundness. We begin with defining the following events:

Eq: $\xi\cdot f(x) = \xi\cdot l(x)q(x) - V\cdot x^{n-1} - \xi\cdot g(x)Z_{\mathbb{H}}(x)$.

LD: $\mathsf{deg}(p) \leq n-2$.

LdT: $\mathsf{FRI.LDT}(\mathsf{com}_f, f, n-2)$ passes.

EqT: The equality check in Step (4) passes on all κ points.

Pass: $\mathsf{LdT}\wedge\mathsf{EqT}$.

Bad: $V \neq \sum_k c_k w_k$.

Awry: $\neg\mathsf{LD}\vee\neg\mathsf{Eq}$.

Next, we will prove $\mathsf{Bad}\Rightarrow\neg\mathsf{LD}\vee\neg\mathsf{Eq}$, that is, $\mathsf{Bad}\Rightarrow\mathsf{Awry}$. Note

$$\begin{aligned} &\neg(\neg\mathsf{LD}\vee\neg\mathsf{Eq}) \Leftrightarrow \mathsf{LD}\wedge\mathsf{Eq} \\ \Leftrightarrow\ &\mathsf{LD}\wedge\xi f(x) = \xi l(x)q(x) - Vx^{n-1} - \xi g(x)Z_{\mathbb{H}}(x) \\ \Rightarrow\ &\mathsf{LD}\wedge\sum_{x\in\mathbb{H}}\xi f(x) = \sum_{x\in\mathbb{H}}\xi l(x)q(x) - Vx^{n-1} - \xi g(x)Z_{\mathbb{H}}(x) \\ \Rightarrow\ &0 = \xi\cdot\sum_{x\in\mathbb{H}} l(x)\cdot q(x) - V\cdot\xi - 0 \\ \Rightarrow\ &V = \sum_{x\in\mathbb{H}} l(x)\cdot q(x) \Leftrightarrow \neg\mathsf{Bad} \end{aligned}$$

The last implication depends on the fact that $\xi\neq 0$, which can be derived as a corollary of [28, Theorem 1] (by setting their $V=\mathbb{H}$, $\alpha=0$ and $p^m=n$, so their $S_h(V;\alpha)$ is exactly our ξ).

Second, let $\mathbb{D}$ be the subset of points where the polynomials $\hat{l}, f, g$ were committed. Because LD implies Equation (17) has at most $n-2$ roots, should Equation (17) not hold, the κ equality checks in Step (4) will pass with probability at most $\left(\frac{n-2}{|\mathbb{D}|}\right)^{\kappa}$. That is, $\Pr[\mathsf{LdT}\mid\mathsf{LD}\wedge\neg\mathsf{Eq}] \leq \left(\frac{n-2}{|\mathbb{D}|}\right)^{\kappa}$, which is negligible in κ as $n\ll|\mathbb{D}|$.

Now that $\mathsf{Bad}\Rightarrow\mathsf{Awry}$ and the completeness implies $\neg\mathsf{Bad}\Rightarrow\mathsf{Pass}$, we can apply Lemma 1 to infer

$$\Pr[\mathsf{Pass}|\mathsf{Bad}] \leq \max(\Pr[\mathsf{LdT}|\neg\mathsf{LD}], \Pr[\mathsf{EqT}|\mathsf{LD}\wedge\neg\mathsf{Eq}])$$

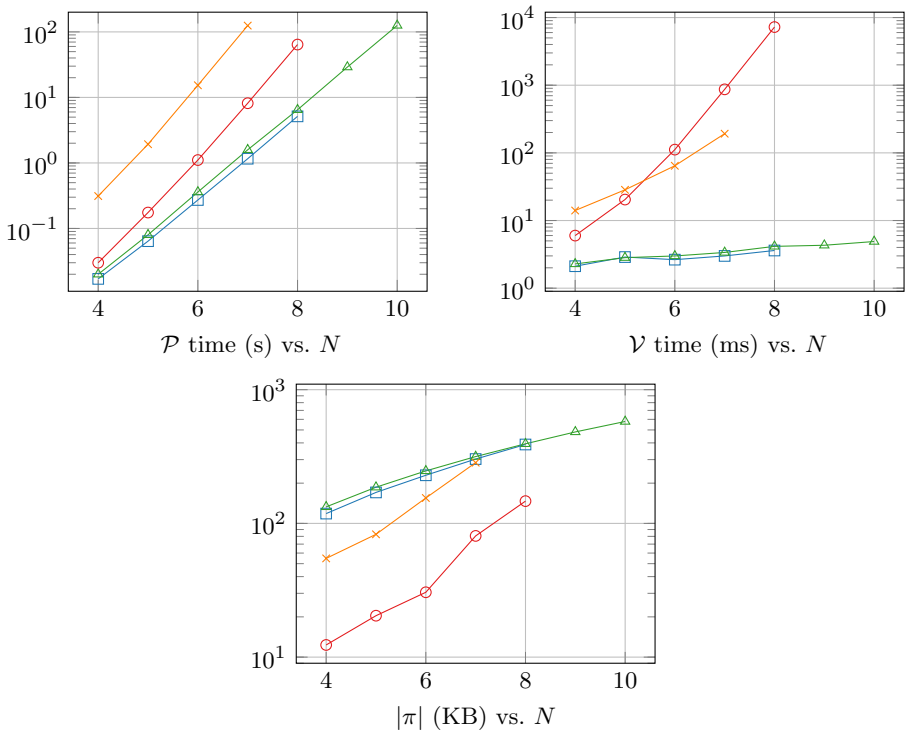


Fig. 6: Emulate $2^N \times 2^N$ Infinite Field Matrix Multiplication using $\mathbb{F}_p$.
Ours (192-bit p), Ours (256-bit p), SpartanNIZK, SpartanSNARK

$$= \max\left(\mathcal{E}_{\mathsf{FRI}}, \left(\frac{n-2}{|\mathbb{D}|}\right)^{\kappa}\right)$$

where $\mathcal{E}_{\mathsf{FRI}}$ is the negligible soundness error of the FRI protocol. This completes the proof of soundness.

The proofs for *knowledge soundness* and *zero-knowledge* are very similar to those for multiplicative cosets.

□

4 Experiments

We run experiments on a Linux laptop (Intel i9-11900H CPU, 64GB DDR4). We set protocol parameters to achieve 128-bit provable computational security, except those for comparing with Orion whose implementation can't be easily tuned up to align with ours.

4.1 Matrix Multiplication

Issues in Virgo's Matrix Multiplication. Matrix multiplication has been a popular benchmark to evaluate the performance of ZKP systems. However, Virgo's

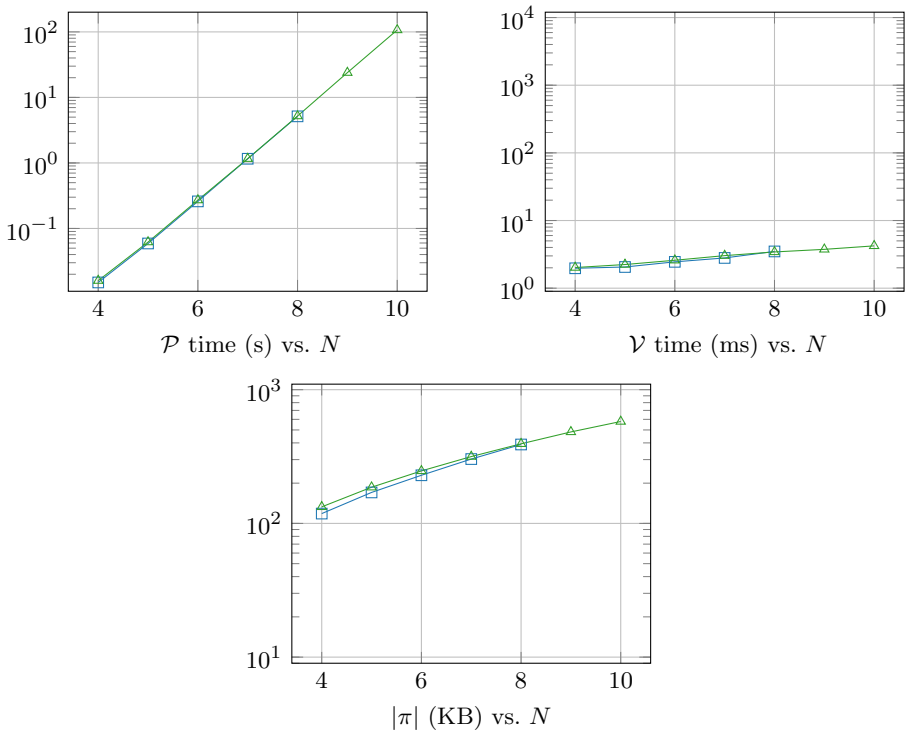


Fig. 7: $2^N \times 2^N$ Matrix Multiplication on Binary Extension Fields.
$\mathbb{F} = GF(2^{192})$, $\mathbb{F} = GF(2^{256})$

approach to matrix multiplication, and to arithmetic fields computations in general, allows security attacks.

Recall that Virgo relies on squared Mersenne prime fields for better performance. Unfortunately, it has been largely ignored that the intended computations to prove are often defined on simple prime fields or the infinite integer field (which can be emulated by a sufficiently large prime field), but not the few squared Mersenne prime fields Virgo supports. To prove computations over mod-p prime fields, Virgo needs to be instrumented with expensive range checks to ensure the witnesses are indeed integers of Z_p rather than Z_{p^2}. For instance, Virgo used the prime $p = 2^{61} - 1$ to emulate arithmetic operations in the infinite integer ring. Borrowing the notation of complex numbers, without proper range checking, a malicious prover can prove, e.g., $10^3 \times 10^3 = 1,000,001$, simply by inserting a 1 and -1 into the imaginary parts of the multiplicand and the multiplier, making it $(10^3 + i) \times (10^3 - i) = 1,000,001$. For large prime field like GF$(2^{61} - 1)$, range-checking all the secret inputs will incur significant performance overhead.

Suggested Fixes and Evaluation. To resolve the issue without expensive range checking, we suggest using large prime fields that still support FRI. For example, we used GF(p), with $p = 2^{192} - 15 \cdot 2^{56} + 1$ being a 192-bit prime, to guarantee 128-bit

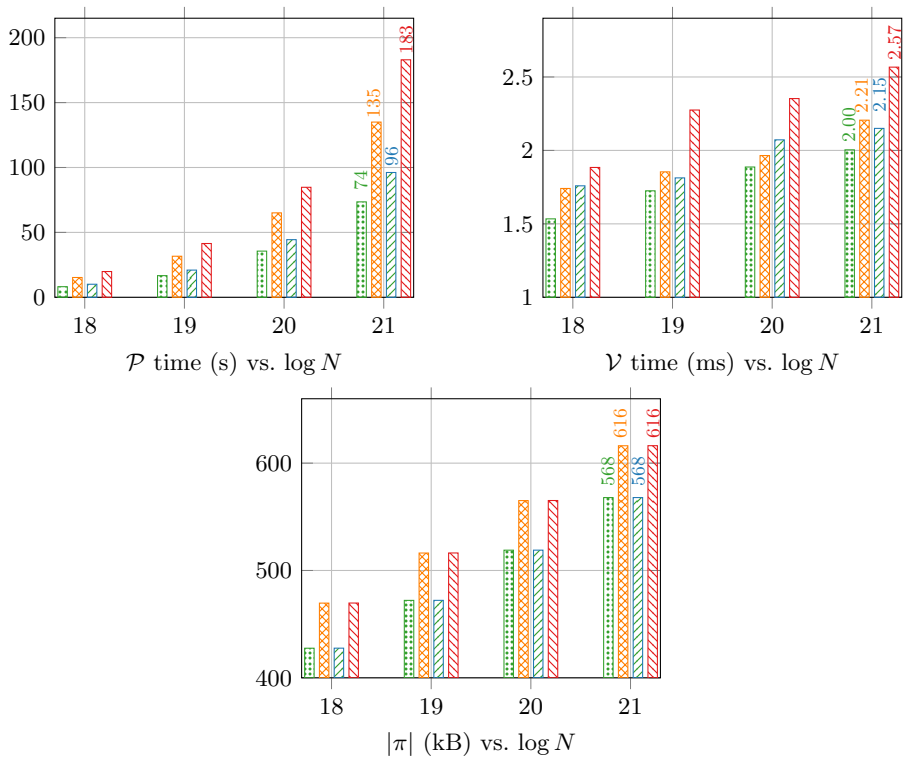


Fig. 8: Compare VPD/zkVPD on GF $\left((2^{127}-1)^2\right)$.
VPD (Ours), VPD (Virgo), zkVPD (Ours), zkVPD (Virgo)

computational security for multiplying matrices of size up to 512×512, and use the 256-bit prime $p = 2^{256} - 19 \cdot 2^{50} + 1$ for matrices of size up to $2^{24}\times2^{24}$. Using our improved zkVPD, along with MPC-in-the-Head technique to handle the linear constraints out of the SumCheck, the resulting protocol turns out very efficient, using only several milliseconds verifier time and several hundreds KB proofs for matrices up to 1024×1024. Figure 6 shows our protocol offers concretely more efficient prover and verifier than Spartan, which uses the Ristretto255 prime-order ECC group. Our protocol is quantum-resistant and has better memory efficiency (as Spartan run out of memory when $N > 8$ on the same hardware). Our proof size is larger but is expected to break even at $N \geq 7$ for SpartanSNARK and at $N \geq 10$ for SpartanNIZK. In Figure 7, we also plotted the costs of matrix multiplication over GF(2^{192}) and GF(2^{256}), which can indicate the costs of proving some useful computations such as binary coding/decoding and some lattice-based cryptography. We did not include Virgo in both figures because it requires substantial changes in Virgo's implementation to support such fields.

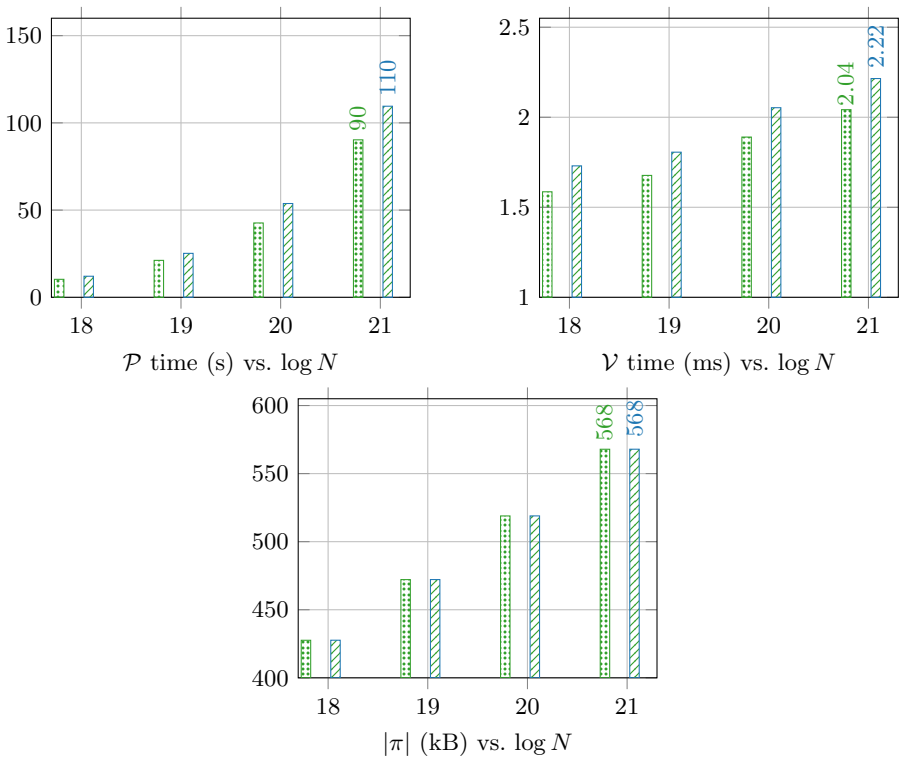


Fig. 9: Our VPD/zkVPD on GF(2^{256}) with $\kappa = 75$. VPD, zkVPD

4.2 zkVPD

Figure 8 depicts how our VPD and zkVPD protocols perform in comparison with those of Virgo's. All protocols are run on fields sufficiently large with the same code rate $\rho = 1/16$ and $\kappa = 67$ to achieve 128-bit provable security. For 2^{21} secret inputs, our improved VPD prover and zkVPD prover run 82% and 91% faster, our verifier is 11% and 20% faster, and proof size is 8% smaller than Virgo's respective peers. Figure 9 benchmarks our zkVPD on binary extension fields, which are not supported by Virgo.

Figure 10 compares our zkVPD with Orion. Experiments of [6, Figure 3 & Figure 4] already show that Orion performs preferably as compared to Aurora, Ligero, and the field-agnostic scheme Brakedown [25]. We note that Orion's implementation invokes the original Virgo as a subroutine to handle part of their computations, thus is *vulnerable to the attacks we described earlier*, whereas ours is *not*. We only compare with Orion on the squared 61-bit Mersenne prime field that achieves ≈100-bit conjectured computational security because that is the only field implemented in Orion. We find Orion's prover 4.2–16.8 times faster whereas our verifier is 33.6–95.0 times faster, and our proof size is 27–38.2 times smaller. We note that Orion did not implement sublinear verification, thus their verification time actually grew linearly with the length of the witness.

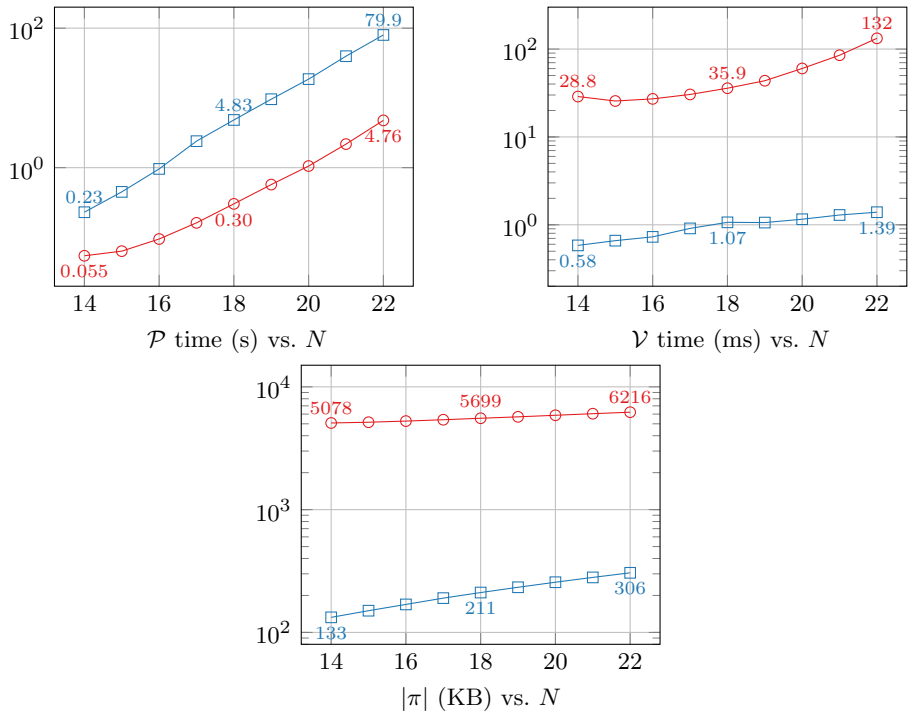


Fig. 10: Compare our zkVPD with Orion on $\mathrm{GF}((2^{61}-1)^2)$.
Ours, Orion

5 Conclusion

The utility of Virgo is somewhat restricted due to its reliance on specific cryptographic fields. Moreover, deficiencies in the design and implementation of Virgo's zkVPD protocol introduce exploitable vulnerabilities that can compromise Virgo's soundness and zero-knowledge guarantees. We have rectified the security flaws and enhanced its VPD protocol to accommodate a broader range of computations. Our improvements also enable more effective support for standard provable (as opposed to conjectural) security claims in practice.

6 List of abbreviations

ZK zero-knowledge
zkSNARK zero-knowledge succinct non-interactive argument of knowledge
VPD verifiable polynomial delegation
zkVPD zero-knowledge verifiable polynomial delegation
FFT fast Fourier transform
LDT low degree test

7 Declarations

Availability of data and materials

The implementation source code is available at https://gitlab.com/zk1252304/zk.

Funding

We received no funding to conduct this research work.

Acknowledgements

We thank the authors of Virgo for email correspondence, answering our questions and confirming the issues mentioned in the paper.

References

[1] Zhang, J., Xie, T., Zhang, Y., Song, D.: Transparent polynomial delegation and its applications to zero knowledge proof. In: 2020 IEEE Symposium on Security and Privacy, pp. 859–876 (2020). https://doi.org/10.1109/SP40000.2020.00052

[2] Lyubashevsky, V., Nguyen, N.K., Plançon, M.: Lattice-based zero-knowledge proofs and applications: Shorter, simpler, and more general. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part II. LNCS, vol. 13508, pp. 71–101 (2022). https://doi.org/10.1007/978-3-031-15979-4_3

[3] Bhadauria, R., Fang, Z., Hazay, C., Venkitasubramaniam, M., Xie, T., Zhang, Y.: Ligero++: A new optimized sublinear IOP. In: Ligatti, J., Ou, X., Katz, J., Vigna, G. (eds.) ACM CCS 2020, pp. 2025–2038 (2020). https://doi.org/10.1145/3372297.3417893

[4] Liu, T., Xie, X., Zhang, Y.: zkCNN: Zero knowledge proofs for convolutional neural network predictions and accuracy. In: Vigna, G., Shi, E. (eds.) ACM CCS 2021, pp. 2968–2985 (2021). https://doi.org/10.1145/3460120.3485379

[5] Zhang, J., Xie, T., Hoang, T., Shi, E., Zhang, Y.: Polynomial commitment with a one-to-many prover and applications. In: Butler, K.R.B., Thomas, K. (eds.) USENIX Security 2022, pp. 2965–2982 (2022)

[6] Xie, T., Zhang, Y., Song, D.: Orion: Zero knowledge proof with linear prover time. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part IV. LNCS, vol. 13510, pp. 299–328 (2022). https://doi.org/10.1007/978-3-031-15985-5_11

[7] Xie, T., Zhang, J., Cheng, Z., Zhang, F., Zhang, Y., Jia, Y., Boneh, D., Song, D.: zkBridge: Trustless cross-chain bridges made practical. In: Yin, H., Stavrou, A., Cremers, C., Shi, E. (eds.) ACM CCS 2022, pp. 3003–3017 (2022). https://doi.org/10.1145/3548606.3560652

[8] Fu, S., Gong, G.: Polaris: Transparent succinct zero-knowledge arguments for R1CS with efficient verifier. PoPETs **2022**(1), 544–564 (2022) https://doi.org/10.2478/popets-2022-0027

[9] Liu, T., Xie, T., Zhang, J., Song, D., Zhang, Y.: Pianist: Scalable zkrollups via fully distributed zero-knowledge proofs. In: 2024 IEEE Symposium on Security and Privacy, pp. 1777–1793 (2024). https://doi.org/10.1109/SP54263.2024.00035

[10] Ben-Sasson, E., Chiesa, A., Riabzev, M., Spooner, N., Virza, M., Ward, N.P.: Aurora: Transparent succinct arguments for R1CS. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part I. LNCS, vol. 11476, pp. 103–128 (2019). https://doi.org/10.1007/978-3-030-17653-2_4

[11] Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: Delegating computation: interactive proofs for muggles. In: Ladner, R.E., Dwork, C. (eds.) 40th ACM STOC, pp. 113–122 (2008). https://doi.org/10.1145/1374376.1374396

[12] Gao, S., Mateer, T.: Additive fast fourier transforms over finite fields. IEEE Transactions on Information Theory (2010)

[13] Lin, S.-J., Al-Naffouri, T.Y., Han, Y.S.: FFT algorithm for binary extension finite fields and its application to reed–solomon codes. IEEE Transactions on Information Theory (2016)

[14] Chiesa, A., Wu, A., Ovsiankin, M., Hu, Y., Ward, N., etc., A.R.: Aurora: C++ library for IOP-based zkSNARKs. Accessed in August, 2024. https://github.com/scipr-lab/libiop

[15] Diamond, B.E., Posen, J.: Succinct Arguments over Towers of Binary Fields. Cryptology ePrint Archive, Report 2023/1784 (2023). https://eprint.iacr.org/2023/1784

[16] Diamond, B.E., Posen, J.: Polylogarithmic Proofs for Multilinears over Binary Towers. Cryptology ePrint Archive, Report 2024/504 (2024). https://eprint.iacr.org/2024/504

[17] Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Fast reed-solomon interactive oracle proofs of proximity. In: Chatzigiannakis, I., Kaklamanis, C., Marx, D., Sannella, D. (eds.) ICALP 2018. LIPIcs, vol. 107, pp. 14–11417 (2018). https://doi.org/10.4230/LIPIcs.ICALP.2018.14

[18] Attema, T., Cramer, R., Kohl, L.: A compressed Σ-protocol theory for lattices. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part II. LNCS, vol. 12826, pp. 549–579. Virtual Event (2021). https://doi.org/10.1007/978-3-030-84245-1_19

[19] Merkle, R.C.: A digital signature based on a conventional encryption function. In: Pomerance, C. (ed.) CRYPTO'87. LNCS, vol. 293, pp. 369–378 (1988). https:

//doi.org/10.1007/3-540-48184-2_32

[20] Rubinfeld, R., Sudan, M.: Self-testing polynomial functions efficiently and over rational domains. In: Frederickson, G.N. (ed.) 3rd SODA, pp. 23–32 (1992)

[21] Ben-Sasson, E., Carmon, D., Ishai, Y., Kopparty, S., Saraf, S.: Proximity gaps for reed-solomon codes. In: 61st FOCS, pp. 900–909 (2020). https://doi.org/10.1109/FOCS46700.2020.00088

[22] Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: Short proofs for confidential transactions and more. In: 2018 IEEE Symposium on Security and Privacy, pp. 315–334 (2018). https://doi.org/10.1109/SP.2018.00020

[23] Wahby, R.S., Tzialla, I., shelat, a., Thaler, J., Walfish, M.: Doubly-efficient zkSNARKs without trusted setup. In: 2018 IEEE Symposium on Security and Privacy, pp. 926–943 (2018). https://doi.org/10.1109/SP.2018.00060

[24] Xie, T., Zhang, J., Zhang, Y., Papamanthou, C., Song, D.: Libra: Succinct zero-knowledge proofs with optimal prover computation. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part III. LNCS, vol. 11694, pp. 733–764 (2019). https://doi.org/10.1007/978-3-030-26954-8_24

[25] Golovnev, A., Lee, J., Setty, S.T.V., Thaler, J., Wahby, R.S.: Brakedown: Linear-time and field-agnostic SNARKs for R1CS. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part II. LNCS, vol. 14082, pp. 193–226 (2023). https://doi.org/10.1007/978-3-031-38545-2_7

[26] Lund, C., Fortnow, L., Karloff, H.J., Nisan, N.: Algebraic methods for interactive proof systems. In: 31st FOCS, pp. 2–10 (1990). https://doi.org/10.1109/FSCS.1990.89518

[27] Chiesa, A., Forbes, M.A., Spooner, N.: A Zero Knowledge Sumcheck and its Applications. Cryptology ePrint Archive, Report 2017/305 (2017). https://eprint.iacr.org/2017/305

[28] Byott, N.P., Chapman, R.J.: Power sums over finite subspaces of a field. Finite Fields and Their Applications (1999)