Efficient Functional Commitments:
How to Commit to a Private Function

Dan Boneh, Wilson Nguyen, and Alex Ozdemir

Stanford University,
{dabo, wdnguyen, aozdemir}@cs.stanford.edu

Abstract. We construct efficient (function hiding) functional commitments for arithmetic circuits of polynomial size. A (function hiding) functional commitment scheme enables a *committer* to commit to a secret function f and later prove that $y = f(x)$ for public x and y—without revealing any other information about f. As such, functional commitments allow the operator of a secret process to prove that the process is being applied uniformly to everyone. For example, one can commit to the secret function that computes credit scores and then prove that it is applied uniformly to all. To build a functional commitment scheme, we introduce a new primitive called a *proof of function relation* (PFR) to show that a committed relation is a function. We show that combining a suitable preprocessing zk-SNARK (or more precisely, an Algebraic Holographic Proof) with a PFR yields a secure functional commitment scheme. We then construct efficient PFRs for two popular preprocessing zk-SNARKs, and obtain two functional commitment schemes for arithmetic circuits. Along the way we develop new techniques for proving interesting properties of committed polynomials, which may be of independent interest.

1 Introduction

We introduce a new cryptographic primitive called a (*function-hiding*) *functional commitment*: a *committer* commits to a secret function $f : \mathcal{X} \to \mathcal{Y}$ using a succinct hiding and binding commitment scheme. Later, the committer can reliably open this function at any public point in the domain of f without revealing anything else about f. Specifically, for a public pair $(x \in \mathcal{X}, y \in \mathcal{Y})$, the committer can prove to a verifier that the committed function f satisfies $y = f(x)$, without revealing anything else about f.

In more detail, a (function hiding) functional commitment scheme is a triple $(\mathsf{Setup}, \mathsf{Commit}, \mathsf{Eval})$. $\mathsf{Setup}(1^\lambda, N)$ is a randomized algorithm that outputs public parameters pp; these parameters support commitments to functions of complexity at most N. $\mathsf{Commit}(\mathsf{pp}, f, r)$ is a deterministic algorithm that takes as input the description of a function $f \in \mathcal{F}$, where $\mathcal{F}$ is a function space, and randomness r, and outputs a hiding and binding commitment c. Eval is a protocol between a prover $\mathcal{P}_{\mathsf{E}}(\mathsf{pp}, f, r, x, y)$ and a verifier $\mathcal{V}_{\mathsf{E}}(\mathsf{pp}, c, x, y)$ that is designed to convince the verifier that $f(x) = y$. Informally, the evaluation protocol Eval

should be (i) complete, (ii) zero knowledge, and (iii) an argument of knowledge for a function $f \in \mathcal{F}$. We define these properties formally in Section 3.

Several existing cryptographic primitives can be viewed as special cases of functional commitments. In a polynomial commitment scheme (PCS) [28], the committer commits to a polynomial of bounded degree in $\mathbb{F}[X]$, and can later open the polynomial at any public point in $\mathbb{F}$. A verifiable random function (VRF) [34] is a functional commitment where the committer commits to a pseudorandom function (PRF) instantiated using a particular random key. Later, the PRF can be reliably opened at any point in its domain. Other examples include vector commitments [10, 13, 14, 23, 36], accumulators [8, 10, 12] and zero knowledge sets [15, 16, 33]. A vector commitment can be viewed as a function-hiding functional commitment where the function is described as a truth table.

In this paper, we generalize these examples and study the general question of how to commit to a secret function. We construct two efficient (function hiding) functional commitment schemes for the set of all functions that can be expressed as an arithmetic circuit of some bounded size.

An important property of a functional commitment, implied by its security properties listed above, is called *evaluation binding* which means that a malicious prover cannot convince the verifier that $f(x) = y$ and $f(x) = y'$ for some $y \neq y'$. More precisely, it should be infeasible to find c, x, y, y', where $y \neq y'$, such that the verifier accepts the inputs $\mathcal{V}_{\mathsf{E}}(\mathsf{pp}, c, x, y)$ and $\mathcal{V}_{\mathsf{E}}(\mathsf{pp}, c, x, y')$. Evaluation binding ensures that c is a commitment to a function: there is a unique output for every input.

Example applications. In the United States, a credit bureau is an organization the computes a person's credit score based on their financial record. To ensure that the credit bureau uses the same function for the entire population, the credit bureau can publish a commitment to its secret function. Then, given a person's financial record, say Bob, the credit bureau can compute Bob's credit score, and prove to Bob that the score was computed correctly with respect to the committed function. Here Bob plays the role of the verifier. Evaluation binding ensures that Bob's credit score is determined uniquely by his financial record; the credit bureau has no leeway in choosing Bob's score. Furthermore, the function can be audited by an auditor who is trusted to examine its inner workings. If the auditor is satisfied that the function is "fair," it can indicate that by signing the public commitment to the function.

Other applications of functional commitments may include:

- *Bail decisions*: recent proposals suggest using an algorithm to decide whether a defendant is granted bail [27]. The algorithm used might need to remain secret to prevent exploitation. With a functional commitment scheme, the courts can show that the same algorithm is being used for all defendants.
- *Software-as-a-Service*: Consider a cloud service that charges per query to an image classifier A. Suppose B is a classifier that is less accurate than A, but cheaper to evaluate. The cloud could save money by using B and lie that it

is using A. By committing to the classifier A, the cloud can prove that it is providing the same service to all of its customers.

– *Price discrimination*: a company could commit to a pricing function that takes product and market data as input and outputs a price. Then, it could prove to every customer that the price is strictly a function of these inputs—and *not* user-specific data.

More generally, functional commitments are relevant to the area of *algorithmic fairness* [3, 29]. A functional commitment ensures that a process is applied uniformly to everyone. However, while uniform application is necessary for fairness, it is not sufficient. In fact, since "fairness" is a social construct, formalizing it is the subject of a great deal of work [3, 29]. Ultimately, one would like to prove that the same function is applied to everyone *and* that the function satisfies an agreed upon fairness criterion. The latter condition can be verified by a trusted auditor or by some other cryptographic technique (as discussed in Section 6), while the former is ensured by a functional commitment.

1.1 Related work

Previous works on functional commitments [30, 31, 36] consider a dual notion, which we call ***input-hiding*** *functional commitments*: the committer commits to an input x, and later the committer proves that $f(x) = y$, for some public function f and a value y. In the current paper we focus on ***function-hiding*** *functional commitments*, where the committer commits to a function f and later proves that $f(y) = x$ for some public pair (x, y). These two notions can be shown to be equivalent using a universal function evaluator $\mathcal{U}(f, x)$, where $\mathcal{U}(f, x) = f(x)$. However, in practice they are quite different due to efficiency considerations. An efficient input-hiding functional commitment scheme can be constructed directly from a standard succinct commitment scheme and a zk-SNARK, as observed in [30]. In contrast, constructing an efficient function-hiding functional commitment scheme, as we do here, requires additional tools to efficiently prove that the committed function is well formed. In particular, one has to prove that the commitment really is to a *function*: every input has exactly one output. We do so using a new tool we call a Proof of Function Relation (PFR), as discussed in Section 1.2.

Prior to the present work, the term *functional commitment* meant strictly *input-hiding* functional commitments. They were implicitly constructed by Gorbunov, Vaikuntanathan, and Wichs [24], although commitments produced by their commitment scheme are not succinct. The term *functional commitment* was introduced by Libert, Ramanna and Yung [30] and further developed in [31, 36]. The focus of these works is on efficient *input-hiding* functional commitments under falsifiable assumptions (the simple zk-SNARK based construction requires a non falsifiable assumption). Libert et al. [30] and Lipmaa and Pavlyk [31] give an input-hiding functional commitment scheme for the family of linear (or linearizable) functions. Peikert, Pepin, and Sharp [36] give a lattice construction for low depth boolean circuits.

Function Hiding Functional commitments from circuit garbling. Consider a family of circuits that have the same wiring (i.e., the same circuit topology), but differ in the choice of gate for each location. Then one can use Yao garbled circuits [39] to construct a functional commitment scheme for this family, where a commitment supports a single evaluation. The reusable garbled circuits scheme of Goldwasser, Kalai, Popa, Vaikuntanathan, and Zeldovich [22] extends this to multiple evaluations. In fact, [22] achieves a stronger property, where the function is hidden from the verifier, and the input x is hidden from the committer. However, the evaluation protocol is not succinct or fast to verify, and requires fairly heavy cryptographic primitives such as fully hom. encryption.

1.2 Technical Overview

Constructing a functional commitment scheme: the challenge. Let $\mathcal{F}$ be the set of functions that can be computed by a bounded size arithmetic circuit over a field $\mathbb{F}$. A strawman functional commitment scheme for $\mathcal{F}$ can be built from a standard succinct commitment scheme and a general zero knowledge proof system, using universal circuits. Let $\langle f \rangle$ be en explicit description of some function $f \in \mathcal{F}$ and let $c \leftarrow \mathsf{Commit}(\langle f \rangle, r)$. Let $\mathcal{U}(\cdot, \cdot)$ be a universal function evaluator. The prover must convince the verifier that it knows a witness for the following relation:

$$R := \big\{(c, x, y \; ; \; \langle f \rangle, r) \; : \; c = \mathsf{Commit}(\langle f \rangle, r) \;\text{ and }\; \mathcal{U}(\langle f \rangle, x) = y \;\text{ and }\; \langle f \rangle \in \mathcal{F}\big\}$$

However, we aim to construct a functional commitment scheme where the evaluation proof is non-interactive, succinct, and fast to verify.

A natural starting point is a preprocessing zk-SNARK such as Marlin [17], Plonk [20], or many others [1, 18, 25, 26, 32, 35, 37, 38]. For a function $f : \mathcal{X} \to \mathcal{Y}$ define the relation (with no witness)

$$R_f = \big\{(x, y \; ; \; \bot) \; : \; y = f(x)\big\} \subseteq \mathcal{X} \times \mathcal{Y}.$$

Let i be a binary string, called an **index** in [17], that describes the relation R_f (e.g., i is a description of an arithmetic circuit for f). Informally, a preprocessing zk-SNARK for R_f operates in two phases (following a setup step).

– The first phase of a preprocessing zk-SNARK is a deterministic **encoding algorithm** to preprocess the relation R_f. The algorithm takes the index i as input, and outputs a succinct **index key** ik that represents R_f.

– In the second phase, called the online phase, the zk-SNARK prover takes as input i and a pair (x, y), where $y = f(x)$, and outputs a succinct proof π that $(x, y; \bot) \in R_f$. The verifier takes as input ik, (x, y), and the proof π, and outputs accept or reject.

In Marlin and Plonk, the size of the indexing key ik and the size of the proof π depend only on the security parameter λ. Their size is independent of the complexity of f. The verifier's run time depends logarithmically on the complexity of f and linearly on the length of (x, y). The prover's run time is quasi-linear in the complexity of f.

To build a functional commitment scheme from a preprocessing zk-SNARK one might try to use the encoding algorithm as the Commit algorithm, where the generated index key ik is the commitment string to the function. Then use the zk-SNARK prover and verifier as the Eval protocol of the functional commitment.

Unfortunately, this simple approach is insecure for a number of reasons. First, the index key ik may leak information about the committed function. However, this is easily corrected. We show how to enhance the indexing algorithms in both Marlin and Plonk so that the indexing key is a succinct hiding and binding commitment to the function. Moreover, we alter the the zk-SNARK proof so that it does not leak information about the committed function itself.

The bigger problem is that ik (which comes from the *untrusted* committer) may not represent a function. First, ik might not encode a relation R at all. Second, since zk-SNARKs support *relations*—which generalize functions—R might be a relation that is not a function; we explain this with a small example.

Example 1. Consider the relation $R := \big\{(x, y\,; w) \,:\, y = x+w,\ w \in \{0,1\}\big\} \subseteq \mathbb{F}^3$. For every $x \in \mathbb{F}$, both pairs (x, x) and $(x, x+1)$ are in the language defined by this relation: $w = 0$ and $w = 1$ are witnesses for the first and second respectively. The zk-SNARK's encoding algorithm can take a description of R as input and output an index key ik for R. Suppose the committer publishes ik as a commitment to its "function." Now for an input $x \in \mathbb{F}$ from user Bob, the committer can open the "function" as either $y \leftarrow x$ or $y \leftarrow x + 1$, and produce a valid proof for either y. This violates the evaluation binding requirement for a functional commitment.

The problem is that a zk-SNARK is designed for *general* relations, including relations that do not define a function. Therefore, to build a functional commitment from a preprocessing zk-SNARK we need an additional protocol that lets the committer prove that ik is an index key for a relation R that defines a function. We call this a *proof of function relation* or PFR. Designing an efficient PFR for Marlin and for Plonk is one of the main contributions of this paper.

Proof of function relation (PFR). A preprocessing zk-SNARK has an encoding algorithm Enc that takes as input the description i of a relation and outputs an index key $\mathsf{ik} \leftarrow \mathrm{Enc}(i)$. A PFR is a ZK proof that lets the committer efficiently prove in zero knowledge that ik is the encoding of a relation that defines a function. More precisely, for a public statement ik, the prover uses a PFR to convince the verifier that there is a witness i such that (1) $\mathsf{ik} = \mathrm{Enc}(i)$ and (2) i defines a relation R_f where for every $x \in \mathcal{X}$ there is a unique $y \in \mathcal{Y}$ for which (x, y) is in the language defined by R_f. We define this formally in Section 4.1.

In Section 4 we show how to construct a secure functional commitment scheme from a preprocessing zk-SNARK and a PFR. The challenge then is to design an efficient PFR for commonly used preprocessing zk-SNARKs.

– **Marlin:** The Marlin encoding algorithm takes as input a rank one constraint system (R1CS) which is a tuple of three matrices $A, B, C \in \mathbb{F}^{n \times n}$. We design an efficient proof that a Marlin index key ik is a commitment to an R1CS program (A, B, C) where A and B are t-strictly lower triangular (i.e. strictly lower triangular and the top t rows are zero) and C is t-diagonal (i.e. diagonal and the top t rows are zero). Here t is the size of the input. We show that this ensures that the R1CS program has a unique output for every input, without limiting the expressive power of R1CS. See Section 5.

– **Plonk:** The Plonk encoding algorithm takes as input the description of an arithmetic circuit over a field $\mathbb{F}$. We design an efficient proof that a Plonk index key ik is a commitment to an arithmetic circuit whose graph is acyclic and has no undeclared inputs. This is sufficient to ensure that every input has a unique output. See Supplement H.

This work requires new algebraic tools and new sub-protocols to prove properties of committed polynomials. These may be of independent interest.

An overview of our techniques. The Marlin and Plonk index keys contain a small number of polynomial commitments. We review what these polynomials are in Section 5 and Supplement H. Constructing an efficient PFR for these polynomial commitments requires proofs that the committed polynomials satisfy certain complex algebraic properties. We give three examples.

– **Discrete log comparison**: (Protocol 5) For both Plonk and Marlin there is a need to prove that certain values appear in a particular order. In Section 5 we devise a new efficient zk-SNARK for the following relation: Let $\mathbb{K}$ and $\mathbb{H}$ be multiplicative subgroups of the finite field $\mathbb{F}$, and let ω generate $\mathbb{H}$. Let $f, g \in \mathbb{F}[X]$ be two committed polynomials of bounded degree. The prover outputs a succinct proof that

$$\forall k \in \mathbb{K}: \quad f(k) \in \mathbb{H} \quad \text{and} \quad g(k) \in \mathbb{H} \quad \text{and} \quad \log_\omega f(k) > \log_\omega g(k).$$

– **Geometric subset check:** (part of Protocol 6) In Marlin, there is a need to prove that the top rows of the R1CS matrices $A, B, C \in \mathbb{F}^{n\times n}$ are zero. In the Marlin index key, each matrix is expressed as a commitment to three polynomials called *row*, *col*, and *val* (nine commitments total). Let $\mathbb{K}$ be multiplicative subgroup of $\mathbb{F}$ and let $\omega \in \mathbb{F}$. As we will see, to prove that the top t rows of A are zero (and similarly for B and C), the prover needs to output a succinct proof that the committed polynomial row_A satisfies

$$\forall k \in \mathbb{K}: \quad row_A(k) \in \{\omega^t, \ldots, \omega^{n-1}\}.$$

– **Representative check**: (Protocol 15) The Plonk indexing key ik contains a commitment to a *wiring* polynomial w that is a permutation of a subgroup $\mathbb{K}$ of $\mathbb{F}$. That is, $\{\mathsf{w}(k) : k \in \mathbb{K}\} = \mathbb{K}$. This w induces a permutation on $\mathbb{K}$ that can be treated as a collection of cycles. Each cycle represents a wire in the circuit. Let I be the set of declared input wires to the circuit along with the output wires from every gate. To prove that the committed circuit has no hidden inputs we design a novel zk-SNARK to prove that the set $I \subseteq \mathbb{K}$

intersects every cycle of the committed polynomial w. This is sufficient to prove that the committed circuit has no hidden inputs.

We design succinct proofs for all these properties, and more. We use them to prove that a Marlin index key is a commitment to a well formed R1CS program, and that a Plonk index key is a commitment to a well formed arithmetic circuit.

Efficiency. Our functional commitments are concretely efficient. For our Marlin-based functional commitment, commitments have the same size as Marlin's index key which is $\approx$850 bytes. Our evaluation proofs are somewhat larger: they are $\approx$21 kB; whereas Marlin's proofs are $\approx$2 kB. The overhead is due to the PFR proof. All sizes are quoted at the 128 bit security level. We stress that these sizes are *independent* of n: the complexity of the committed function. Generating an evaluation proof takes time $\tilde{O}(n)$ and verifying it takes time $O(m + \log n)$ where m is the size of inputs x and outputs y.

Future work. Our work motivates the design of efficient PFRs for other popular zk-SNARKs such as Spartan [37], Fractal [18], Ligero [1], Libra [38], and many others. Designing efficient PFRs for these will likely require new ideas.

Polynomial commitments are complete. In summary, we show that both Plonk and Marlin can be enhanced to provide an efficient functional commitment scheme for all arithmetic circuits of bounded size. Since Plonk and Marlin are built from a generic polynomial commitment scheme, we obtain a "completeness" theorem for functional commitments:

Theorem 1 (informal). *A functional commitment scheme for univariate polynomials of degree at most d suffices to construct a functional commitment scheme for all arithmetic circuits of size at most αd for some constant α. Evaluation proofs in the derived scheme have about the same length and verification complexity as evaluation proofs in the underlying polynomial commitment scheme.*

2 Preliminaries

2.1 Mathematical notation

For $n \in \mathbb{N}_{>0}$ we let $[n]$ be the sequence $(1, 2, \dots, n)$. Unless otherwise noted, we 1-index sequences. Let $\{\{\cdot\}\}$ denote a multiset. Thus, $\{\{1, 1\}\} \neq \{\{1\}\}$. We use $\|$ to denote the concatenation operator. Thus $\|_{i=1}^{n}(1, 1)$ denotes $2n$ ones. Let λ be the security parameter. A function $f(n)$ is $\mathsf{poly}(n)$ if there exists a $c \in \mathbb{N}$ such that $f(n) = O(n^c)$. If for all $c \in \mathbb{N}$, $f(n)$ is $o(n^{-c})$, then $f(n)$ is $\mathsf{negl}(n)$. We call f *negligible* and a probability that is $1 - \mathsf{negl}(n)$ *overwhelming*.

Let $\mathbb{F}$ be a field of large prime order p such that $\log(p) = \Omega(\lambda)$ and 2^k divides $(p - 1)$ for some $k \in \mathbb{N}$. For our Plonk construction, we also require 3 divides $(p - 1)$. For $\gamma \in \mathbb{F}^*$, let $\langle\gamma\rangle$ denote the set $\{\gamma^i\}_{i \in \mathbb{N}}$. We assume $\mathbb{F}$ is equipped with a canonical ordering of its elements. For a prime field this can be the order of their natural number representatives.

Let $\mathbb{F}^{(<d)}[X]$ denote the set of polynomials in formal variable X with coefficients from $\mathbb{F}$ with degree less than d. For a finite set of polynomials $P \subset \mathbb{F}[X]$, we denote the sum of the individual degrees as $\|P\| := \sum_{p \in P} \deg(P)$. The vanishing polynomial of $\mathbb{K} \subseteq \mathbb{F}$ is $z_{\mathbb{K}}(X) := \prod_{k \in \mathbb{K}}(X - k)$. When $\mathbb{K}$ is a multiplicative subgroup or a coset, then evaluating $z_{\mathbb{K}}(X)$ can be done in time logarithmic in the size of $\mathbb{K}$. For a set $\mathbb{S} \subseteq \mathbb{F}$ and a function f, $f(\mathbb{S})$ denotes the *set* $\{f(s) : s \in \mathbb{S}\}$. If $\mathbb{S}$ has a canonical ordering, then $\mathrm{seq}_{\mathbb{S}}(f)$ denotes the *sequence* $(f(s) : s \in \mathbb{S})$.

Consider two families of probability distributions, $\{D_\lambda\}_{\lambda \in \mathbb{N}}$ and $\{D'_\lambda\}_{\lambda \in \mathbb{N}}$, indexed by the security parameter λ. When unambiguous, we write $\{D\} = \{D'\}$ to denote that the distributions are the same.

2.2 Commitment schemes

A commitment scheme for messages $x \in \mathcal{X}$ is a pair $(\mathsf{Setup}, \mathsf{Commit})$ where

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$: Given the security parameter, sample public parameters. A randomized algorithm.
- $\mathsf{Commit}(\mathsf{pp}, x \in \mathcal{X}, r \in \mathsf{R}) \to c \in \mathcal{C}$: Given public parameters, a message x, and randomness r produce a commitment c to x. A deterministic algorithm.

A commitment scheme must satisfy two properties: *hiding* and *binding*:

- **Binding**: For all PPT adversaries $\mathcal{A}$:

$$\Pr\left[\begin{array}{l} x_1 \neq x_2 \land \\ \mathsf{Commit}(\mathsf{pp}, x_1, r_1) \\ = \mathsf{Commit}(\mathsf{pp}, x_2, r_2) \end{array} \middle| \begin{array}{l} \mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda) \\ (x_1, r_1, x_2, r_2) \xleftarrow{\$} \mathcal{A}(\mathsf{pp}) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

- **Perfect hiding**: For all $x, x' \in \mathcal{X}$, for $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$,

$$\{\mathsf{Commit}(\mathsf{pp}, x, r) : r \xleftarrow{\$} \mathsf{R}\} = \{\mathsf{Commit}(\mathsf{pp}, x', r') : r' \xleftarrow{\$} \mathsf{R}\}$$

2.3 Polynomial commitment schemes

A polynomial commitment scheme (PCS) [11, 17, 28] enables a prover to commit to a polynomial $f \in \mathbb{F}[X]$ with degree bound d. Later, a prover can convince a verifier that the committed polynomial f opens to $y = f(z)$ for $z \in \mathbb{F}$ and $\deg(f) < d$. For simplicity, we use a non-batched notation for our scheme (see Appendices B–D in [17] for full details).

- $\mathsf{PC.Setup}(1^\lambda, \mathbf{d} = \{d_i\}_i) \to (\mathsf{ck}, \mathsf{vk})$: Given the security parameter and a set of degree bounds $\mathbf{d}$, output a commitment key ck and verifying key vk.
- $\mathsf{PC.Commit}(\mathsf{ck}, f \in \mathbb{F}^{<d_i}[X], d_i \in \mathbf{d}, r \in \mathcal{R}) \to c$: Given the commitment key ck, polynomial f with degree less than $d_i \in \mathbf{d}$, and commitment randomness r, output a commitment c.

– $\mathsf{PC.Eval}(\mathsf{ck}, f \in \mathbb{F}^{<d_i}[X], d_i \in \mathbf{d}, r \in \mathcal{R}, z \in \mathbb{F}) \rightarrow \pi$: Given the commitment key ck, polynomial f with degree less than $d_i \in \mathbf{d}$, commitment randomness r, and evaluation point z, output an evaluation proof π.
– $\mathsf{PC.Check}(\mathsf{vk}, c, d_i \in \mathbf{d}, z \in \mathbb{F}, y \in \mathbb{F}, \pi) \rightarrow \{0,1\}$: Given verifying key vk, commitment c, degree bound d_i, commitment randomness r, evaluation point z, and claimed evaluation value y, output decision bit $\{0,1\}$.

A PCS is *secure* if it is a binding and hiding commitment with an evaluation proof that is an argument of knowledge. Evaluation proofs may also be zero-knowledge. We use the PCS from [17] (which refines [28]) and use the technique of [11] to make $\mathsf{PC.Eval}$ honest-verifier-zero-knowledge. For completeness we describe the PCS in Supplement B.

Homomorphisms For commitments respecting the same degree bound, the polynomial commitment schemes from [11, 17, 28] have commitment and randomness spaces that are linearly homomorphic. In other words, a verifier with commitments c_1 and c_2 for polynomials f_1 and f_2, both with degree bound d, can derive a commitment c_3 for a polynomial that is a linear combination of f_1 and f_2. The prover can similarly derive the commitment randomness for c_3 from the commitment randomness r_1 and r_2 for c_1 and c_2 respectively.

2.4 Interactive arguments

Let $\mathsf{Name}(\mathcal{P}(a), \mathcal{V}(b)) \rightarrow \{0,1\}$ denote an interactive protocol called Name. $\mathcal{P}$ takes input a. $\mathcal{V}$ takes input b and outputs 0 (reject) or 1 (accept). For randomized interactive machines $\mathcal{P}$ and $\mathcal{V}$, let $\langle \mathcal{P}(a), \mathcal{V}(b) \rangle$ denote the random variable that is the output of their interaction.

An interactive argument $\mathit{\Pi}$ for a relation $R \subseteq \mathcal{X} \times \mathcal{W}$ is an interactive protocol between a pair of PPT algorithms, a prover $\mathcal{P}$ and verifier $\mathcal{V}$. $\mathcal{P}(x, w)$ attempts to convince $\mathcal{V}(x)$ that it knows a w such that $(x, w) \in R$.[1] The outcome of the protocol is that the verifier accepts or rejects.

Definition 1 (Completeness). *An interactive argument $(\mathcal{P}, \mathcal{V})$ for a relation $R \subseteq \mathcal{X} \times \mathcal{W}$ is **complete**, if for all $(x, w) \in R$,* $\Pr[\langle \mathcal{P}(x, w), \mathcal{V}(x) \rangle = 1] = 1$.

Definition 2 (Knowledge-Soundness). *Let $(\mathcal{P}, \mathcal{V})$ be an interactive argument for a relation $R \subseteq \mathcal{X} \times \mathcal{W}$. If for all pairs of PPT adversaries $(\mathcal{P}_1, \mathcal{P}_2)$, there exists a PPT extractor Ext such that for $(x, \mathsf{st}) \leftarrow \mathcal{P}_1$,*

$$\Pr\left[(x, w) \in R : w \leftarrow \mathsf{Ext}^{\mathcal{P}_2(\mathsf{st})}(x)^{1}\right] \geq \Pr\left[\langle \mathcal{P}_2(\mathsf{st}), \mathcal{V}(x) \rangle = 1\right] - \epsilon$$

*where $\mathsf{Ext}^{\mathcal{P}_2(\mathsf{st})}$ denotes that Ext has oracle access to the "interactive function" $\mathcal{P}_2(\mathsf{st})$ (see [4]), then the argument has **knowledge error** ϵ. If ϵ is negligible, we say that the argument is **knowledge-sound**. If it is complete and knowledge-sound then we say that it is an **argument of knowledge**.*

[1]When there are public parameters pp, $\mathcal{P}$, $\mathcal{V}$, and Ext take in pp as well.

Definition 3 (HVZK). *An interactive argument $(\mathcal{P},\mathcal{V})$ for a relation $R \subseteq \mathcal{X} \times \mathcal{W}$ is (perfect)* ***honest verifier zero knowledge (HVZK)****, if there exists* PPT *simulator* Sim *such that for all $(x,w) \in R$, we have*

$$\{\mathsf{Sim}(x)\} = \{\mathit{View}_{\mathcal{V}}(\langle \mathcal{P}(x,w), \mathcal{V}(x)\rangle)\},$$

where $\mathit{View}_{\mathcal{V}}(\langle \mathcal{P}(x,w), \mathcal{V}(x)\rangle)$ denotes the view of the verifier, namely its randomness and the protocol transcript.

2.5 Polynomial interactive oracle proofs

Polynomial interactive oracle proofs (polyIOPs) refine interactive oracle proofs [7], themselves generalizations of probabilistically checkable proofs [2].[2] polyIOPs and similar yield succinct arguments [6, 17, 18, 32] and proof-carrying data [11]. Here, we review and give a self-contained definition.

Oracle Relation Let $\mathbb{F}$ be a field. For a constant $c \geq 1$, let $\mathcal{O} \subseteq (\mathbb{F}[X])^c$ be an oracle space, $\mathcal{X} \subseteq \{0,1\}^*$ be an instance space, and $\mathcal{W} \subseteq \{0,1\}^*$ be a witness space. An *oracle relation* $\mathcal{R} \subseteq \mathcal{O} \times \mathcal{X} \times \mathcal{W}$ is a set of triples $(\vec{o}, x, w)$. The language $\mathcal{L}(\mathcal{R})$ is the set of pairs $(\vec{o}, x)$ for which there exists a witness w such that $(\vec{o}, x, w) \in \mathcal{R}$. Given a size bound $N \in \mathbb{N}$, let $\mathcal{R}_N$ denote the restriction of $\mathcal{R}$ to oracles where $\|\vec{o}\| \leq N$.

Polynomial IOPs A polyIOP for an oracle relation $\mathcal{R}$ is an interactive proof between a prover $\mathcal{P}$ and verifier $\mathcal{V}$, where $\mathcal{P}$ sends *polynomial oracles* that $\mathcal{V}$ can subsequently access only via evaluation queries. More precisely, a Polynomial Interactive Oracle Proof (polyIOP) is a tuple

$$\mathsf{polyIOP} = (\mathcal{P},\quad \mathcal{V},\quad k \in \mathbb{N},\quad s : \mathbb{N} \to \mathbb{N},\quad d : \mathbb{N}^3 \to \mathbb{N}) \tag{1}$$

where the prover $\mathcal{P}$ and verifier $\mathcal{V}$ are PPT interactive algorithms. For a triple $(\vec{o}, x, w) \in \mathcal{R}$, the prover $\mathcal{P}$ receives inputs $(\mathbb{F}, \vec{o}, x, w)$ and $\mathcal{V}$ receives $(\mathbb{F}, x, N)$ and *input oracles* $\vec{o}$. Next, $\mathcal{P}$ and $\mathcal{V}$ engage in a k-round protocol; during the interaction, $\mathcal{P}$ can only send oracles. At any point, the verifier can query any input oracles and any of the oracles it receives from $\mathcal{P}$. After the k rounds of interaction, $\mathcal{V}$ outputs a decision in $\{0,1\}$.

The functions s and d in (1) bound the number of polynomials sent from $\mathcal{P}$, and their degrees. Specifically, in round $i \in [k]$, $\mathcal{V}$ sends a message $m_i \in \mathbb{F}^*$ to $\mathcal{P}$ and $\mathcal{P}$ sends back $s(i)$ oracles $o_{i,1}, ..., o_{i,s(i)}$ in $\mathbb{F}[X]$ to the verifier. Each oracle $o_{i,j}$ must have degree less than $d(N,i,j)$. The input oracles are denoted $\vec{o} = (o_{0,1}, ..., o_{0,s(0)})$; thus $s(0)$ gives the number of input oracles and $d(N,0,j)$ is the degree bound for oracle $o_{0,j}$. A pair $(\vec{o}, \tilde{\mathcal{P}})$ of input oracles and a prover $\tilde{\mathcal{P}}$ is *admissible* if the degree bounds specified by d are respected. That is, if for all $i \in [k] \cup \{0\}$ and $j \in [s(i)]$, $\deg(o_{i,j}) \leq d(N,i,j)$.

[2]Reed-Solomon encoded IOPs [6, 18] are closely related to polyIOPs.

Oracle polynomials are *additive*: for two oracle polynomials f and g, $\mathcal{V}$ can efficiently derive a new oracle $h := f + g$. Derived oracles can be queried.

A polyIOP can have a number of properties:

- **Completeness**: For all $(\vec{o}, x, w) \in \mathcal{R}$,

$$\Pr\left[\langle \mathcal{P}(\mathbb{F}, \vec{o}, x, w), \mathcal{V}^{\vec{o}}(\mathbb{F}, x, N)\rangle = 1\right] \leq \mathsf{negl}(\lambda)$$

- **Soundness**: For all $(\vec{o}, x) \notin \mathcal{L}(\mathcal{R})$ and PPT prover $\tilde{\mathcal{P}}$ such that $(\vec{o}, \tilde{\mathcal{P}})$ is admissible,

$$\Pr\left[\langle \tilde{\mathcal{P}}, \mathcal{V}^{\vec{o}}(\mathbb{F}, x, N)\rangle = 1\right] \leq \mathsf{negl}(\lambda)$$

- **Knowledge Soundness**: A polyIOP has knowledge error ϵ if there exists a PPT extractor Ext such that for all oracles $\vec{o}$, instances x, and PPT adversaries $\tilde{\mathcal{P}}$ such that $(\vec{o}, \tilde{\mathcal{P}})$ is admissible,

$$\Pr\left[w \leftarrow \mathsf{Ext}^{\tilde{\mathcal{P}}}(\mathbb{F}, \vec{o}, x, N) \wedge (\vec{o}, x, w) \in \mathcal{R}\right] \geq \Pr\left[\langle \tilde{\mathcal{P}}, \mathcal{V}^{\vec{o}}(\mathbb{F}, x, N)\rangle = 1\right] - \epsilon$$

  where $\mathsf{Ext}^{\tilde{\mathcal{P}}}$ means that the extractor Ext has blackbox access to the prover $\tilde{\mathcal{P}}$ as a set of next-message functions. Thus, the extractor has *rewind* access to the potentially malicious prover $\tilde{\mathcal{P}}$.
- **Perfect Honest Verifier Zero Knowledge**: There exist a PPT simulator Sim, for all $(\vec{o}, x, w) \in \mathcal{R}$,

$$\{\mathsf{Sim}(\mathbb{F}, x, N)\} = \left\{\mathrm{View}\left(\langle \mathcal{P}(\mathbb{F}, \vec{o}, x, w), \mathcal{V}^{\vec{o}}(\mathbb{F}, x, N)\rangle\right)\right\}$$

  where $\mathrm{View}\left(\langle \mathcal{P}(\mathbb{F}, \vec{o}, x, w), \mathcal{V}^{\vec{o}}(\mathbb{F}, x, N)\rangle\right)$ is the view of the honest verifier $\mathcal{V}$ during the interaction. For a q-query $\mathcal{V}$, this view comprises $(r, v_1, ..., v_q)$: the $\mathcal{V}$'s randomness and the responses to $\mathcal{V}$'s oracle queries. Note that Sim is not given the oracles $\vec{o}$.
- **Additional Properties**:
  - **Public Coin**: All verifier messages are uniformly random strings of a specified length.
  - **Non-adaptive queries**: All verifier queries are based solely on verifier randomness and inputs (i.e. do not depend on the results of other oracle queries).

Definition 4. *A* polyIOP *for relation $\mathcal{R}$ is* ***secure*** *if it is complete, knowledge sound, and has (perfect) honest-verifier zero-knowledge for that relation.*

In a polyIOP that is *public coin* and *non-adaptive*, a verifier $\mathcal{V}$ can be viewed as a pair of algorithms: a query algorithm Q_V and a decision algorithm D_V. The query algorithm outputs a query set $Q = \{(\mathsf{ID}, z)\} \leftarrow Q_V(\mathbb{F}, x, \rho_1, ..., \rho_k, r)$ (query oracle o_{ID} at z) where $\rho_1, ..., \rho_k$ are the verifier's random messages and r is additional query randomness. The decision algorithm outputs a decision bit $\{0, 1\} \leftarrow D_V(\mathbb{F}, x, \{v_q\}_{q \in Q}, \rho_1, ..., \rho_k, r)$.

Index Relations An *index relation* $R \subseteq \mathcal{I} \times \mathsf{X} \times \mathcal{W}$ is a ternary relation between an index space $\mathcal{I}$, instance space X, and witness space $\mathcal{W}$. An index $i \in \mathcal{I}$ is the explicit representation of a binary relation $\mathfrak{R} \subseteq \mathsf{X} \times \mathcal{W}$ where $x \in \mathsf{X}$ represents public inputs and $w \in \mathcal{W}$ represents witness inputs. Given a size bound $N \in \mathbb{N}$, let R_N denote the restriction of R to indices $|i| \leq N$.

Algebraic Holographic Proofs (AHPs) A constant-round *Algebraic Holographic Proof* [17] for an index relation $R \subseteq \mathcal{I} \times \mathsf{X} \times \mathcal{W}$ is a tuple

$$(\mathsf{Enc}_{\mathsf{AHP}}, \quad \mathcal{P}_{\mathsf{AHP}}, \quad \mathcal{V}_{\mathsf{AHP}}, \quad k \in \mathbb{N}, \quad d : \mathbb{N}^3 \to \mathbb{N}, \quad s : \mathbb{N} \to \mathbb{N})$$

where (1) $\mathsf{Enc}_{\mathsf{AHP}}$ is a deterministic algorithm that maps an index $i \in \mathcal{I}$ to an *encoded index* $\vec{o} \in \mathcal{O}$, and (2) the tuple $(\mathcal{P}_{\mathsf{AHP}}, \mathcal{V}_{\mathsf{AHP}}, k, d, s)$ is (syntactically) a $\mathsf{polyIOP}$ for the oracle relation

$$\mathcal{R}_{\mathsf{AHP}} := \left\{ \left(\vec{o} \in \mathcal{O}, \ x, \ (i, w) \right) : (i, x, w) \in R \right\}.$$

For $i \in \mathcal{I}$, the encoded index $\vec{o} = \mathsf{Enc}_{\mathsf{AHP}}(i)$ is a list of $s(0)$ polynomials with degree bounds $d(|i|, 0, j)$ for $j \in [s(0)]$.

Prior work [17] defines AHP security, which differs slightly from $\mathsf{polyIOP}$ security. In particular, soundness assumes that $\vec{o} = \mathsf{Enc}_{\mathsf{AHP}}(i)$, and the knowledge soundness extractor receives i instead of $\vec{o}$. It outputs w such that $(i, x, w) \in R$.

Virtual Oracles The initial oracles $\vec{o}$ and those sent by the prover are called *concrete* oracles. *Virtual oracles* [5][3] are polynomials that are not concrete oracles, but whose evaluations can be efficiently computed from the evaluations of concrete oracles. For example, $\mathcal{V}$ with oracle access to f and g can also query $h(X) = f(\alpha X) \cdot g(X) + \beta$ at z by querying $f(\alpha z)$ and $g(z)$. More formally, let $f_1, ..., f_n \in \mathbb{F}^{(<B)}[X]$ be concrete oracles. A virtual oracle $F \in \mathbb{F}^{(<D)}[X]$ has form

$$F(X) := G\left(X, h_1\left(v_1(X)\right), h_2\left(v_2(X)\right), ..., h_m\left(v_m(X)\right)\right)$$

where for $i \in [m]$, $h_i \in \{f_1, ..., f_n\}$, $v_i \in \mathbb{F}^{(<b)}[X]$, $G \in \mathbb{F}[X, X_1, ..., X_m]$, and G and $\{v_i\}_i$ are public. We are interested in virtual oracles where D is $\mathsf{negl}(\lambda)$; and $m, b, \deg(G)$, and the number of non-zero terms of G are constants.

Oracles in $\mathsf{polyIOP}$s can be substituted with virtual oracles, since the soundness of $\mathsf{polyIOP}$s depend solely on the evaluations of underlying polynomials. We prove zero knowledge for relevant protocols when $\{v_i = \alpha_i X\}_i$ for $\alpha_i \in \mathbb{F}^*$.

Compilation A *polynomial commitment scheme* [28] (PCS) enables a $\mathsf{polyIOP}$ to be compiled into a standard protocol. With a PCS, $\mathcal{P}$ *commits* to each polynomial it sends to the verifier during the $\mathsf{polyIOP}$ interaction. Upon an evaluation query, $\mathcal{P}$ can send $\mathcal{V}$ the desired evaluation, and prove to $\mathcal{V}$ that evaluation is consistent with the committed polynomial.

[3]Related to "polynomial identities" [20].

The derived protocol can be proven to have the desired security properties. For instance, one can construct a secure preprocessing argument from an AHP and a suitable PCS [17]. In the derived protocol, the encoded index $\vec{o} = \mathsf{Enc}_{\mathsf{AHP}}(i)$ becomes a tuple of polynomial commitments called an **index key** denoted ik. The ik contains polynomial commitments to the polynomials in $\vec{o}$.

In Section 4, we give a compiler that follows the same approach to produce a secure functional commitment from a suitable $\mathsf{polyIOP}$.

2.6 Arithmetic circuits

Informally, an arithmetic circuit is a directed acyclic graph of gates and wires. Wires carry values from $\mathbb{F}$. Each gate is binary, and it adds or multiplies.

Formally, an arithmetic circuit C—with n_{i} inputs, n_{g} gates, and $n_{\mathsf{o}} \leq n_{\mathsf{g}}$ outputs—is a sequence of gate tuples $(l_i, r_i, s_i)_{i=1}^{n_{\mathsf{g}}} \in ([n_{\mathsf{i}} + n_{\mathsf{g}}] \times [n_{\mathsf{i}} + n_{\mathsf{g}}] \times \{+, \times\})^{n_{\mathsf{g}}}$ subject to the constraint $l_i, r_i < i + n_{\mathsf{i}}$. For gate i, we will refer to l_i, r_i as the left and right input wire indices, $i + n_{\mathsf{i}}$ as the output wire index, and s_i as the gate selector. The set of circuit input wire indices is $[n_{\mathsf{i}}]$. Let $\mathcal{AC}_{n_{\mathsf{i}}, n_{\mathsf{g}}, n_{\mathsf{o}}}$ denote the set of arithmetic circuits with n_{i} inputs, n_{g} gates, and $n_{\mathsf{o}} \leq n_{\mathsf{g}}$ outputs.

To evaluate C on input $(x_1, \ldots, x_{n_{\mathsf{i}}}) \in \mathbb{F}^{n_{\mathsf{i}}}$, one computes (in order) $n_{\mathsf{g}} + n_{\mathsf{i}}$ wire values: $w_1, \ldots, w_{n_{\mathsf{g}} + n_{\mathsf{i}}}$. The first n_{i} wire values are just the inputs: $w_i = x_i$ for $i \in [n_{\mathsf{i}}]$. Then, for $i \in [n_{\mathsf{g}}]$, $w_{i+n_{\mathsf{i}}}$ is $w_{l_i} + w_{r_i}$, if $s_i = +$ otherwise $w_{l_i} \times w_{r_i}$. The last n_{o} wire values are the circuit output. Through evaluation, any circuit C defines a function from $\vec{x} \in \mathbb{F}^{n_{\mathsf{i}}}$ to $\vec{y} \in \mathbb{F}^{n_{\mathsf{o}}}$, with evaluation denoted as $\vec{y} = C(\vec{x})$.

3 Functional Commitments

We begin by defining what is a (function-hiding) functional commitment scheme. A functional commitment scheme allows the committer to commit to a secret function and then prove evaluations of this function.

Let $\{\mathcal{X}_\lambda\}_{\lambda \in \mathbb{N}}$ and $\{\mathcal{Y}_\lambda\}_{\lambda \in \mathbb{N}}$ be families of input and outputs spaces. Let $\{(\mathcal{F}_\lambda, \mathsf{Evaluate}_\lambda)\}_\lambda$ be a family of encoded function spaces. An *encoded function space* is a finite set $\mathcal{F}_\lambda$ of $\mathsf{poly}(\lambda)$ length strings equipped with a deterministic evaluation algorithm $\mathsf{Evaluate}_\lambda : \mathcal{F}_\lambda \times \mathcal{X}_\lambda \to \mathcal{Y}_\lambda$. We omit λ indices when unambiguous. For $f \in \mathcal{F}$ and $x \in \mathcal{X}$, we abbreviate $\mathsf{Evaluate}(f, x)$ as $f(x)$.

Examples of functions and their encodings include:

- **univariate polynomials** ($\mathbb{F}^{(<d)}[X]$): The input and output spaces are $\mathbb{F}$. The encoded function space consists of d-tuples of coefficients. $\mathsf{Evaluate}(f, x)$ evaluates the polynomial whose coefficients are the d-tuple.
- **arithmetic circuits**: The input and output spaces are $\mathbb{F}^{n_i}$ and $\mathbb{F}^{n_o}$. The encoded function space consists of $\mathsf{poly}(\lambda)$ size, directed acyclic graphs of additions and multiplications. $\mathsf{Evaluate}(f, x)$ evaluates the arithmetic circuit represented by the graph.

A functional commitment scheme FC for $\mathcal{F}$, is a tuple $(\mathsf{Setup}, \mathsf{Commit}, \mathsf{Eval})$ where

– $\mathsf{Setup}(1^\lambda, N) \to \mathsf{pp}$: Given the security parameter and max size (i.e. number of gates), sample public parameters. A randomized algorithm.
– $\mathsf{Commit}(\mathsf{pp}, f \in \mathcal{F}, r \in \mathsf{R}) \to c \in \mathcal{C}$: Given pp, an encoded function f, and randomness r, produce a commitment c to f. A deterministic algorithm.
– $\mathsf{Eval}(\mathcal{P}_\mathsf{E}(\mathsf{pp}, f \in \mathcal{F}, r \in \mathsf{R}, x \in \mathcal{X}, y \in \mathcal{Y}),\ \mathcal{V}_\mathsf{E}(\mathsf{pp}, c, x, y)) \to \{0, 1\}$: an interactive protocol for $\mathcal{P}_\mathsf{E}$ to convince $\mathcal{V}_\mathsf{E}$ that $f(x) = y$.

We call R the *randomness space* and $\mathcal{C}$ the *commitment space*. A secure functional commitment should have the following properties, captured formally in Definition 5 below:

1. *Binding*: computing distinct function encodings with equivalent commitments is infeasible.
2. *Hiding*: commitments to different function encodings are indistinguishable.
3. *Completeness*: correct evaluation proofs are always accepted.
4. *Evaluation zero-knowledge*: an evaluation proof reveals nothing other than the evaluation
5. *Knowledge soundness*: evaluation proofs show that $\mathcal{P}_\mathsf{E}$ knows a function encoding consistent with the evaluation *and* the commitment.
6. *Evaluation Binding*: A malicious prover cannot construct valid evaluation proofs for different evaluations on the same input.

The binding and hiding requirements are exactly those for classical commitments. We omit *evaluation binding*, which Supplement A shows is implied by binding and extractability.

Definition 5 (Secure functional commitment) *A functional commitment scheme is* ***secure*** *if is has the following properties:*

– ***Committing:*** *The tuple* $(\mathsf{Setup}, \mathsf{Commit})$ *is a* ***hiding*** *and* ***binding*** *commitment scheme for message space* $\mathcal{F}$ *and randomness space* R.
– ***Complete:*** Eval *is a complete protocol for the following relation:*

$$R_{eval}(\mathsf{pp}) = \{(c, x, y; f, r) : f \in \mathcal{F} \wedge f(x) = y \wedge c = \mathsf{Commit}(\mathsf{pp}, f, r)\}$$

– ***Extractable:*** Eval *is an argument of knowledge for* $R_{eval}(\mathsf{pp})$.
– ***Evaluation honest-verifier zero-knowledge:*** Eval *is an honest verifier zero knowledge protocol for* $R_{eval}(\mathsf{pp})$.

The Eval protocol is often run in parallel with multiple verifiers. By the parallel composition of HVZK protocols, this is still HVZK.

4 Functional commitments from AHPs

We will construct an efficient functional commitment scheme from a suitable algebraic holographic proof (AHP). The AHP must satisfy a non-standard property: it must be zero-knowledge for the *index* as well as the witness. In addition, we will need a $\mathsf{polyIOP}$ called a *Proof of Function Relation* (PFR).

We begin with the non-standard AHP property. Recall that an AHP for an index relation $R \subseteq \mathcal{I} \times \mathsf{X} \times \mathcal{W}$ comprises $(\mathsf{Enc}_{\mathsf{AHP}}, \mathcal{P}_{\mathsf{AHP}}, \mathcal{V}_{\mathsf{AHP}})$. While standard AHP zero-knowledge [17] requires that $\mathcal{V}_{\mathsf{AHP}}$'s view is simulatable from the index and instance, our stronger property (*index-privacy*, Definition 6) requires that the view be simulatable from the instance alone.

Definition 6 (Index-private AHP). *Let* AHP *be an* AHP *with prover* $\mathcal{P}_{\mathsf{AHP}}$ *and verifier* $\mathcal{V}_{\mathsf{AHP}}$*. For field* $\mathbb{F}$ *and* $(i, x, w) \in R$*, let* $\mathit{View}(\langle \mathcal{P}_{\mathsf{AHP}}(\mathbb{F}, i, x, w), \mathcal{V}_{\mathsf{AHP}}^{\mathsf{Enc}_{\mathsf{AHP}}(i)}(x) \rangle)$ *be the view of* $\mathcal{V}_{\mathsf{AHP}}$*.* AHP *is* ***index-private*** *if it is complete, knowledge-sound [17], and there exists a PPT simulator* $\mathcal{S}$ *such that for all* $(i, x, w) \in R$ *and field* $\mathbb{F}$*,* $\mathcal{S}(\mathbb{F}, x, 1^{|i|})$ *is indistinguishable from the view of* $\mathcal{V}_{\mathsf{AHP}}$*.*

4.1 Proof of Function Relation (PFR)

An AHP may support proofs about relations that are not functions. Thus, we need a protocol to prove that an oracle-encoded relation is a function: that every input has a unique output. To capture this, we first define the concept of a *functional set*: a subset of indices that encode functions.

Definition 7 (Functional Sets for Index Relations). *Let* $R \subseteq \mathcal{I} \times (\mathcal{X} \times \mathcal{Y}) \times \mathcal{W}$*. A subset* $\mathcal{I}_f \subseteq \mathcal{I}$ *is a* ***functional set*** *if it contains only indices for which the residual* $\mathcal{X} \times \mathcal{Y}$ *relation is a function. That is, if for all* $i \in \mathcal{I}_f$*, for all* $x \in \mathcal{X}$*, there exists a* ***unique*** $y \in \mathcal{Y}$ *such that there exists* $w \in \mathcal{W}$ *such that* $(i, (x, y), w) \in R$*. Furthermore,* $\mathcal{I}_f$ *must be equipped with a poly-time algorithm* $\mathsf{Extend}(i, x) \to (y, w)$ *such that* $(i, (x, y), w) \in R$*, for all* $(i, x) \in \mathcal{I}_f \times \mathcal{X}$*.*

A functional set $\mathcal{I}_f$ can naturally be viewed as an encoded function space with the following $\mathsf{Evaluate}$ algorithm: Let $i \in \mathcal{I}_f$ and $x \in \mathcal{X}$. $\mathsf{Evaluate}(i, x)$ does the following: (i) compute $(y, w) \leftarrow \mathsf{Extend}(i, x)$ and (ii) output y. By the definition of functional set, y is unique, so the output is deterministic.

A *Proof of Function Relation* (Definition 8) for a functional set $\mathcal{I}_f$ is a $\mathsf{polyIOP}$ Π that verifies that an oracle encodes an index in $\mathcal{I}_f$. The protocol Π must be complete, knowledge-sound, and zero-knowledge.

Definition 8 (Proof of function relation) *Let* $\mathsf{Enc}_{\mathsf{AHP}}$ *be the encoding function for an AHP with encoding function* $\mathsf{Enc}_{\mathsf{AHP}} : \mathcal{I} \to \mathcal{O}$ *and encoded index space* $\mathcal{O} = (\mathbb{F}[X])^c$*. Let* Π *be a* $\mathsf{polyIOP}$ *between* $\mathcal{P}_f$ *and* $\mathcal{V}_f$ *for the following oracle relation:*

$$R_{\mathsf{func}} = \left\{ (\vec{o} \in \mathcal{O}, \bot, i) \ : \ i \in \mathcal{I}_f \ \wedge \ \vec{o} = \mathsf{Enc}_{\mathsf{AHP}}(i) \right\}$$

The protocol Π *is a* ***proof of function relation*** *(PFR) if it is a secure* $\mathsf{polyIOP}$ *for* R_{func}*.*

In the next subsection, we show that an index-private AHP and a PFR together yield a secure functional commitment. The construction compiles these protocols into a standard interactive protocol using a polynomial commitment scheme.

4.2 Functional commitments from algebraic holographic proofs

We construct a functional commitment scheme in two parts (Construction 1). The Commit algorithm uses a hiding polynomial commitment scheme to commit to the polynomials that encode an AHP's index. The Eval protocol begins by using the PFR to ensure that (a) the commitment encodes *some* index and (b) the index lies in the functional set $\mathcal{I}_f$. Then, the AHP ensures that the prover-provided witness and function output are consistent with that index.

Construction 1 (Functional commitment compiler)
Let $\mathsf{AHP} = (\mathsf{Enc}_{\mathsf{AHP}}, \mathcal{P}_{\mathsf{AHP}}, \mathcal{V}_{\mathsf{AHP}}, k_{\mathsf{AHP}}, s_{\mathsf{AHP}}, d_{\mathsf{AHP}})$ be an AHP for index relation $R \subseteq \mathcal{I} \times (\mathcal{X} \times \mathcal{Y}) \times \mathcal{W}$. Let $\Pi = (\mathcal{P}_f, \mathcal{V}_f, k_f, s_f, d_f)$ be a proof of function relation for functional set $\mathcal{I}_f$, equipped with Extend, such that $s_{\mathsf{AHP}}(0) = s_f(0)$ and for all $i \in [s_{\mathsf{AHP}}(0)]$, $d_{\mathsf{AHP}}(N, 0, i) = d_f(N, 0, i)$.

Let $\mathsf{PC} = (\mathsf{PC.Setup}, \mathsf{PC.Commit}, \mathsf{PC.Eval}, \mathsf{PC.Check})$ be a polynomial commitment scheme. Let N be the maximum supported index size. Define

$$\mathbf{d} := \{d_{\mathsf{AHP}}(N, i, j)\}_{i \in [k_{\mathsf{AHP}}] \cup \{0\}, j \in [s_{\mathsf{AHP}}(i)]} \cup \{d_f(N, i, j)\}_{i \in [k_f], j \in [s_f(i)]}$$

Then $\mathsf{FC}_{\mathsf{AHP}, \mathcal{I}_f, \Pi, \mathsf{PC}}$ is the following tuple:

- $\mathsf{Setup}(1^\lambda, N)$: Compute $\mathbf{d}$ as above, output $\mathsf{pp} \leftarrow \mathsf{PC.Setup}(1^\lambda, \mathbf{d})$
- $\mathsf{Commit}(\mathsf{pp}, i \in \mathcal{I}_f, r \in \mathsf{R} = \mathcal{R}^{s_{\mathsf{AHP}}(0)})$:
  - Parse $(r_1, \ldots, r_{s_{\mathsf{AHP}}(0)}) = r$ and $(\mathsf{ck}, \mathsf{vk}) = \mathsf{pp}$
  - Set $\vec{o} \leftarrow \mathsf{Enc}_{\mathsf{AHP}}(i)$,
  - For $i \in s_{\mathsf{AHP}}(0)$ set $c_{o_{0,i}} \leftarrow \mathsf{PC.Commit}(\mathsf{ck}, \vec{o}[i], d_{\mathsf{AHP}}(N, 0, i), r_{o_{0,i}})$
  - Output $c \leftarrow (c_{o_{0,1}}, \ldots, c_{o_{0,s_{\mathsf{AHP}}(0)}})$.
- $\mathsf{Eval}(\mathcal{P}_{\mathsf{E}}(\mathsf{pp}, i, r, x, y), \mathcal{V}_{\mathsf{E}}(\mathsf{pp}, c, x, y))$:
  - $\mathcal{P}_{\mathsf{E}}$ and $\mathcal{V}_{\mathsf{E}}$ both parse $(\mathsf{ck}, \mathsf{vk}) = \mathsf{pp}$
  - $\mathcal{P}_{\mathsf{E}}$ computes $\vec{o} \leftarrow \mathsf{Enc}_{\mathsf{AHP}}(i)$; $(y', w) \leftarrow \mathsf{Extend}(i, x)$; abort if $y \neq y'$.
  - $\mathcal{P}_{\mathsf{E}}$ and $\mathcal{V}_{\mathsf{E}}$ run the following polynomial IOPs. Let $\mathsf{piop} \in \{\mathsf{AHP}, f\}$.
    * $\langle \mathcal{P}_f(\vec{o}, \perp, i), \mathcal{V}_f^{\vec{o}}(\perp) \rangle$ // proof of function relation
    * $\langle \mathcal{P}_{\mathsf{AHP}}(i, (x, y), w), \mathcal{V}_{\mathsf{AHP}}^{\vec{o}}((x, y)) \rangle$ // proof for R
    * For all $i \in [k_{\mathsf{piop}}]$ and $j \in s_{\mathsf{piop}}(i)$, when $\mathcal{P}_{\mathsf{piop}}$ sends oracle polynomial $o_{i,j}$, $\mathcal{P}_{\mathsf{E}}$ computes and sends
    $$c_{o_{i,j}} \leftarrow \mathsf{PC.Commit}(\mathsf{ck}, o_{i,j}, d_{\mathsf{piop}}(N, i, j), r_{o_{i,j}} \overset{\$}{\leftarrow} \mathcal{R})$$
    * When $\mathcal{V}_{\mathsf{piop}}$ derives an oracle o that is a linear combination of other oracles, $\mathcal{V}_{\mathsf{E}}$ derives c_o through the PCS homomorphism. $\mathcal{P}_{\mathsf{E}}$ similarly derives the commitment randomness r_o.
    * When $\mathcal{V}_{\mathsf{piop}}$ queries oracle polynomial o (regardless of whether o was sent by $\mathcal{P}_{\mathsf{piop}}$, encoded in c, or derived from other oracles) with degree bound d_o at $z \in \mathbb{F}$ to receive $y = o(z)$:
      · $\mathcal{V}_{\mathsf{E}}$ sends z
      · $\mathcal{P}_{\mathsf{E}}$ retrieves r_o, computes $\pi \leftarrow \mathsf{PC.Eval}(\mathsf{ck}, o, d_o, r_o, z)$, and sends π and $y \leftarrow o(z)$.
      · $\mathcal{V}_{\mathsf{E}}$ retrieves c_o and asserts $1 = \mathsf{PC.Check}(\mathsf{vk}, c_o, d_o, z, y, \pi)$

Theorem 2. *Let* AHP *be an index-private algebraic holographic proof with injective* $\mathsf{Enc}_{\mathsf{AHP}}$*, let* Π *be be a secure PFR for functional set* $\mathcal{I}_f$*, and let* PC *be a perfectly hiding PCS with knowledge-sound and HVZK evaluation. Then* $\mathsf{FC}_{\mathsf{AHP}, \mathcal{I}_f, \Pi, \mathsf{PC}}$ *(Construction 1) is a secure functional commitment scheme for function encodings* $i \in \mathcal{I}_f$*.*

Here, we sketch proofs of hiding, binding, and complete, knowledge-sound, and honest-verifier zero-knowledge evaluation. Supplement C gives full proofs that Eval is knowledge-sound and honest-verifier zero-knowledge.

Proof Sketch: Completeness follows from sub-protocol completeness. The commitment is binding because $\mathsf{Enc}_{\mathsf{AHP}}$ is injective and $\mathsf{PC.Commit}$ is binding. The commitment is hiding because $\mathsf{PC.Commit}$ is hiding.

The functional commitment extractor $\mathcal{E}_{\mathrm{FC}}$ builds on the PFR extractor $\mathcal{E}_f$, the polynomial commitment extractor $\mathcal{E}_{\mathrm{PC}}$, and the AHP extractor $\mathcal{E}_{\mathsf{AHP}}$. Throughout, it uses $\mathcal{E}_{\mathrm{PC}}$ to extract polynomials and commitment randomness for the adversary's commitments and evaluations. From the polynomials, it uses $\mathcal{E}_f$ to extract $i \in \mathcal{I}_f$. Furthermore, it uses $\mathcal{E}_{\mathsf{AHP}}$ to extract $w \in \mathcal{W}$. If these extractors succeed, $\mathcal{E}_{\mathrm{FC}}$ has $f = i \in \mathcal{I}_f$, the randomness for i's commitment, and w such that $(i, (x, y), w) \in R$, which implies that $f(x) = y$.

We build an honest-verifier zero-knowledge simulator as follows. First, we use the simulators for $\mathit{\Pi}$ and AHP to simulate oracle queries and evaluations. AHP's index-privacy is critical—we do not have the index i. Then, with the PCS evaluation simulator, we simulate PCS evaluation proofs. □

5 Functional commitments from Marlin

Marlin [17] is an algebraic holographic proof (AHP) for rank-1 constraint systems (R1CS). In this section, we use it to construct a functional commitment. First, we review R1CS and Marlin's arithmetization of R1CS. Second, we give a functional set for R1CS and a compiler from arithmetic circuits to our function set (Sec. 5.1). Third, we make Marlin index-private (Sec. 5.2). Fourth, we develop a $\mathsf{polyIOP}$ to compare the discrete logarithms of polynomial evaluations (Sec. 5.3); this is used in our proof of function relation (PFR) for Marlin (Sec. 5.4). Finally, Theorem 2 yields a functional commitment.

Relation and index. Marlin is an AHP for $R_{R1CS}(n, h)$ as in Definition 9. Thus, an index is three matrices: $(A, B, C) \in (\mathbb{F}^{n\times n})^3 = \mathcal{I}$. In what follows we use $u \circ v$ to denote component-wise product of two equal size vectors u and v.

Definition 9 (Rank-1 Constraint System (R1CS)).
For $n \in \mathbb{N}$ constraints and $h \leq n$ instance variables, the rank-1 constraint system (R1CS) index relations is:

$$R_{R1CS}(n,h) := \left\{ \left((A,B,C) \in (\mathbb{F}^{n\times n})^3, x \in \mathbb{F}^h \;;\; w \in \mathbb{F}^{n-h}\right) \;:\; \begin{array}{c} z := (x, w) \\ (Az) \circ (Bz) = Cz \end{array} \right\}$$

Arithmetization As Marlin is holographic, its verifier accesses A, B, and C through queries to polynomials that encode these matrices. The encoding uses two cyclic multiplicative subgroups of $\mathbb{F}$. Let $\mathbb{H} = \langle \omega \rangle$ be a multiplicative subgroup of $\mathbb{F}$ of order n. Let each (sparse) index matrix contain at most $m = |\mathbb{K}| = O(n)$ non-zero entries, where $\mathbb{K} = \langle \gamma \rangle$.

Each matrix is encoded by three polynomials. For $M \in \{A, B, C\}$ with entries $M_{r_i,c_i} = v_i$, let the list $(r_i, c_i, v_i)_{i=0}^{m-1} \in (\mathbb{N} \times \mathbb{N} \times \mathbb{F})^m$ represent M. Let row_M, col_M, and val_M be the unique polynomials of degree less than $|\mathbb{K}|$ such that $\text{row}_M(\gamma^i) = \omega^{r_i}$, $\text{col}_M(\gamma^i) = \omega^{c_i}$, and $\text{val}_M(\gamma^i) = v_i$,[4] for $i \in \{0, \ldots, m-1\}$. The encoded index contains row_M, col_M, and val_M for all $M \in \{A, B, C\}$.

5.1 A functional set for R1CS

Unfortunately, not all rank-1 constraint systems encode deterministic functions. More precisely, for $s, t \in \mathbb{N}_{>0}$ such that $s + t = h$, if we set $\mathcal{X} = \mathbb{F}^t$, $\mathcal{Y} = \mathbb{F}^s$, and $\mathsf{X} = \mathcal{X} \times \mathcal{Y}$, then the set of all indices $\mathcal{I} = (\mathbb{F}^{n \times n})^3$ (for the relation $R_{R1CS}(n, h)$) is not a functional set. In this section, we give a restriction of R1CS that captures bounded sized arithmetic circuits and excludes non-functions. For ease of exposition, we modify the R1CS slightly: we move the output instance variables to the *end* of the vector z.

Definition 10 (Output-final R1CS).
For $n, t, s \in \mathbb{N}$ such that $t, s, s + t \in [n]$. Let $\mathcal{I} = (\mathbb{F}^{n \times n})^3$, $\mathcal{X} = \mathbb{F}^t$, $\mathcal{Y} = \mathbb{F}^s$, $\mathsf{X} = \mathcal{X} \times \mathcal{Y}$, $\mathcal{W} = \mathbb{F}^{n-t-s}$. The index relation $R_{R1CS\text{-}f}(n, t, s) \subseteq \mathcal{I} \times \mathsf{X} \times \mathcal{W}$ is:

$$R_{R1CS\text{-}f}(n, t, s) = \left\{ \begin{pmatrix} (A, B, C) \in \mathcal{I}, \\ (x \in \mathcal{X}, y \in \mathcal{Y}) \in \mathsf{X}; w \in \mathcal{W} \end{pmatrix} : \begin{array}{c} z := (x, w, y), \\ Az \circ Bz = Cz \end{array} \right\}$$

$R_{\text{R1CS-f}}$ admits a natural functional set: t-FT (Def. 11). For indices in t-FT, each element of z beyond x is uniquely determined by the previous elements. Theorem 3 states that t-FT is a functional set for $R_{\text{R1CS-f}}$ (proof: Supplement D).

Definition 11 (Functional Triple (t-FT)).
*Let $n, t \in \mathbb{N}$ such that $t \in [n]$. A matrix $M \in \mathbb{F}^{n \times n}$ is **t-diagonal** if and only if M is a diagonal matrix, the first t entries along the diagonal are zero, and the last $n - t$ entries are nonzero. Let **t-Diag** be the set of such matrices.*

*M is **t-strictly lower triangular** if and only if M is a strictly lower triangular matrix and the first t rows are zero. Let **t-SLT** be the set of such matrices.*

*A triple of matrices $(A, B, C) \in (\mathbb{F}^{n \times n})^3$ is a **functional triple** if and only if A and B are t-SLT and C is t-Diag. Let **t-FT** be the set of such triples.*

Theorem 3. *For $R_{R1CS\text{-}f}(n, t, s)$, t-FT $\subseteq \mathcal{I}$ is a functional set.*

Compiling to t-FT To obtain a functional commitment for arithmetic circuits from a preprocessing argument for $R_{\text{R1CS-f}}$ and a proof of function relation for t-FT, we need a compiler. In Supplement D we construct $\mathsf{AC2tFT}$: a compiler from arithmetic circuits to t-FT.

Theorem 4. *For any bounded sized circuit $C \in \mathcal{AC}_{n_\mathsf{i}, n_\mathsf{g}, n_\mathsf{o}}$, $\mathsf{AC2tFT}(C) \in t$-FT. Additionally, for $x \in \mathbb{F}^{n_\mathsf{i}}$, and $y \in \mathbb{F}^{n_\mathsf{o}}$, if $y = C(x)$, then there exists $w \in \mathbb{F}^{n_\mathsf{g} - n_\mathsf{o}}$ such that $(\mathsf{AC2tFT}(C), ((1, x), y), w) \in R_{R1CS\text{-}f}(n_\mathsf{g} + n_\mathsf{i} + 1, n_\mathsf{i} + 1, n_\mathsf{o})$*

[4]Technically, $\text{val}_M(\gamma^i)$ evaluates to $v_i / f(\text{row}_M(\gamma^i), \text{col}_M(\gamma^i))$ for a public function f (a formal derivative) defined in [17]. The difference is unimportant to our protocols.

5.2 Extending Marlin

We begin with small changes to Marlin's relation and arithmetization. These changes have no effect on Marlin's security. Then, we extend Marlin to obtain the properties required by Theorem 2.

Output-final R1CS Our index relation is not R_{R1CS}, but $R_{\text{R1CS-f}}$ (Def. 10). Adapting Marlin to support the latter is straightforward; see Supplement E.1.

Index privacy Our final scheme requires an *index-private* AHP (see Theorem 2). Marlin very nearly has this property already: we discuss the necessary change in Supplement E.1. A zero-knowledge zero test (Protocol 2) is the key ingredient.

Restricting the index encoding We will only work with C matrices that are t-diagonal (Def. 11), so we restrict the encoding of C. We fix both $\text{seq}_{\mathbb{K}}(\text{row}_C)$ and $\text{seq}_{\mathbb{K}}(\text{col}_C)$ to be the sequence: $\omega^t, \omega^{t+1}, \ldots \omega^{n-1}, 1, 1, \ldots, 1$. Furthermore, we fix $\text{seq}_{\mathbb{K}}(\text{val}_C)$ to be the sequence: $C_{t,t}, C_{t+1,t+1}, \ldots, C_{n-1,n-1}, 0, 0, \ldots, 0$.

This encoding captures any t-diagonal C. Since this encoding is a restriction of Marlin's original encoding, it requires no protocol modifications.

5.3 Comparing logarithms

Recall that our functional set for R1CS (Def. 11) requires that matrices A and B are strictly lower triangular. Recall also that for each non-zero entry $M_{r,c}$ in matrix $M \in \{A, B\}$, Marlin requires that $\text{row}_M(\gamma^i) = \omega^r$ and $\text{col}_M(\gamma^i) = \omega^c$. Thus, building a PFR for Marlin requires proving $\log_\omega(\text{row}_M(\gamma^i)) > \log_\omega(\text{col}_M(\gamma^i))$. In this subsection, we develop a $\mathsf{polyIOP}$ for this relationship.

Throughout, let $\gamma \in \mathbb{F}^*$ be an element of order m that generates $\mathbb{K}$ and induces the canonical order $\{1, \gamma, \ldots, \gamma^{m-1}\}$ on $\mathbb{K}$. Additionally, $\frac{|\mathbb{K}|}{|\mathbb{F} \setminus \mathbb{K}|} \le \mathsf{negl}(\lambda)$.

Unless otherwise noted, all our $\mathsf{polyIOP}$ protocols share the following properties. First, $\mathcal{P}$ sends a constant number of polynomials to $\mathcal{V}$. Second, $\mathcal{V}$ queries those polynomials at a constant number of points. Third, they have perfect completeness. Fourth, the prover's running time is always quasi-linear in the degree of the provided polynomials and verifier time is logarithmic in $|\mathbb{K}|$.

Define $B_{\mathsf{AHP}} := \max_{i \in [k_{\mathsf{AHP}}] \cup \{0\}, j \in [s_{\mathsf{AHP}}(i)]} d_{\mathsf{AHP}}(N, i, j)$ be the max degree bound of an oracle in the Marlin AHP (where N is the instance size) [17]. Define $B := \max(B_{\mathsf{AHP}}, |\mathbb{K}| + 2)$; since $\frac{B_{\mathsf{AHP}}}{|\mathbb{F}^* \setminus \mathbb{K}|}$ and $\frac{|\mathbb{K}|+2}{|\mathbb{F}^* \setminus \mathbb{K}|}$ are $\le \mathsf{negl}(\lambda)$, $\frac{B}{|\mathbb{F}^* \setminus \mathbb{K}|} \le \mathsf{negl}(\lambda)$.

Protocol 1 (Subset over $\mathbb{K}$) Previous work [19] gives a $\mathsf{polyIOP}$ for relation

$$\mathcal{R}_{\text{subset}} := \{((f, t \in \mathbb{F}^{(<B)}[X]), \perp, \perp) : f(\mathbb{K}) \subseteq t(\mathbb{K})\}$$

assuming the $\text{seq}_{\mathbb{K}}(t)$ has all repeat elements adjacent to one another. This structured assumption will suffice for our use case.

Theorem 5. *Protocol 1 (Subset over $\mathbb{K}$) is a secure* $\mathsf{polyIOP}$ *(Definition 4) for relation $\mathcal{R}_{subset}$.*

Proof. There is no witness, so it suffices to show completeness, soundness, and honest-verifier zero-knowledge. Prior work proves completeness and soundness [19]. HVZK requires a small modification (Supplement E.2).

Zero over $\mathbb{K}$ We construct a secure polyIOP to test if a virtual oracle is zero over a multiplicative subgroup $\mathbb{K} \subseteq \mathbb{F}$. Prior works [17, 20, 21, 32] construct similar protocols. However, none provide a *secure* polyIOP for this property; indeed, all leak some extra information about the oracle being tested.[5] Here, we give a secure polyIOP for zero-testing. We explicitly support virtual oracles. Let

$$F(X) := G\Big(X, f_{j_1}(\alpha_1 X), f_{j_2}(\alpha_2 X), \ldots, f_{j_t}(\alpha_t X)\Big) \in \mathbb{F}^{(<D)}[X] \tag{2}$$

be a virtual oracle where $f_1, ..., f_n$ are concrete oracles, $j_i \in [n]$; $\alpha_i \in \mathbb{F}^*$; $G \in \mathbb{F}[X, X_1, ..., X_t]$; and D, t, $\deg(G) = d_G$, and G's monomial count are $\mathsf{negl}(\lambda)$.

Our polyIOP shows that for all $k \in \mathbb{K}$, $F(k) = 0$. This also enables equality testing over $\mathbb{K}$; oracles f and g are equal over $\mathbb{K}$ if $F = f - g$ is zero over $\mathbb{K}$.

Protocol 2 (Zero over $\mathbb{K}$)
Relation:

$$\mathcal{R}_{\text{zero}} = \Big\{\big((f_1, ..., f_n), (\vec{\alpha}, \vec{j}, G), \bot\big) : f_i \in \mathbb{F}^{(<B)}[X],\ \vec{\alpha} \in (\mathbb{F}^*)^t,\ \vec{j} \in [n]^t,$$
$$\forall k \in \mathbb{K},\ F(k) = 0 \ \text{ where } F \text{ is define in } (2)\Big\}$$

1. For $i \in [t]$, let $h_i = f_{j_i}$. $\mathcal{P}$ samples random $r_i \xleftarrow{\$} \mathbb{F}^{(<2)}[X]$, computes mask $m_i(X) = r_i(\alpha_i^{-1} X) \cdot z_{\mathbb{K}}(\alpha_i^{-1} X)$, and computes $h_i' = h_i + m_i$.
2. $\mathcal{P}$ computes $F'(X) = G\left(X, h_1'(\alpha_1 X), h_2'(\alpha_2 X), ..., h_t'(\alpha_t X)\right)$ and quotient $q_1 = F'/z_{\mathbb{K}}$. $\mathcal{P}$ sends polynomials $\{m_i\}_i$, $\{r_i\}_i$, and q_1 with degree bounds B, 2 and $d_G \cdot B - |\mathbb{K}|$ respectively.
3. For $j \in [t]$, $\mathcal{V}$ derives $h_i' = h_i + m_i$ through additive homomorphism. $\mathcal{V}$ samples $\beta_1, \beta_2, c \xleftarrow{\$} \mathbb{F}^* \setminus \mathbb{K}$ and sends c.
4. $\mathcal{P}$ computes $q_2 = r_1 + c r_2 + ... + c^{t-1} r_t$. $\mathcal{V}$ derives concrete oracle q_2 through additive homomorphism.
5. Let $M(X) = m_1(\alpha_1 X) + c m_2(\alpha_2 X) + ... + c^{t-1} m_t(\alpha_t X)$. $\mathcal{V}$ computes $z_{\mathbb{K}}(\beta_1)$, $z_{\mathbb{K}}(\beta_2)$, queries $q_1(\beta_1)$ and $q_2(\beta_2)$, and for $i \in [t]$, queries $h_i'(\alpha_i \beta_1)$ and $m_i(\alpha_i \beta_2)$. $\mathcal{V}$ asserts two identities:

$$M(\beta_2) - q_2(\beta_2) \cdot z_{\mathbb{K}}(\beta_2) \stackrel{?}{=} 0 \tag{3}$$

$$F'(\beta_1) - q_1(\beta_1) \cdot z_{\mathbb{K}}(\beta_1) \stackrel{?}{=} 0 \tag{4}$$

Theorem 6. *Protocol 2 (Zero test) is a secure* polyIOP *(Definition 4) for relation $\mathcal{R}_{zero}$.*

[5]Prior works obtain zero-knowledge for their overall protocols through other means.

Proof. There is no witness, so it suffices to show completeness, soundness, and honest-verifier zero-knowledge.

Completeness Assume $F(k) = 0$ for all $k \in \mathbb{K}$. Thus, $F(X) \equiv 0 \pmod{z_{\mathbb{K}}(X)}$. By a series of substitutions,

$$\begin{aligned} h_i'(\alpha_i X) &= (f_{j_i} + m_i)(\alpha_i X) = f_{j_i}(\alpha_i X) + m_i(\alpha_i X) \\ &= h_i\left(\alpha_i X\right) + r_i\left(\alpha_i^{-1}\alpha_i X\right) \cdot z_{\mathbb{K}}\left(\alpha_i^{-1}\alpha_i X\right) \\ &= h_i\left(\alpha_i X\right) + r_i\left(X\right) \cdot z_{\mathbb{K}}\left(X\right) \\ &\equiv h_i\left(\alpha_i X\right) \pmod{z_{\mathbb{K}}(X)} \end{aligned}$$

for $i \in [t]$. Thus, by substituting h_i' for h_i, we have $F'(X) \equiv F(X) \equiv 0 \pmod{z_{\mathbb{K}}(X)}$. Then, $z_{\mathbb{K}}|F'$, so $\mathcal{P}$'s division (Step 2) is without remainder. Identities (3) and (4) hold by construction.

Soundness Recall that the verifier $\mathcal{V}$ receives oracles $\{m_i\}_i$, $\{r_i\}_i$, q_1 with admissible degree bounds B, 2, $d_G \cdot B - |\mathbb{K}|$ respectively from $\mathcal{P}$. Define

$$W(X, C) = m_1(\alpha_1 X) + C m_2(\alpha_2 X) + ... + C^{t-1} m_t(\alpha_t X)$$

and similarly define $Q_2(X, C) = \sum_{i=1}^{t} C^{i-1} r_i(X)$.

Lemma 1. *If $z_{\mathbb{K}}(X)|W(X, C)$ and $z_{\mathbb{K}}(X)|F'(X)$, then $z_{\mathbb{K}}(X)|F(X)$.*

The first condition implies that $z_{\mathbb{K}}(X)|m_i(\alpha_i X)$ for all $i \in [t]$. Thus:

$$\begin{aligned} h_i'\left(\alpha_i X\right) &= \left(f_{j_i} + m_i\right)\left(\alpha_i X\right) = f_{j_i}\left(\alpha_i X\right) + m_i\left(\alpha_i X\right) \\ &\equiv h_i\left(\alpha_i X\right) \pmod{z_{\mathbb{K}}(X)} \end{aligned}$$

This implies that $F'(X) \equiv F(X) \pmod{z_{\mathbb{K}}(X)}$. Since $z_{\mathbb{K}}|F'(X)$ was given, we now have that $z_{\mathbb{K}}|F(X)$. □

Assume that $F(X) \not\equiv 0 \pmod{z_{\mathbb{K}}(X)}$. The converse of Lemma 1 implies that either $z_{\mathbb{K}}(X) \nmid W(X, C)$ or $z_{\mathbb{K}}(X) \nmid F'(X)$.

If $z_{\mathbb{K}}(X) \nmid W(X, C)$, then $W(X, C) - Q_2(X, C) z_{\mathbb{K}}(X) \neq 0$. Identity (3) tests this polynomial equation at random $c, \beta_2 \xleftarrow{\$} \mathbb{F}^* \backslash \mathbb{K}$. Note $\deg(W - Q_2 \cdot z_{\mathbb{K}}) \leq B + t$. Thus, (3) holds with probability less than $\frac{B+t}{|\mathbb{F}^* \backslash \mathbb{K}|}$ (by the Schwartz-Zippel lemma).

If $z_{\mathbb{K}}(X) \nmid F'(X)$, then $F'(X) - q_1(X) z_{\mathbb{K}}(X) \neq 0$. Identity (4) tests this polynomial equation at random $\beta_1 \xleftarrow{\$} \mathbb{F}^* \setminus \mathbb{K}$. We know $\max_{i \in [t]} \deg(h_i') = \max_{i \in [t]}(\deg(f_{j_i}), \deg(m_i)) < B$. Thus, $\deg(F') < d_G \cdot B$. Since $\deg(q_1) \leq d_G \cdot B - |\mathbb{K}|$, we have $\deg(F' - q_1 \cdot z_{\mathbb{K}}) \leq d_G \cdot B$, and (4) holds with probability $< \frac{d_G \cdot B}{|\mathbb{F}^* \backslash \mathbb{K}|}$ (Schwartz-Zippel).

By union bound, the probability that $\mathcal{V}$ accepts (i.e., (3) and (4) hold when $F(X) \not\equiv 0 \pmod{z_{\mathbb{K}}(X)}$) is less than $\frac{B+t}{|\mathbb{F}^* \backslash \mathbb{K}|} + \frac{d_G \cdot B}{|\mathbb{F}^* \backslash \mathbb{K}|} \leq \mathsf{negl}(\lambda)$.

HVZK Consider the view of an honest execution of the protocol given a valid F. For $i \in [t]$, $\mathcal{V}$ queries $h'_i(\alpha_i\beta_1)$ and $m_i(\alpha_i\beta_2)$. Expanding,

$$\begin{aligned} h'_i(\alpha_i\beta_1) &= f_{j_i}(\alpha_i\beta_1) + m_i(\alpha_i\beta_1) = ... + r_i(\alpha_i^{-1}\alpha_i\beta_1)z_{\mathbb{K}}(\alpha^{-1}\alpha_i\beta_1) \\ &= ... + r_i(\beta_1)z_{\mathbb{K}}(\beta_1) \\ m_i(\alpha_i\beta_2) &= r_i(\alpha_i^{-1}\alpha_i\beta_2)z_{\mathbb{K}}(\alpha_i^{-1}\alpha_i\beta_2) = r_i(\beta_2)z_{\mathbb{K}}(\beta_2) \end{aligned}$$

Since $\beta_1, \beta_2 \in \mathbb{F}^* \setminus \mathbb{K}$, we have $z_{\mathbb{K}}(\beta_1) \neq 0 \neq z_{\mathbb{K}}(\beta_2)$. Since r_i is a random linear polynomial, $r_i(\beta_1)z_{\mathbb{K}}(\beta_1)$ and $r_i(\beta_2)z_{\mathbb{K}}(\beta_2)$ are distributed independently and uniformly at random. Thus, $h'_i(\alpha_i\beta_1)$ and $m_i(\alpha_i\beta_2)$ are also distributed independently and uniformly.

The simulator Sim samples $c, \beta_1, \beta_2 \overset{\$}{\leftarrow} \mathbb{F}^* \setminus \mathbb{K}$ and for $i \in [t]$, samples $\xi_i, \mu_i \overset{\$}{\leftarrow} \mathbb{F}$. It computes $\eta = G(\beta_1, \xi_1, ..., \xi_t)$ and $\nu = \mu_1 + c\mu_2 + ... + c^{t-1}\mu_{t-1}$. It outputs simulated transcript $(c, \beta_1, \beta_2, \xi_1, ..., \xi_t, \mu_1, ..., \mu_t, \eta/z_{\mathbb{K}}(\beta_1), \nu/z_{\mathbb{K}}(\beta_2))$. This is distributed identically to the view of the honest verifier $\mathcal{V}$: $(c, \beta_1, \beta_2, h'_1(\alpha_1\beta_1), ..., h'_t(\alpha_t\beta_1), m_1(\alpha_1\beta_2), ..., m_t(\alpha_t\beta_2), q_1(\beta_1), q_2(\beta_2))$.

Non-zero over $\mathbb{K}$ We present a $\mathsf{polyIOP}$ that shows $0 \notin f(\mathbb{K})$. This protocol is a thin wrapper around the previous protocol.

Protocol 3 (Non-zero over $\mathbb{K}$)
Relation: $\mathcal{R}_{\text{non-zero}} = \{(f \in \mathbb{F}^{(<B)}[X], \perp, \perp) : \forall k \in \mathbb{K}, f(k) \neq 0\}$

1. Let $g \in \mathbb{F}^{(<|\mathbb{K}|)}[X]$ be such that $\forall k \in \mathbb{K}$, $g(k) = (f(k))^{-1}$. $\mathcal{P}$ interpolates g with admissible degree bound B and sends it to $\mathcal{V}$.
2. $\mathcal{P}$ and $\mathcal{V}$ invoke Zero over $\mathbb{K}$ to check $f \cdot g = 1$ over $\mathbb{K}$.

Theorem 7. *Protocol 3 (Non-zero test) is a secure* $\mathsf{polyIOP}$ *(Definition 4) for relation $\mathcal{R}_{\text{non-zero}}$.*

Proof. Completeness and *HVZK* follow immediately from the same properties of Protocol 2. For soundness, suppose $f(k) = 0$ for some $k \in \mathbb{K}$. Then, $f(k) \cdot g(k) - 1 \neq 0$. Thus, the zero test succeeds with negligible probability.

Geometric sequence We present a $\mathsf{polyIOP}$ that shows the sequence $\text{seq}_{\mathbb{K}}(f)$ is the concatenation of geometric sequences that share the same multiplicative factor. More formally, let $a_1, a_2, ..., a_n \in \mathbb{F}$ be initial values for a set of geometric sequences that share the same multiplicative factor $r \in \mathbb{F}^*$. Let $c_1, c_2, ..., c_n \in \mathbb{N}$, where $c_1 + ... + c_n = m$, be the number of terms in each geometric sequence. For a polynomial f, we want to verify that $f(\gamma^0), f(\gamma^1), \ldots, f(\gamma^{m-1})$ is the sequence

$$a_1, a_1 r, \ldots, a_1 r^{c_1-1}, \qquad a_2, a_2 r, \ldots, a_2 r^{c_2-1}, \qquad \ldots, \qquad a_n, a_n r, \ldots, a_n r^{c_n-1}$$

The parameters r, $\vec{a}$, and $\vec{c}$ are known to both the prover and the verifier.

Protocol 4 (Geometric Sequence Test)
$\mathcal{R}_{\text{geo}} = \{(f \in \mathbb{F}^{(<B)}[X], (r, \vec{a}, \vec{c}), \bot) : \text{seq}_{\mathbb{K}}(f) =\|_{i=1}^{n} (a_i, a_i r, \ldots, a_i r^{c_i - 1})\}$

For all $i \in [n]$, let $p_i = \sum_{j<i} c_j$.

1. $\mathcal{V}$ checks $\forall i \in [n], f(\gamma^{p_i}) \overset{?}{=} a_i$
2. $\mathcal{P}$ and $\mathcal{V}$ run Zero over $\mathbb{K}$ to check for all $k \in \mathbb{K}$,
$$(f(\gamma \cdot k) - r \cdot f(k)) \cdot \prod_{i \in [n]} (k - \gamma^{p_i + c_i - 1}) \overset{?}{=} 0$$
3. $\mathcal{V}$ outputs accept if all checks pass, otherwise reject.

Theorem 8. *Protocol 4 (Geometric sequence test) is a secure* polyIOP *(Definition 4) for relation $\mathcal{R}_{geo}$.*

Proof. There is no witness, so it suffices to show completeness, soundness, and honest-verifier zero-knowledge.

Completeness The first i identities hold because the first term of each sequence is correct. The final identity follows from two cases:

- $k \notin \{\gamma^{p_i + c_i - 1} : i \in [n]\}$: This implies that $f(\gamma \cdot k)$ and $f(k)$ are in the same geometric sequence. Thus, $f(\gamma \cdot k) = r \cdot f(k)$. Hence, the left factor of the identity is zero.
- $k \in \{\gamma^{p_i + c_i - 1} : i \in [n]\}$: Then, k is a root of the right factor. Hence, the right factor is equal to zero.

Lemma 2. *If for all $i \in [n]$, $f(\gamma^{p_i}) = a_i$ and $(f(\gamma X) - r f(X)) \prod_{i \in [n]} (X - \gamma^{p_i + c_i - 1}) \equiv 0 \pmod{z_{\mathbb{K}}(X)}$, then seq$_{\mathbb{K}}(f) =\|_{i=1}^{n} (a_i, a_i r, \ldots, a_i r^{c_i - 1})$.*

For all $i \in [n]$, we prove by induction on j that the sequence $(f(\gamma^j) : p_i \le j \le p_i + c_i - 1)$ is the geometric sequence $a_i, a_i r, ..., a_i r^{c_i - 1}$. Consider $j = p_1$: we already know that $f(\gamma^{p_i}) = a_i$.

Otherwise, consider j such that $p_i \le j < p_i + c_i - 1$. Let $k = \gamma^j$. Since $k \ne \gamma^{p_i + c_i - 1}$, the right factor of the second identity is nonzero. Thus, the left factor is zero, so $f(\gamma \cdot k) = r \cdot f(k)$; in other words, the element next in the sequence must be the current multiplied by r. Thus, by induction, the sequence $(f(\gamma^j) : p_i \le j \le p_i + c_i - 1)$ is the required geometric sequence. Since we considered an arbitrary $i \in [n]$, seq$_{\mathbb{K}}(f)$ is the required concatenation of sequences. □

Soundness If seq$_{\mathbb{K}}(f)$ is not the desired concatenation, one of Lemma 2's hypotheses must be false. If the first is false, $\mathcal{V}$ accepts with probability zero. If the second is false, $\mathcal{V}$ accepts with negligible probability (since Zero over $\mathbb{K}$ is sound). By union bound, the probability the verifier accepts is negligible.

HVZK Let tr_z be the simulated transcript outputted by the Zero over $\mathbb{K}$ simulator. The simulator Sim will output transcript $(a_1, ..., a_n) \parallel \mathsf{tr}_z$.

Discrete-log comparison We present the primary protocol for this subsection: a polyIOP for polynomials f and g that verifies:

$$f(\mathbb{K}), g(\mathbb{K}) \subseteq \langle \omega \rangle = \mathbb{H} \qquad \text{and} \qquad \forall k \in \mathbb{K}, \log_\omega(f(k)) > \log_\omega(g(k))$$

where $\mathbb{H} = \langle \omega \rangle$. We assume the existence of $\Delta \in \mathbb{F}$ such that $\Delta^2 = \omega$ and $\text{ord}(\Delta) = 2{\cdot}\text{ord}(\omega)$. At a high level, the $\mathcal{P}$ sends a polynomial s whose evaluations encode (in the exponent) the difference between the logarithms of the evaluations of f and g. Then, we take a square root of the evaluations of f, g, and s. By checking the difference equation on both the original polynomials and the square roots, we show the desired relationship.

Protocol 5 (Discrete-log Comparison)

$$\mathcal{R}_{\text{dlog}<} = \left\{ (f, g \in \mathbb{F}^{(<B)}[X], (\Delta, n), \perp) : \begin{array}{c} f(\mathbb{K}), g(\mathbb{K}) \subseteq \{1, \omega^1, \ldots, \omega^{n-1}\} \wedge \\ \forall k \in \mathbb{K}, \log_\omega(f(k)) > \log_\omega(g(k)) \end{array} \right\}$$

where $m = |\mathbb{K}|$, $\omega = \Delta^2$, $\text{ord}(\Delta) = 2n$, and $n \leq m$.

1. $\mathcal{P}$ interpolates $s \in \mathbb{F}^{(<|\mathbb{K}|)}[X]$ that agrees with f/g on $\mathbb{K}$. $\mathcal{P}$ sends s to $\mathcal{V}$ with admissible degree bound B.
2. For $b \in \{f, g, s\}$, $\mathcal{P}$ interpolates b' such that for all $k \in \mathbb{K}$, $b'(k) = \Delta^{\log_\omega(b(k))}$. $\mathcal{P}$ sends to $\mathcal{V}$ polynomials f', g', and s' with admissible degree bound B.
3. $\mathcal{P}$ interpolates and sends h with admissible degree bound B such that $\text{seq}_{\mathbb{K}}(h)$ is the following sequence: $1, \Delta, \Delta^2, ..., \Delta^{n-1}, 0, ..., 0$ with $m - n$ zeroes.
4. $\mathcal{P}$ and $\mathcal{V}$ run Zero over $\mathbb{K}$ four times to check the following equalities over $\mathbb{K}$: $f' = s' \cdot g'$, $f = (f')^2$, $g = (g')^2$, $s = (s')^2$
5. $\mathcal{P}$ and $\mathcal{V}$ run Geometric Sequence Test on h with multiplicative factor Δ, initial values $1, 0$, and $c_1 = n, c_2 = m - n$.
6. $\mathcal{P}$ and $\mathcal{V}$ run Subset over $\mathbb{K}$ three times with (p, h) for $p \in \{f', g', s'\}$.
7. $\mathcal{P}$ and $\mathcal{V}$ run Non-zero over $\mathbb{K}$ to test $f', g', s', s(X) - 1 \neq 0$ anywhere on $\mathbb{K}$.

Theorem 9. *Protocol 5 (Discrete-log comparison) is a secure* polyIOP *(Definition 4) for relation $\mathcal{R}_{dlog<}$.*

Proof. There is no witness, so it suffices to show completeness, soundness, and HVZK. The protocol is HVZK because its sub-protocols are.

Completeness For all $k \in \mathbb{K}$ and $h \in \{f, g, s\}$, $h'(k) = \Delta^{\log_\omega(h(k))}$, so $(h'(k))^2 = (\Delta^2)^{\log_\omega(h(k))} = \omega^{\log_\omega(h(k))} = h(k)$. Thus, by this and construction, step 4 succeeds. For $p \in \{f', g', s'\}$, by construction, we have $p(\mathbb{K}) \subseteq h(\mathbb{K}) = \{\Delta^{e-1} : e \in [n]\}$. Thus, step 6 succeeds. Finally, step 7 succeeds because the strict inequality $\forall k \in \mathbb{K}$, $\log_\omega(f(k)) > \log_\omega(g(k))$ implies that $s(k)$ can never equal 1 and by construction, f', g', s' cannot be zero over $\mathbb{K}$.

Soundness $\mathcal{V}$ receives oracles s, f', g', s', h from $\mathcal{P}$.

Lemma 3. *If $seq_{\mathbb{K}}(h) = \Delta^0, \Delta^1, ..., \Delta^{n-1}, 0, ..., 0$; $f' = s' \cdot g'$; $f = (f')^2$; $g = (g')^2$; $s = (s')^2$, $f'(\mathbb{K}), g'(\mathbb{K}), s'(\mathbb{K}) \subseteq h(\mathbb{K})$; and $f', g', s', s-1 \neq 0$ everywhere on $\mathbb{K}$, then $f(\mathbb{K}), g(\mathbb{K}) \subseteq \{1, \omega^1, \dots, \omega^{n-1}\}$ and $\forall k \in \mathbb{K}, \log_\omega(f(k)) > \log_\omega(g(k))$.*

Given the value of $\text{seq}_{\mathbb{K}}(h)$, we have $h(\mathbb{K}) = \{\Delta^{e-1} : e \in [n]\} \cup \{0\}$. Then, the subset relations imply that $f'(\mathbb{K}), g'(\mathbb{K}), s'(\mathbb{K}) \subseteq \{\Delta^{e-1} : e \in [n]\} \cup \{0\}$. By the non-zero property, $f'(\mathbb{K}), g'(\mathbb{K}), s'(\mathbb{K}) \subseteq \{\Delta^{e-1} : e \in [n]\}$.

Thus, for all $k \in \mathbb{K}$, there exist integers $0 \leq a, b, c < n$ such that $f'(k) = \Delta^a$, $g'(k) = \Delta^b$, $s'(k) = \Delta^c$. Because $f' = s'g'$ over $\mathbb{K}$, we know that $a \equiv b + c \pmod{2n}$; given the range of a, b, and c and $2n$, this implies that $a = b + c$ (as integers). Since $\Delta^2 = \omega$, $(f')^2 = f$, $(g')^2 = g$, and $(s')^2 = s$, we have that $\log_\omega(f(k)) = a$, $\log_\omega(g(k)) = b$, $\log_\omega(s(k)) = c$. This immediately shows that $f(\mathbb{K}), g(\mathbb{K}) \subseteq \{1, \omega, \dots, \omega^{n-1}\}$.

Furthermore, $\log_\omega(f(k)) = \log_\omega(g(k)) + \log_\omega(s(k))$. Since $\log_\omega(s(k)) = c \geq 0$ and $s(k) \neq 1$ for all $k \in \mathbb{K}$, we have $\log_\omega(f(k)) > \log_\omega(g(k))$. $\square$

Suppose $g \not\subseteq$ or $f \not\subseteq \{1, \omega^1, ..., \omega^n\}$ or $\log_\omega(f(k)) \not> \log_\omega(g(k))$. By the contrapositive of Lemma 3, we must have that either one of the equalities is false, h does not have the desired geometric sequence over $\mathbb{K}$, one of the subset relations does not hold, or one of the polynomials has a zero in $\mathbb{K}$. These properties are check by sound subprotocols in steps 4, 5, 6, and 7. Thus, by union bound, the verifier accepts with negligible probability.

5.4 Proof of function relation for t-FT

In this section, we construct a proof of function relation for the t-FT functional set. Let $n, t, s \in \mathbb{N}$ such that $t, s, s+t \in [n]$. Let (A, B, C) be an index for the relation $R_{\text{R1CS-f}}(n, t, s)$. Our protocol is a polyIOP that shows that polynomials $\{\text{row}_M, \text{col}_M, \text{val}_M\}_{M \in \{A,B,C\}}$ represent matrices $(A, B, C) \in t\text{-FT}$. That is, A and B must be t-strictly lower triangular and C must be t-diagonal.

***t*-strictly lower triangular** For $M \in \{A, B\}$, we want to show that row_M and col_M encode matrix $M \in t\text{-SLT}$. To do so, t-SLT Test shows:

1. *the matrix is strictly lower triangular*: for all $i \in \{0, \dots, m-1\}$, $\log_\omega(\text{row}_M(\gamma^i)) > \log_\omega(\text{col}_M(\gamma^i))$ and
2. *the top t rows are zero:* $\text{row}_M(\mathbb{K}) \subseteq \{\omega^t, \dots, \omega^{n-1}\}$.

To prove the first, we use Discrete-log Comparison to show (a) that the image of each polynomial over $\mathbb{K}$ is a subset of $\mathbb{H}$ and (b) that the discrete-log inequality holds. To prove the second, we build a polynomial whose image is $\{\omega^e : t \leq e \leq n-1\} \cup \{0\}$ using Geometric Sequence Test. Then, we show that image contains the image of row_M using Subset over $\mathbb{K}$.

Protocol 6 (t-SLT Test)

$$\mathcal{R}_{t\text{-SLT}} = \Big\{((\mathrm{row}_M, \mathrm{col}_M, \mathrm{val}_M), (t, \varDelta, n, \mathbb{K}), \bot) \\ : \mathrm{row}_M, \mathrm{col}_M, \mathrm{val}_M \in \mathbb{F}^{(<B)}[X], \quad \varDelta^2 = \omega \in \mathbb{F}^*, \quad t, n, \in \mathbb{N}, \\ \mathrm{row}_M(\mathbb{K}) \subseteq \{\omega^t, \dots, \omega^{n-1}\} \wedge \mathrm{col}_M(\mathbb{K}) \subseteq \mathbb{H} \\ \wedge \log_\omega(\mathrm{row}_M(\gamma^i)) > \log_\omega(\mathrm{col}_M(\gamma^i)), \forall i \in [m]\Big\}$$

1. $\mathcal{P}$ interpolates and sends polynomial h such that $\mathrm{seq}_{\mathbb{K}}(h)$ is:
$$\omega^t, \omega^{t+1}...., \omega^{n-1}, 0, 0, ..., 0$$
require h has admissible degree bound B.
2. $\mathcal{P}$ and $\mathcal{V}$ run Geometric Sequence Test on h with initial values $\omega^t, 0$, multiplicative factor ω, and $c_1 = n - t, c_2 = m - (n - t)$.
3. $\mathcal{P}$ and $\mathcal{V}$ run Subset over $\mathbb{K}$ between row_M and h.
4. $\mathcal{P}$ and $\mathcal{V}$ run Discrete-log Comparison between row_M and col_M, with parameters $(\varDelta, n = |\mathbb{H}|)$ such that $\mathrm{ord}(\varDelta) = 2n, \varDelta^2 = \omega$.

Theorem 10. *Protocol 6 (t-SLT test) is a secure* polyIOP *(Definition 4) for relation $\mathcal{R}_{t\text{-}SLT}$.*

Proof. There is no witness, so it suffices to show completeness, soundness, and HVZK. The protocol is HVZK because its sub-protocols are.

Completeness If $M \in t$-SLT, then we have

- $\mathrm{row}_M(\mathbb{K}), \mathrm{col}_M(\mathbb{K}) \subseteq \mathbb{H}$
- $\forall k \in \mathbb{K}$, $\log_\omega(\mathrm{row}_M(k)) > \log_\omega(\mathrm{col}_M(k))$
- $\mathrm{row}_M(\mathbb{K}) \subseteq \{\omega^e : t \leq e \leq n-1\}$

Since $h(\mathbb{K}) = \{\omega^e : t \leq e \leq n-1\}$, completeness follows from the completeness of Subset over $\mathbb{K}$, Geometric Sequence Test, and Discrete-log Comparison.

Soundness Suppose $\mathrm{seq}_{\mathbb{K}}(h) = \omega^t, \omega^{t+1}...., \omega^{n-1}, 0, 0, ..., 0$, $\mathrm{row}_M(\mathbb{K}) \subseteq h(\mathbb{K})$, and $\mathrm{row}_M(\mathbb{K}), \mathrm{col}_M(\mathbb{K}) \subseteq \mathbb{H}$ and $\forall k \in \mathbb{K}$, $\log_\omega(\mathrm{row}_M(k)) > \log_\omega(\mathrm{col}_M(k))$.

Then, $h(\mathbb{K}) = \{\omega^e : t \leq e \leq n-1\} \cup \{0\}$. By subset relation, we have $\mathrm{row}_M(\mathbb{K}) \subseteq \{\omega^e : t \leq e \leq n-1\} \cup \{0\}$. Since $\mathrm{row}_M(\mathbb{K}) \subseteq \mathbb{H}$, we have that $\mathrm{row}_M(\mathbb{K}) \subseteq \{\omega^e : t \leq e \leq n-1\}$. Thus, because $\forall k \in \mathbb{K}$, $\log_\omega(\mathrm{row}_M(k)) > \log_\omega(\mathrm{col}_M(k))$, the polynomials encode $M \in t$-SLT.

Thus, if the polynomials do not encode a $M \in t$-SLT, at least one of the suppositions above must be false. Since each is checked by a sound sub-protocol, the verifier rejects with overwhelming probability.

***t*-diagonal** We need to check that $(\mathrm{row}_C, \mathrm{col}_C, \mathrm{val}_C)$ encodes a matrix $C \in t$-Diag. We can restrict the prover to known row_C and col_C polynomials; specifically, we force their evaluations over $\mathbb{K}$ to be: $\omega^t, \omega^{t+1}...., \omega^{n-1}, 1, 1, ..., 1$ as described in Section 5.2. Using the Zero over $\mathbb{K}$ protocol, and the Geometric image protocol, we can verify that row_C and col_C have this form. This implies all entries of C must be at: $(\omega^t, \omega^t), (\omega^{t+1}, \omega^{t+1}), ..., (\omega^{n-1}, \omega^{n-1}), (1, 1)$.

What remains is to test that the values at coordinates not equal to $(1, 1)$ are nonzero and zero at $(1, 1)$. To do this, we test identities between val_C and a polynomial h_2 whose $\mathrm{seq}_{\mathbb{K}}(h_2)$ is: $0, 0, ..., 0, 1, 1, ..., 1$.

Protocol 7 (*t*-Diag Test)

$$\begin{aligned}\mathcal{R}_{t\text{-Diag}} = \Big\{ &((\mathrm{row}_M, \mathrm{col}_M, \mathrm{val}_M), (t, \Delta, n, \gamma, m), \perp) \\ &: \mathrm{row}_M, \mathrm{col}_M, \mathrm{val}_M \in \mathbb{F}^{(<B)}[X], \quad \Delta^2 = \omega, \gamma \in \mathbb{F}^*, \quad t, n, m, \in \mathbb{N}, \\ &\exists \vec{v} \in (\mathbb{F}^*)^{n-t}, \mathrm{seq}_{\mathbb{K}}(\mathrm{val}_M) = \vec{v} \| \vec{0} \\ &\wedge \mathrm{seq}_{\mathbb{K}}(\mathrm{row}_M) = \mathrm{seq}_{\mathbb{K}}(\mathrm{col}_M) = (\omega^t, \omega^{t+1}, \ldots, \omega^{n-1}, 1, 1, \ldots, 1) \Big\}\end{aligned}$$

1. $\mathcal{P}$ interpolates and sends two polynomials $h_1, h_2 \in \mathbb{F}^{(<|\mathbb{K}|)}[X]$ with admissible degree bound B such that
   - $\mathrm{seq}_{\mathbb{K}}(h_1)$ is the following: $\omega^t, \omega^{t+1}...., \omega^{n-1}, 0, 0, ..., 0$ in which there are $m - (n - t)$ zeroes.
   - $\mathrm{seq}_{\mathbb{K}}(h_2)$ is the following: $0, 0, .., 0, 1, 1, ..., 1$ in which there are $n - t$ zeroes and $m - (n - t)$ ones.
2. $\mathcal{P}$ and $\mathcal{V}$ run Geometric Sequence Test on h_1 with initial values $\omega^t, 0$, multiplicative factor ω, and $c_1 = n - t, c_2 = m - (n - t)$.
3. $\mathcal{P}$ and $\mathcal{V}$ run Geometric Sequence Test on h_2 with initial values $0, 1$, multiplicative factor 1, and $c_1 = n - t, c_2 = m - (n - t)$.
4. $\mathcal{V}$ derives $h = h_1 + h_2$. $\mathcal{P}$ and $\mathcal{V}$ run Zero over $\mathbb{K}$ between pairs (h, row_M) and $(\mathrm{row}_M, \mathrm{col}_M)$.
5. $\mathcal{P}$ and $\mathcal{V}$ run Zero over $\mathbb{K}$ to check for all $k \in \mathbb{K}$: $\mathrm{val}_M(k) \cdot h_2(k) \overset{?}{=} 0$.
6. $\mathcal{P}$ and $\mathcal{V}$ run Non-zero over $\mathbb{K}$ to check for all $k \in \mathbb{K}$: $\mathrm{val}_M(k) + h_2(k) \overset{?}{\neq} 0$.

Theorem 11. *Protocol 7 (t-Diag test) is a secure* polyIOP *(Definition 4) for relation $\mathcal{R}_{t\text{-Diag}}$.*

Proof. There is no witness, so it suffices to show completeness, soundness, and HVZK. The protocol is HVZK because its sub-protocols are.

Completeness Steps 2, 3, 4 follow from the completeness of Geometric Sequence Test and Zero over $\mathbb{K}$. If $\mathrm{seq}_{\mathbb{K}}(\mathrm{val}_M)$ is as described, then $\mathrm{seq}_{\mathbb{K}}(\mathrm{val}_M \cdot h_2)$ is $v_1 \cdot 0, v_2 \cdot 0, ..., v_{n-t} \cdot 0, 0, 0, ..., 0$. Thus, step 5 must pass since $\mathrm{val}_M \cdot h_2$ is equivalent to the zero polynomial over $\mathbb{K}$ and Zero over $\mathbb{K}$ is complete. Additionally,

$\text{seq}_{\mathbb{K}}(\text{val}_M + h_2)$ is $v_1 + 0, v_2 + 0, ..., v_{n-t} + 0, 1, 1, ..., 1$ which is nonzero over $\mathbb{K}$. Thus, step 6 passes with the completeness of Non-zero over $\mathbb{K}$.

Soundness Suppose $\text{seq}_{\mathbb{K}}(h_1), \text{seq}_{\mathbb{K}}(h_2)$ are the desired sequences, $h = \text{row}_M$ and $\text{row}_M = \text{col}_M$ over $\mathbb{K}$, and $\text{val}_M \cdot h_2 = 0$ over $\mathbb{K}$, and $\text{val}_M + h_2 \neq 0$ over $\mathbb{K}$.

Since $\text{row}_M = h = h_1 + h_2$, $\text{seq}_{\mathbb{K}}(\text{row}_M)$ is $\omega^t, \omega^{t+1}...., \omega^{n-1}, 1, 1, ..., 1$. This implies all nonzero entries are restricted to coordinates $(\omega^t, \omega^t), (\omega^{t+1}, \omega^{t+1})$, ..., $(\omega^{n-1}, \omega^{n-1}), (1, 1)$. Since h_2 has the desired geometric sequence and the last two properties, we know $\text{seq}_{\mathbb{K}}(\text{val}_M)$ is $v_1, v_2, ..., v_{n-t}, 0, 0, ..., 0$ where for all $i \in [n-t]$, $v_i \in \mathbb{F}^*$. Thus, the values at coordinates not equal to $(1, 1)$ are nonzero and zero at $(1, 1)$. Thus, the polynomials encode $M \in t$-Diag.

Thus, if the polynomials do not encode a $M \in t$-Diag, at least one of the assumptions above must be false. Since each is checked by a sound sub-protocol, the verifier rejects with overwhelming probability.

The proof-of-function relation We next present the main protocol of this section: a polyIOP to show that the nine polynomials $\{\text{row}_M, \text{col}_M, \text{val}_M\}_{M \in \{A,B,C\}}$ encode matrices $(A, B, C) \in t$-FT.

Protocol 8 (t-FT Test)

$$\begin{aligned}\mathcal{R}_{t\text{-FT}} = \Big\{ &\big((\text{row}_M, \text{col}_M, \text{val}_M)_{M \in \{A,B,C\}}, (t, \omega, n, \gamma, m), (A, B, C)\big) \\ &: \ \text{row}_M, \text{col}_M, \text{val}_M \in \mathbb{F}^{(<B)}[X], \quad A, B, C \in \mathbb{F}^{n \times n}, \\ &(A, B, C) \in t\text{-FT} \\ &\wedge \mathsf{Enc}(A, B, C) = (\text{row}_M, \text{col}_M, \text{val}_M)_{M \in \{A,B,C\}} \Big\}\end{aligned}$$

1. $\mathcal{P}$ and $\mathcal{V}$ run t-SLT Test on $(\text{row}_M, \text{col}_M, \text{val}_M)$ for $M \in \{A, B\}$
2. $\mathcal{P}$ and $\mathcal{V}$ run t-Diag Test on $(\text{row}_C, \text{col}_C, \text{val}_C)$

Theorem 12. *Protocol 8 (t-FT test) is a secure* polyIOP *(Definition 4) for relation $\mathcal{R}_{t\text{-}FT}$. Thus it is a secure PFR for the* MARLIN′ *AHP.*

Proof. Completeness, soundness, and zero-knowledge follow from the same properties of the subprotocols.

For knowledge-soundness the extractor $\mathcal{E}$ takes as argument the oracles $(\text{row}_M, \text{col}_M, \text{val}_M)_{M \in \{A,B,C\}}$ and the instance $(t, \Delta, n, \gamma, m)$. It initializes zero matrices A, B, and C in $\mathbb{F}^{n \times n}$. Then, for $i \in \{0, \ldots, m-1\}$, it adds val_M[6] to $M_{\log_\omega(\text{row}_M(\gamma^i)), \log_\omega(\text{col}_M(\gamma^i))}$ for each $M \in \{A, B, C\}$. Then, $\mathcal{E}$ outputs (A, B, C).

For $M \in \{A, B\}$, $\log_\omega(\text{row}_M(\gamma^i))$ is always greater than $\log_\omega(\text{col}_M(\gamma^i))$ and also greater than t (soundness of t-SLT test), so the outputs A and B are in t-SLT. Similarly, the soundness of t-Diag test, implies that output C is in t-Diag.

[6]Technically, it adds $\text{val}_M(\gamma^i) \times f(\text{row}_M(\gamma^i), \text{col}_M(\gamma^i))$. See Note 4.

Corollary 1 (FC-Marlin). *Let Π denote Protocol 8, let* MARLIN$'$ *be the extension of* MARLIN *from Section 5.2, and let* PC *be a functional commitment scheme with perfect hiding and an evaluation protocol that is PoK and HVZK. Then* $\mathsf{FC}_{\text{MARLIN}',t\text{-}FT,\Pi,\mathsf{PC}}$ *is a secure functional commitment scheme.*

Proof. Follows from Theorem 2 and Theorem 12.

6 Conclusion and future work

We defined the concept of a (function-hiding) functional commitment, and showed how to construct such schemes from a preprocessing argument and a proof of function relation (Theorem 2). In Section 5 we construct a proof of function relation (PFR) for a subset of MARLIN index keys which is expressive enough to capture all arithmetic circuits. In Supplement H we construct a PFR for PLONK. Both PFR protocols are public coin, send a constant number of polynomials, and make a constant number of queries. By combining these PFRs with their AHPs we construct two public-coin functional commitments for arithmetic circuits. Verification time is logarithmic in the number of gates and linear in the input size, prover time is quasilinear, and the proof size depends only on the security parameter. The evaluation protocols can be made non-interactive using the Fiat-Shamir heuristic. We hope future work can design efficient proofs of function relation for other proof systems.

An important direction for future work is to go beyond a proof of function relation (PFR). While a PFR is a zero knowledge proof that an index key corresponds to a function (every input has a unique output), ultimately we would like an additional ZK proof to prove that a committed function satisfies some agreed upon fairness criteria [3, 29]. Zero knowledge proofs for algorithmic fairness is an interesting direction for future work.

Acknowledgements

This work was funded by NSF, DARPA, a grant from ONR, the Simons Foundation, and a Google Cloud Platforms grant. Opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA.

References

[1] S. Ames, C. Hazay, Y. Ishai, and M. Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. *ACM CCS*, 2017.
[2] S. Arora and S. Safra. Probabilistic checking of proofs; A new characterization of NP. *FOCS*, 1992.
[3] S. Barocas, M. Hardt, and A. Narayanan. *Fairness in machine learning*. `https://fairmlbook.org/`, 2017.
[4] M. Bellare and O. Goldreich. On defining proofs of knowledge. *CRYPTO*, 1993.

[5] E. Ben-Sasson, A. Chiesa, L. Goldberg, T. Gur, M. Riabzev, and N. Spooner. Linear-size constant-query IOPs for delegating computation. *TCC*, 2019.
[6] E. Ben-Sasson, A. Chiesa, M. Riabzev, N. Spooner, M. Virza, and N. P. Ward. Aurora: Transparent succinct arguments for R1CS. *EUROCRYPT*, 2019.
[7] E. Ben-Sasson, A. Chiesa, and N. Spooner. Interactive oracle proofs. *TCC*, 2016.
[8] J. Benaloh and M. De Mare. One-way accumulators: A decentralized alternative to digital signatures. *Workshop on the Theory and Application of of Cryptographic Techniques*, 1993.
[9] D. Boneh. Building a SNARK, 2020. https://cs251.stanford.edu/lectures/lecture17.pdf.
[10] D. Boneh, B. Bünz, and B. Fisch. Batching techniques for accumulators with applications to IOPs and stateless blockchains. *CRYPTO*, 2019.
[11] D. Boneh, J. Drake, B. Fisch, and A. Gabizon. Halo infinite: Proof-carrying data from additive polynomial commitments. *CRYPTO*, 2021.
[12] J. Camenisch and A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. *CRYPTO*, 2002.
[13] M. Campanelli, D. Fiore, N. Greco, D. Kolonelos, and L. Nizzardo. Incrementally aggregatable vector commitments and applications to verifiable decentralized storage. *ASIACRYPT*, 2020.
[14] D. Catalano and D. Fiore. Vector commitments and their applications. *PKC*, 2013.
[15] D. Catalano, D. Fiore, and M. Messina. Zero-knowledge sets with short proofs. *EUROCRYPT*, 2008.
[16] M. Chase, A. Healy, A. Lysyanskaya, T. Malkin, and L. Reyzin. Mercurial commitments with applications to zero-knowledge sets. *EUROCRYPT*, 2005.
[17] A. Chiesa, Y. Hu, M. Maller, P. Mishra, N. Vesely, and N. P. Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. *EUROCRYPT*, 2020.
[18] A. Chiesa, D. Ojha, and N. Spooner. Fractal: Post-quantum and transparent recursive proofs from holography. *EUROCRYPT*, 2020.
[19] A. Gabizon and Z. J. Williamson. plookup: A simplified polynomial protocol for lookup tables, 2020. https://eprint.iacr.org/2020/315.
[20] A. Gabizon, Z. J. Williamson, and O. Ciobotaru. PLONK: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge, 2019. https://eprint.iacr.org/2019/953.
[21] R. Gennaro, C. Gentry, B. Parno, and M. Raykova. Quadratic span programs and succinct NIZKs without PCPs. *EUROCRYPT*, 2013.
[22] S. Goldwasser, Y. T. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich. Reusable garbled circuits and succinct functional encryption. *ACM STOC*, 2013.
[23] S. Gorbunov, L. Reyzin, H. Wee, and Z. Zhang. Pointproofs: Aggregating proofs for multiple vector commitments. *ACM CCS*, 2020.
[24] S. Gorbunov, V. Vaikuntanathan, and D. Wichs. Leveled fully homomorphic signatures from standard lattices. *ACM STOC*, 2015.
[25] J. Groth. On the size of pairing-based non-interactive arguments. *EUROCRYPT*, 2016.
[26] J. Groth and M. Maller. Snarky signatures: Minimal signatures of knowledge from simulation-extractable SNARKs. *CRYPTO*, 2017.
[27] R. Hertzberg. Senate bill 10, 2018. https://leginfo.legislature.ca.gov/faces/billVotesClient.xhtml?bill_id=201720180SB10.
[28] A. Kate, G. M. Zaverucha, and I. Goldberg. Constant-size commitments to polynomials and their applications. *ASIACRYPT*, 2010.
[29] M. P.-S. Kim. *A Complexity-Theoretic Perspective on Fairness*. Stanford, 2020.
[30] B. Libert, S. C. Ramanna, and M. Yung. Functional commitment schemes: From polynomial commitments to pairing-based accumulators from simple assumptions. *ICALP*, 2016.
[31] H. Lipmaa and K. Pavlyk. Succinct functional commitment for a large class of arithmetic circuits. *ASIACRYPT*, 2020.
[32] M. Maller, S. Bowe, M. Kohlweiss, and S. Meiklejohn. Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. *ACM CCS*, 2019.
[33] S. Micali, M. O. Rabin, and J. Kilian. Zero-knowledge sets. *FOCS*, 2003.
[34] S. Micali, M. O. Rabin, and S. P. Vadhan. Verifiable random functions. *FOCS*, 1999.

[35] B. Parno, J. Howell, C. Gentry, and M. Raykova. Pinocchio: Nearly practical verifiable computation. *IEEE S&P*, 2013.
[36] C. Peikert, Z. Pepin, and C. Sharp. Vector and functional commitments from lattices. *TCC*, 2021.
[37] S. Setty. Spartan: Efficient and general-purpose zkSNARKs without trusted setup. *CRYPTO*, 2020.
[38] T. Xie, J. Zhang, Y. Zhang, C. Papamanthou, and D. Song. Libra: Succinct zero-knowledge proofs with optimal prover computation. *CRYPTO*, 2019.
[39] A. C.-C. Yao. How to generate and exchange secrets (extended abstract). *FOCS*, 1986.

Supplementary Material

A Evaluation binding

Definition 12 (Evaluation Binding). *Let* $\mathsf{FC} = (\mathsf{Setup}, \mathsf{Commit}, \mathcal{P}_\mathsf{E}, \mathcal{V}_\mathsf{E})$ *be a functional commitment scheme. It is* ***evaluation binding*** *if for all pairs of* PPT *adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, the advantage* $\mathsf{AdvEB}[\mathsf{FC}, \mathcal{A}]$ *is negligible in* λ*.*

$$\mathsf{AdvEB}[\mathsf{FC}, \mathcal{A}] = \Pr\left[\begin{array}{l}\langle \mathcal{A}_2(\mathsf{st}), \mathcal{V}_\mathsf{E}(\mathsf{pp}, c, x, y)\rangle = 1 \wedge \\ \langle \mathcal{A}_2(\mathsf{st}), \mathcal{V}_\mathsf{E}(\mathsf{pp}, c, x, y')\rangle = 1 \wedge \\ y \neq y' \end{array} : \begin{array}{l}\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ (x, y, y', c, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{pp})\end{array}\right]$$

Theorem 13. *If* FC *is a functional commitment scheme that is binding and has proof-of-knowledge evaluation, then it is also evaluation binding.*

Proof. Consider an adversary for Theorem 13. With it, we build an adversary for the commitment's binding property and use the knowledge soundness of the evaluation protocol.

The adversary $\mathcal{A}_{\text{bind}}$ begins by running FC's evaluation PoK extractor $\mathcal{E}$ with arguments (pp, c, x, y) to get f, r. Then, it runs $\mathcal{E}$ with arguments (pp, c, x, y') to get f', r'. It outputs $((f, r), (f', r'))$.

Suppose, to obtain a contradiction, that $\mathsf{AdvEB}[\mathsf{FC}, \mathcal{A}]$ were non-negligible. Since

$$\Pr[\langle \mathcal{A}_2(\mathsf{st}), \mathcal{V}_\mathsf{E}(\mathsf{pp}, c, x, y)\rangle = 1] \geq \mathsf{AdvEB}[\mathsf{FC}, \mathcal{A}]$$

is non-negligible, and FC has negligible soundness error, $c = \mathsf{Commit}(f, r)$, except with negligible probability. Similarly, $c = \mathsf{Commit}(f', r')$, except with negligible probability. Since $y \neq y'$, we have $f \neq f'$. Thus, the advantage of $\mathcal{A}_{\text{bind}}$ in the binding game is non-negligible, which contradicts the binding property of the functional commitment scheme.

B Additive PCS Scheme with ZK Eval

Marlin [17] modifies the PCS from [28] to obtain an additive, perfectly hiding, binding, and extractable PCS with succinct commitments. However, we also need HVZK evaluation; fortunately [11] provides a technique for modifying the evaluation proofs to make them HVZK. HVZK Evaluation presents that technique without addressing norm bounds. While prior work [11] states and proves this protocol is HVZK, we observe that their proof shows a stronger property: *special* HVZK. That is, transcripts can be simulated even when $f(z) \neq y_f$.

Protocol 9 (HVZK Eval)
Given: An additive, hiding, and binding $\mathsf{PCS} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Verify}, \mathsf{Eval})$. Public inputs $(\mathsf{pp}, d, C_f, z, y_f)$ and private prover inputs $(f \in \mathbb{F}^{(<d)}[X], r_f)$.

1. $\mathcal{P}$ samples $g(X) \xleftarrow{\$} \mathbb{F}^{(<d)}[X]$ and computes $C_r = \mathsf{Commit}(\mathsf{ck}, g(X), d, r_g)$. $\mathcal{P}$ sends C_g and $y_g = g(z)$ to the $\mathcal{V}$.
2. $\mathcal{V}$ sends random challenge $\gamma \xleftarrow{\$} \mathbb{F}$ to $\mathcal{P}$.
3. We now invoke the additive property of the PCS. $\mathcal{P}$ computes $s := g + \gamma \cdot f$ and $r_s := r_g + \gamma \cdot r_f$. $\mathcal{P}$ and $\mathcal{V}$ derive $C_s := C_g + \gamma \cdot C_f$.
4. $\mathcal{P}$ runs $\mathsf{PC.Eval}(\mathsf{ck}, s, d, r_s, z) \to \pi$ and sends π to $\mathcal{V}$
5. $\mathcal{V}$ runs $\mathsf{PC.Check}(\mathsf{vk}, C_s, d, z, y_s = y_g + \gamma \cdot y_f, \pi)$. $\mathcal{V}$ accepts if $\mathsf{PC.Check}$ outputs accepts.

C Proof for Theorem 2

C.1 Knowledge Soundness

We prove that Construction 1 has proof-of-knowledge evaluation, assuming a polynomial commitment that is binding and has proof-of-knowledge evaluation,

Consider an adversary $\mathcal{A}$ for a functional commitment scheme constructed per Construction 1. We will show that if this adversary is convincing with non-negligible probability, there is an extractor $\mathcal{E}_{\mathrm{FC}}$ which extracts (f, r) with overwhelming probability. Since the AHP and the PFR are non-adaptive, both can be split into a *commit phase* (k rounds: in round k, $\mathcal{P}$ sends oracles $s(i)$ and $\mathcal{V}$ replies with a random challenge) and a *query phase*, where $\mathcal{V}$ queries the oracles and checks the results.

Our proof exploits an additional property of the polyIOPs presented in this paper: all oracles that are inputs to the polyIOP or are sent by $\mathcal{P}$ are either (a) directly queried by $\mathcal{V}$ or (b) efficiently computable from oracles that are queried. All protocols except the zero test protocol directly query their oracles. In the zero test protocol, each polynomial h_i for $i \in [t]$ is not queried,[7] but $h_i = h'_i - m_i$, where h'_i and m_i are queried. Thus, all oracles can be computed given the oracles that are queried at least once.

Let Q be the number of total number queries, summed across all oracles. Define $k = k_{\mathsf{AHP}} + k_f$, $s(i)$ to be

$$s(i) = \begin{cases} s_f(0) & i = 0 \\ s_f(i) & 0 < i \leq k_f \\ s_{\mathsf{AHP}}(i - k_f) & k_f < i \end{cases}$$

and $d(N, i, j)$ to be

$$d(N, i, j) = \begin{cases} d_f(N, 0, j) & i = 0 \\ d_f(N, i, j) & 0 < i \leq k_f \\ d_{\mathsf{AHP}}(N, i - k_f, j) & k_f < i \end{cases}$$

[7] This is essential for zero-knowledge.

Polynomial extractor We begin with an extractor $\mathcal{E}$ built around the functional commitment adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that computes *polynomials* consistent with the adversary's commitments and claimed evaluations. We will see that if that adversary is convincing with non-negligible probability, then $\mathcal{E}$ succeeds with overwhelming probability. It does this using an extractor $\mathcal{E}_{\text{PC}}$ for the polynomial commitment evaluation protocol; $\mathcal{E}$ exists since the polynomial commitment scheme has knowledge-sound evaluation.

$\mathcal{E}(\mathsf{pp})$ begins by running: $((c, x, y), \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{pp})$ and parsing c as polynomial commitments $(c_{0,1}, \ldots, c_{0,s(N)})$ with degree bounds $(d_{0,1} = d(N, 0, 1), \ldots, d_{0,s(0)} = d(N, 0, s(0)))$ Then, for commit round i from 1 to k: $\mathcal{E}(\mathsf{pp})$ does the following:

1. receive $\vec{c}$ from $\mathcal{A}_2$
2. parse $\vec{c}$ as polynomial commitments $(c_{i,1}, \ldots, c_{i,s(N)})$ with degree bounds $(d_{i,1} = d(N, i, 1), \ldots, d_{i,s(i)} = d(N, i, s(i)))$
3. receive randomness ρ_i from $\mathcal{V}_f(\vec{c})$
4. send ρ_i to $\mathcal{A}_2$.

Let the query $q \in [Q]$ be against oracle j_q from round i_q. Let this oracle be represented by commitment $c_q = c_{i_q,j_q}$, with purported degree bound $d_q = d_{i_q,j_q}$. Let the query be at input x_q and have claimed evaluation y_q. $\mathcal{A}_{\text{PC}}$ uses $\mathcal{E}_{\text{PC}}$ to (by re-winding the evaluation prover) extract $p_q \in \mathbb{F}^{<d_q}[X]$ and randomness $r_{i,j}$ such that $c_q = \mathsf{PC.Commit}(\mathsf{pp}, p_q, r_q)$ and $p_q(x_q) = y_q$. If $\mathcal{A}$ is convincing with non-negligible probability, than all polynomial commitment evaluation proofs are convincing with with non-negligible probability. Then, the knowledge soundness of the evaluation protocol shows that each run of $\mathcal{E}_{\text{PC}}$ succeeds with overwhelming probability. Since $Q = \mathsf{poly}(\lambda)$, all Q extractions succeed with overwhelming probability.

For $i \in \{0, \ldots, k\}$ and $j \in [s(i)]$, let q the first query to the polynomial represented by commitment $c_{i,j}$. Then $\mathcal{E}$ outputs $p_{i,j} = p_q$ and $r_{i,j} = r_q$. If $p_{i,j}$ is not queries, $\mathcal{E}$ computes it from polynomials that are, per the discussion above.

An admissible prover for the PFR From $\mathcal{E}$ we build an admissible prover $\tilde{\mathcal{P}}_f$ for the PFR. $\tilde{\mathcal{P}}_f$ is defined as follows:

1. Internally run $\mathcal{E}$, obtaining polynomials $p_{i,j}$.
2. In round $i \in [k_f]$, send polynomials $p_{i,j}$ for $j \in [s(i)]$.

An admissible prover for the AHP From $\mathcal{E}$ we also build an admissible prover $\tilde{\mathcal{P}}_{\mathsf{AHP}}$ for the AHP. $\tilde{\mathcal{P}}_{\mathsf{AHP}}$ is defined as follows:

1. Internally run $\left\langle \tilde{\mathcal{P}}_f, \mathcal{V}_f \right\rangle$, saving the $p_{i,j}$.
2. In round $i \in [k_{\mathsf{AHP}}]$, send polynomials $p_{i+k_f,j}$ for $j \in [s(i)]$.

A functional commitment extractor We describe an extractor $\mathcal{E}_{\mathrm{FC}}$ for the functional commitment knowledge soundness property. $\mathcal{E}_{\mathrm{FC}}$ uses an extractor $\mathcal{E}_f$ for the PFR and an extractor $\mathcal{E}_{\mathsf{AHP}}$ for the AHP; these exist per the knowledge soundness of the respective prover. $\mathcal{E}_{\mathrm{FC}}$ has access to $(\mathcal{A}_1, \mathcal{A}_2)$ and can rewind them. It will exploit this access through $\mathcal{E}$, $\tilde{\mathcal{P}}_f$, and $\tilde{\mathcal{P}}_{\mathsf{AHP}}$. $\mathcal{E}_{\mathrm{FC}}$ is defined as follows:

1. Obtain the first round of polynomial from $\mathcal{E}$: $o \leftarrow (p_{0,1}, \dots, p_{0,s(0)})$. Also, obtain randomness $r \leftarrow (r_{0,1}, \dots, r_{0,s(0)})$.
2. $i \leftarrow \mathcal{E}_f^{\tilde{\mathcal{P}}_f}(o)$
3. $w \leftarrow \mathcal{E}_{\mathsf{AHP}}^{\tilde{\mathcal{P}}_{\mathsf{AHP}}}(\mathbb{F}, i, (x, y))$
4. Output (i, r).

Analyzing $\mathcal{E}$, $\tilde{\mathcal{P}}_f$, $\tilde{\mathcal{P}}_{\mathsf{AHP}}$, and $\mathcal{E}_{FC}$ We define four events:

- $\mathsf{Fail}_{\mathrm{PC}}$: For some i, j $c_{i,j} \neq \mathsf{PC.Commit}(\mathsf{pp}, p_{i,j}, r_{i,j})$ or for some query q at x to oracle j in round i that yields evaluation y, $p_{i,j}(x) \neq y$.
- $\mathsf{Fail}_{\mathrm{PFR}}$: $\neg[o = \mathsf{Enc}_{\mathsf{AHP}}(i) \wedge i \in \mathcal{I}_f]$
- $\mathsf{Fail}_{\mathsf{AHP}}$: $\neg[(i, (x, y), w) \in R]$
- $\mathsf{Fail}_{\mathrm{FC}}$: $\neg[(i, (x, y), w) \in R \wedge i \in \mathcal{I}_f \wedge c = \mathsf{Commit}(\mathsf{pp}, i, r)$

First, we show that $\Pr[\mathsf{Fail}_{\mathrm{PC}}]$ is negligible assuming an extractable and binding polynomial commitment scheme. If $c_{i,j} \neq \mathsf{PC.Commit}(\mathsf{pp}, p_{i,j}, r_{i,j})$ for some i, j, then the extractor $\mathcal{E}_{\mathrm{PC}}$ for the first query q to oracle j in round i failed, which happens with negligible probability per the extractability of the commitment scheme. If some other query q' to that same oracle is inconsistent with $p_{i,j}$. That is, a polynomial $p' \neq p_{i,j}$ and randomness r' is extracted for query q', then we have $(p', r') \neq (p_{i,j}, r_{i,j})$ such that $\mathsf{PC.Commit}(\mathsf{pp}, p', r') = \mathsf{PC.Commit}(\mathsf{pp}, p_{i,j}, r_{i,j})$: a collision in the commitment scheme, which happens with negligible probability since the scheme is binding.

Second, we show that $\Pr[\mathsf{Fail}_{\mathrm{PFR}} | \neg\mathsf{Fail}_{\mathrm{PC}}]$ is negligible, assuming a knowledge-sound PFR. Since $\neg\mathsf{Fail}_{\mathrm{PC}}$ holds, $\tilde{\mathcal{P}}_f$ sends oracles consistent with the commitments sent be $\mathcal{A}$, i.e., queries to those oracles agree with evaluations sent by $\mathcal{A}$. Thus, since $\mathcal{V}$ accepts with non-negligible probability, $\mathcal{V}_f^o$ does too. Thus, the extractor $\mathcal{E}_f$ obtains (with overwhelming probability) i such that $i = \mathsf{Enc}_{\mathsf{AHP}}(o)$ and $i \in \mathcal{I}_f$.

Third, we show that $\Pr[\mathsf{Fail}_{\mathsf{AHP}} | \neg\mathsf{Fail}_{\mathrm{PC}} \wedge \neg\mathsf{Fail}_{\mathrm{PFR}}]$ is negligible, assuming a knowledge-sound AHP. Since $\neg\mathsf{Fail}_{\mathrm{PC}}$ holds, $\tilde{\mathcal{P}}_{\mathsf{AHP}}$ sends oracles consistent with the commitments sent be $\mathcal{A}$, i.e., queries to those oracles agree with evaluations sent by $\mathcal{A}$. Thus, since $o = \mathsf{Enc}_{\mathsf{AHP}}(i)$ and $\mathcal{V}$ accepts with non-negligible probability, $\mathcal{V}_{\mathsf{AHP}}^o$ does too. Thus, the extractor $\mathcal{E}_{\mathsf{AHP}}$ obtains (with overwhelming probability) w such that $(i, (x, y), w) \in R$.

Fourth, we show that $\neg\mathsf{Fail}_{\mathrm{PC}} \wedge \neg\mathsf{Fail}_{\mathrm{PFR}} \wedge \neg\mathsf{Fail}_{\mathsf{AHP}}$ imply $\neg\mathsf{Fail}_{\mathrm{FC}}$. Define $f := i$. The first non-failure gives $c = (\mathsf{PC.Commit}(\mathsf{pp}, o_i, r_i)$ for (o_i, p_i) in $(\mathsf{Enc}_{\mathsf{AHP}}(f), r)) = \mathsf{Commit}(\mathsf{pp}, f, r)$. The second non-failure gives $o = \mathsf{Enc}_{\mathsf{AHP}}(f)$. The third non-failure gives $(i, (x, y), w) \in R$, which implies $f(x) = y$.

Finally, a union bound gives

$$\Pr[\mathsf{Fail}_{\mathrm{FC}}] \leq \Pr[\mathsf{Fail}_{\mathrm{PC}}] + \Pr[\mathsf{Fail}_{\mathrm{PFR}} | \neg \mathsf{Fail}_{\mathrm{PC}}] + \Pr[\mathsf{Fail}_{\mathsf{AHP}} | \neg \mathsf{Fail}_{\mathrm{PFR}} \wedge \neg \mathsf{Fail}_{\mathrm{PC}}]$$

Since all probabilities on the right are negligible, the extractor $\mathcal{E}_{\mathrm{FC}}$ fails with negligible probability. □

C.2 Honest-Verifier Zero-Knowledge

We show that that there exists a simulator $\mathcal{S}$ such that for any $((c, x, y), (f, r)) \in R_{\mathrm{eval}}(\mathsf{pp})$,

$$\{\mathsf{Sim}(\mathsf{pp}, (c, x, y))\} = \{\mathrm{View}\,(\langle \mathcal{P}(\mathsf{pp}, (c, x, y), (f, r)), \mathcal{V}(\mathsf{pp}, (c, x, y))\rangle)\}$$

i.e., that Construction 1 has (perfect) honest-verifier zero-knowledge evaluation.

Let $\mathcal{S}_{\mathrm{PFR}}$ and $\mathcal{S}_{\mathrm{PC}}$ be simulators that witness the (perfect) honest-verifier zero knowledge of the PFR and polynomial commitment evaluation protocol respectively. Let $\mathcal{S}_{\mathsf{AHP}}$ be the simulator for the index-private AHP. The polynomial commitment must also be perfectly hiding. Let $\mathcal{F}$ be the encoded function space, which contains functions $f \in \mathcal{F}$ of size N. Recall that a transcript for the functional commitment contains commitments to polynomials, verifier challenges, and polynomial evaluations and proofs. However, a transcript for the AHP or PFR contains only verifier challenges, opaque oracles, and oracle evaluations. $\mathcal{S}(\mathsf{pp}, (c, x, y))$ does as follows:

1. Compute $\mathsf{tr}'_1 \leftarrow \mathcal{S}_{\mathsf{AHP}}(\mathbb{F}, (x, y), N)$
2. Compute $\mathsf{tr}'_2 \leftarrow \mathcal{S}_{\mathrm{PFR}}(N)$
3. Parse the combined transcripts as n_r verifier challenges: $\rho'_1, \ldots, \rho'_{n_r}$, n_e query locations: $x'_1, \ldots, x'_{n_e}$, and n_e evaluations: $y'_1, \ldots, y'_{n_e}$ (the queries are each to one of n_c opaque oracles of degrees $d_1, \ldots, d_{n_c}$, let the first n_o of these oracles be sent by the prover during the protocol, and the last $n_c - n_o$ be inputs to the protocol).
4. For $i \in [n_o]$, set $p'_i \leftarrow 0$ (subject to agreement with the evaluations), sample randomness $r'_i \xleftarrow{\$} \mathcal{R}$, and compute $c'_i \leftarrow \mathsf{PC.Commit}(\mathsf{pp}, p_i, r'_i)$.
5. For $q \in [n_e]$, use the appropriate verifier (from the AHP or PFR) compute $\pi'_q \leftarrow \mathcal{S}_{\mathrm{PC}}(\mathsf{pp}, c'_{i_q}, x'_q, y'_q)$.
6. Outputs a transcript containing the commitments $\{c'_i\}_{i \in n_o}$, the verifier challenges $\{r'_i\}_{i \in n_r}$, the evaluations $\{y'_q\}_{q \in n_e}$, and the evaluation proof transcripts $\{\pi'_q\}_{q \in n'_e}$.

Let the distribution of transcripts computed by $\mathcal{S}$ be $\mathcal{D}_0$. We show that this distribution is equivalent to the distribution of transcripts produced by the real protocol execution using a hybrid argument. In these hybrids, let f denote the real function, let $r_1, \ldots, r_{n_o}$ be the randomness with which the prover commits to polynomials $p_1, \ldots, p_{n_o}$ that is wishes to send in the protocol, and let $r_{n_o+1}, \ldots, r_{n_c}$ be commitment randomness for the polynomials $p_{n_o+1}, \ldots, p_{n_c}$ in $\mathsf{Enc}_{\mathsf{AHP}}(f)$.

In the first hybrid, $\mathcal{D}_1$, we replace tr_1' with tr_1: the transcript obtained from $\langle \mathcal{P}_{\mathsf{AHP}}(\mathbb{F}, f, (x, y), w), \mathcal{V}_{\mathsf{AHP}}^{\mathsf{Enc}_{\mathsf{AHP}}(f)}(\mathbb{F}, (x, y))\rangle$, where w is from $\mathsf{Extend}(f, x)$. Since $\mathcal{S}_{\mathsf{AHP}}$ witnesses the index-privacy of the AHP, tr_1 and tr_1' are distributed identically. Thus, $\mathcal{D}_0$ and $\mathcal{D}_1$—which contain functions of these transcripts—are equivalent.

In the second hybrid, $\mathcal{D}_2$, we replace tr_2' with tr_2: the transcript obtained from $\langle \mathcal{P}_f(\mathsf{Enc}_{\mathsf{AHP}}(f), \perp, f), \mathcal{V}_f^{\mathsf{Enc}_{\mathsf{AHP}}(f)}(\perp)\rangle$. Since $\mathcal{S}_{\mathrm{PFR}}$ is a (perfect) honest-verifier zero-knowledge simulator for the PFR, tr_2 and tr_2' are distributed identically. Thus, $\mathcal{D}_1$ and $\mathcal{D}_2$ are equivalent.

In the third hybrid, $\mathcal{D}_3$, we replace $p_1', \ldots, p_{n_o}'$ with $p_1, \ldots, p_{n_o}$ and $r_1, \ldots, r_{n_o}$ with $r_1', \ldots, r_{n_o}'$. Call the resulting commitments c_i. Since the polynomial commitment scheme is perfectly hiding, and both the r_i and r_i' are sampled uniformly, the commitment tuples $(c_1, \ldots, c_{n_o})$ and $(c_1', \ldots, c_{n_o}')$ are distributed identically. Thus, $\mathcal{D}_2$ and $\mathcal{D}_3$ are equivalent.

In the fourth hybrid, $\mathcal{D}_4$, we replace each $\pi_q' = \mathcal{S}_{\mathrm{PC}}(\mathsf{pp}, c_{i_q}, x_{i_1}', y_{i_q}')$ (for $q \in [n_e]$) with $\pi_q \leftarrow \mathsf{PC.Prove}(\mathsf{pp}, p_{i_q}, r_{i_q} x_q', y_q')$. Since the queries x_q and evaluations y_q are the same, and since the polynomial commitment scheme has *special* honest verifier zero-knowledge, each π_q is distributed identically to π_q'. Thus, $\mathcal{D}_3$ and $\mathcal{D}_4$ are equivalent.

Since the queries x_q' and evaluations y_q' are parsed from the real transcripts tr_1 and tr_2, this final hybrid $\mathcal{D}_4$ constructs the same transcripts as the real protocol. $\square$

D R1CS-f Details

D.1 *t*-FT is a Functional Set

We give a proof of Theorem 3 below.

Proof. We want to show that

$$\forall\, (A, B, C) \in t\text{-FT}, \forall x \in \mathcal{X}, \exists!\; y \in \mathcal{Y}, \exists w \in \mathcal{W},\ \big((A, B, C), (x, y), w\big) \in R_{\text{R1CS-f}}\,(n, t, s)$$

Let (A, B, C) be a functional triple. Consider an arbitrary $x \in \mathcal{X}$. We will construct a unique $w' = (w, y)$ that is determined by x and $z = x||w'$ satisfies $Az \circ Bz = Cz$. Since w' is unique, we have a unique y; this will satisfy the required condition.

Since $(A, B, C) \in t$-FT, we have the first $|x|$ rows of each are zero. Thus, any $x \in \mathcal{X}$ will satisfy the first $|x|$ constraints imposed by these rows. We will now prove by induction on the constraints imposed by the rows that w' is determined by x. The $(|x| + 1)$'th rows of A, B, and C must have the following form:

$$\begin{aligned} A_{|x|+1} &= (a_1, ..., a_{|x|}, 0, 0, ..., 0) \\ B_{|x|+1} &= (b_1, ..., b_{|x|}, 0, 0, ..., 0) \\ C_{|x|+1} &= (0, 0, ..., 0, c_1, 0, ..., 0) \end{aligned}$$

Construction 2 (Compiler from arithmetic circuits to functional R1CS)
Given: an arithmetic circuit with n_g gates, n_i inputs, and $n_\mathsf{o} \leq n_\mathsf{g}$ outputs, defined by gates $(l_i, r_i, s_i)_{i=1}^{n_\mathsf{g}}$.

Produces: An index for $R_{\text{R1CS-f}}(n_\mathsf{g} + n_\mathsf{i} + 1, n_\mathsf{i} + 1, n_\mathsf{o})$.

Procedure AC2tFT:

```
1. Initialize three square matrices $A, B, C$ over $\mathbb{F}$ of height and width $n_\mathsf{i} + n_\mathsf{o} + 1$
   with zeros everywhere.
2. For $i \in [n_\mathsf{g}]$:
   (a) Set: $C_{1+n_\mathsf{i}+i,1+n_\mathsf{i}+i} \leftarrow 1$
   (b) If $s_i = +$, set:
       • $A_{1+n_\mathsf{i}+i,1} \leftarrow 1$
       • $B_{1+n_\mathsf{i}+i,1+l_i} \leftarrow 1$
       • $B_{1+n_\mathsf{i}+i,1+r_i} \leftarrow 1$
   (c) If $s_i = \times$, set:
       • $A_{1+n_\mathsf{i}+i,1+l_i} \leftarrow 1$
       • $B_{1+n_\mathsf{i}+i,1+r_i} \leftarrow 1$
3. Output $i \leftarrow (A, B, C)$
```

where $a_i, b_i \in \mathbb{F}$ for all $i \in [|x|]$ and $c_1 \neq 0$. Thus, the $(|x|+1)$'th constraint must have the following form:

$$\langle a, x \rangle \cdot \langle b, x \rangle = c_1 w'_1$$

with $a, b \in \mathbb{F}^{|x|}$. Solving for w'_1, we see that w'_1 is fixed as a function of x. The constraints following must have the form:

$$\left\langle a', (x, w'_1, ..., w'_j) \right\rangle \cdot \left\langle b', (x, w'_1, ..., w'_j) \right\rangle = c_{j+1} w'_{j+1}$$

with $a', b' \in \mathbb{F}^{|x|+j}$ and $c_{j+1} \neq 0$ for $j \in [|w'| - 1]$. Therefore, w'_{j+1} is fixed as a function of $(x, w'_1, ..., w'_j)$. Thus, inductively, w' is determined by x. Since $w' = (w, y)$ is a unique solution, we must have y is unique. Since we considered an arbitrary $x \in \mathcal{X}$, this implies we have the required condition and t-FT is a functional set.

D.2 Compiling Arithmetic Circuits to t-FT

Construction 2 compiles from arithmetic circuits to indices for relation $R_{\text{R1CS-f}}$. For any input circuit with n_i inputs and n_g it creates matrices A, B, C, which define an R1CS index. There is one constraint (row) for each gate, and a number of additional zero constraints so that the matrices are square.

E Modifications for Marlin & plookup

E.1 Marlin

Marlin [17] (Section 5.3) gives an AHP with $\mathcal{P}_{\mathsf{AHP}}(\mathbb{F}, (A, B, C), x, w)$ and $\mathcal{V}_{\mathsf{AHP}}^{\mathsf{Enc}_{\mathsf{AHP}}(A,B,C)}(x)$ for the R1CS index relation:

$$R_{\mathsf{R1CS}} = \{((A, B, C), x, \mathbf{w}) : \text{let } z := (x, \mathbf{w}), Az \circ Bz = Cz\}$$

where $\mathsf{Enc}_{\mathsf{AHP}}(A, B, C) := \{\mathsf{row}_M, \mathsf{col}_M, \mathsf{val}_M\}_{M\in\{A,B,C\}}$ outputs a polynomial encoding of the matrices.

Output Final Modification We want to obtain an AHP for Output-final R1CS. We will modify the online protocol in Section 5.3.2 [17].

Run the Marlin AHP with $z := (x, (w, y))$; thus, the sub-AHP witness is $\mathbf{w} = (w, y)$. Define $\hat{y}(X)$ be a polynomial of degree less than $|y|$ that agrees with y on $H[> |x| + |w|]$. Add a Zero over $\mathbb{H}$ check for the following identity:

$$\overline{w} \cdot v_{H[\leq|H|-|y|]} - \hat{y} = 0$$

This checks that the last entries of $\overline{w}$ agree with y.

Index Privacy Modification The AHP for R1CS does not meet our definition of zero knowledge. In particular, we require the simulator $\mathsf{Sim}(\mathbb{F}, x, N)$ does not have access to the oracles. Marlin assumes the oracles (polynomials encoding R1CS matrices A, B, and C) are not private. We will modify the online protocol in Section 5.3.2 [17].

Since the main sum-check is blinded by a random polynomial $s(X)$ (i.e the mask; they refer to this technique as a zero knowledge sum-check), the only responses that are not simulatable are the two checks necessary for equation (8). In particular, the sum-check for $f_3(X)$ and the zero check $a(X) - b(X)f_3(X) = 0$ over $\mathbb{K}$. We simply need to 1) convert the sum-check for $f_3(X)$ to a zero knowledge sumcheck using their technique and 2) replacing the zero check with our Zero over $\mathbb{H}$.

E.2 plookup

plookup [19] gives a $\mathsf{polyIOP}$ for the relation

$$\{((f, t \in \mathbb{F}^{(<B)}[X]), \perp, \perp) : f(\mathbb{K}) \subseteq t(\mathbb{K})\}$$

Zero-Knowledge Modification The only responses consist of queries to check identities in step 6) of the protocol in Section 3. Thus, we simply check these identities by using our Zero over $\mathbb{H}$ protocol. The simulator for plookup simply invokes the Zero over $\mathbb{H}$ Sim for each identity check.

Multiple Polynomial Subset over $\mathbb{K}$

Protocol 10 (Multiple Polynomial Subset over $\mathbb{K}$) The polyIOP in [19] can be adapted to check the relation

$$\mathcal{R}_{\text{mpsub}} = \{((f, t_1, ..., t_c \in \mathbb{F}^{(<B)}[X])), \perp, \perp) : f(\mathbb{K}) \subseteq t_1(\mathbb{K}) \cup ... \cup t_c(\mathbb{K})\} \quad (5)$$

for some small constant $c \geq 1$ and polynomials t_1, ..., t_c such that $\|_{i=1}^{c} \operatorname{seq}_{\mathbb{K}}(t_i)$ has all repeat elements adjacent to one another. The requirement for t to have a structured image was implicitly assumed in [19].

We will refer to section 3 of [19]. Let $f \in \mathbb{F}^n$ and $t_i \in \mathbb{F}^d$ for $i \in [c]$. Let $s \in \mathbb{F}^{n+cd}$ be $(f, t_1, ..., t_c)$ sorted by $(t_1, ..., t_c)$. We will update the bivariate polynomials F, G to be:

$$F(\beta, \gamma) := (1+\beta)^n \prod_{i \in [n]} (\gamma + f_i) \cdot \prod_{j \in [c]} \left[\prod_{i \in [d-1]} (\gamma(1+\beta) + t_{j,i} + \beta t_{j,i+1})\right] \cdot \prod_{i \in [c-1]} \left[\gamma(1+\beta) + t_{j,d} + \beta t_{j+1,1}\right]$$

$$G(\beta, \gamma) := \prod_{i \in [n+cd-1]} (\gamma(1+\beta) + s_i + \beta s_{i+1})$$

The remaining proof and protocol (polyIOP) from section 3 of [19] follow with only minor modifications.

Theorem 14. *Protocol 10 (Multiple Polynomial Subset over $\mathbb{K}$) is a secure* polyIOP *(Definition 4) for relation $\mathcal{R}_{mpsub}$.*

Proof. Since there is no witness, it suffices to show completeness, soundness, and HVZK. We omit the proofs of completeness and soundness as they are minor changes to the respective proofs in [19]. HVZK requires the same modification as in Supplement E.2.

F polyIOPs Necessary for plonk FCS

Note our protocols from Section 5.3 can be updated to work when $\mathbb{K}$ is a coset of a multiplicative subgroup (with logarithmic evaluation time [17]) or an arbitrary subset of $\mathbb{F}^*$ (with evaluation time linear in the set size). Let the vanishing polynomial $z_S = \prod_{s \in S}(X - s)$. The vanishing polynomial $v_{\mathbb{K}}$ simply needs to be replaced with z_S instead.

F.1 polyIOP for image multiset equality

We present a polyIOP for verifying that the images of two polynomials are equal as multisets.[8] That is, that $\{\!\{f(k) : k \in \mathbb{K}\}\!\} = \{\!\{g(k) : k \in \mathbb{K}\}\!\}$. Note $\mathbb{K} = \langle\gamma\rangle$ and $|\mathbb{K}| = m$.

[8]This protocol can be modified to have perfect completeness as done in [20].

Protocol 11 (Multiset Equality over $\mathbb{K}$)
$\mathcal{R}_{ms} = \left\{ ((f, g \in \mathbb{F}^{(<B)}[X]), \perp, \perp) : \{\!\{ f(k) : k \in \mathbb{K} \}\!\} = \{\!\{ g(k) : k \in \mathbb{K} \}\!\} \right\}$

---

1. $\mathcal{V}$ sends challenge $c \xleftarrow{\$} \mathbb{F}$ to $\mathcal{P}$.
2. $\mathcal{P}$ interpolates $z(X)$ defined below and sends $z(X)$ to $\mathcal{V}$.

$$z(1) = 1, \ \forall i \in [m], z(\gamma^i) = \prod_{1 \le j < i} \frac{f(\gamma^j) - c}{g(\gamma^j) - c}$$

3. $\mathcal{V}$ queries to check if $z(1) \overset{?}{=} 1$
4. $\mathcal{P}$ and $\mathcal{V}$ run Zero over $\mathbb{K}$ to check:

$$\forall k \in \mathbb{K} : \ z(k) \cdot (f(k) - c) \overset{?}{=} (g(k) - c) \cdot z(\gamma \cdot k)$$

Theorem 15. *Protocol 11 (Multiset Equality over $\mathbb{K}$) is a secure* polyIOP *(Definition 4) for relation $\mathcal{R}_{ms}$.*

Proof. Since there is no witness, it suffices to show completeness, soundness, and HVZK.

Completeness The verifier's second check holds by z's construction. The first check holds because

$$z(1) = z(\gamma^m) = \frac{f(\gamma^0) - c}{g(\gamma^0) - c} \cdots \frac{f(\gamma^{m-1}) - c}{g(\gamma^{m-1}) - c} = 1$$

where the last equality is implied by the multiset equality.

Soundness By induction and the $z(1) = 1$ condition, the verifier knows that

$$1 = z(\gamma^m) = \frac{f(\gamma^0) - c}{g(\gamma^0) - c} \cdots \frac{f(\gamma^{m-1}) - c}{g(\gamma^{m-1}) - c}$$

for uniformly random c chosen independently of f, g. Per the Schwartz-Zippel lemma, this implies the polynomials in C, $(f(\gamma^0) - C) \cdots (f(\gamma^{m-1}) - C)$ and $(g(\gamma^0) - C) \cdots (g(\gamma^{m-1}) - C)$ are equivalent, except with the negligible probability $m/|\mathbb{F}|$. Equivalent polynomials have the same multisets of roots, so $\{\!\{ f(k) : k \in \mathbb{K} \}\!\}$ and $\{\!\{ g(k) : k \in \mathbb{K} \}\!\}$ are equal.

Zero Knowledge Let tr_z denote the simulated transcript from running the Zero over $\mathbb{K}$ Sim for the identity in step 4). The simulator for Multiset Equality over $\mathbb{K}$ outputs transcript $(1, \mathsf{tr}_z)$.

F.2 Permutation composition over $\mathbb{K}$

Previous work [20] also gives a polyIOP for the relation

$$\{((f, g, \mathsf{w} \in \mathbb{F}^{(<B)}[X]), \perp, \perp) : \forall k \in \mathbb{K}, \ f(k) = g(\mathsf{w}(k))\} \quad (6)$$

assuming w is known to be a permutation on $\mathbb{K}$ (i.e. $\mathsf{w}(\mathbb{K}) = \mathbb{K}$).

We note that Permutation Composition over $\mathbb{K}$ cannot be implemented by directly applying Zero over $\mathbb{K}$ to the polynomials $f(X)$ and $g(\mathsf{w}(X))$. The difficulty is that if g and w are of degree $|\mathbb{K}|$, then the polynomial $g(\mathsf{w}(X))$ has degree $|\mathbb{K}|^2$, and computing it will make the prover too inefficient. Instead Gabizon et al. [20] develop an elegant protocol for proving (6) where the prover only manipulates polynomials of degree $|\mathbb{K}|$. Technically, the protocol is for a slightly different relation:

$$\left\{((f, g, \mathsf{w} \in \mathbb{F}^{(<B)}[X]), \perp, \perp) : \forall k \in \mathbb{K},\ f(k) = g(\gamma^{\mathsf{w}(k)})\right\}$$

for γ that generates $\mathbb{K}$ and $\mathsf{w}(\mathbb{K}) = [|\mathbb{K}|]$. However, adapting their protocol to our relation is straightforward, We present a $\mathsf{polyIOP}$ for the relation in (6).

Protocol 12 (Permutation Composition over $\mathbb{K}$)

$$\mathcal{R}_{pcomp} = \{((f, g, \mathsf{w} \in \mathbb{F}^{(<B)}[X]), \perp, \perp) : \forall k \in \mathbb{K},\ f(k) = g(w(k))\}$$

where w is known to be a permutation on $\mathbb{K}$ (i.e. $\mathsf{w}(\mathbb{K}) = \mathbb{K}$).

---

1. $\mathcal{V}$ samples and sends $\beta \xleftarrow{\$} \mathbb{F}^* \setminus \mathbb{K}$.
2. Run Multiset Equality over $\mathbb{K}$ to show that the evaluations of $\mathsf{w}(X) + \beta f(X)$ and $X + \beta g(X)$ are equal as multisets.

Theorem 16. *Protocol 12 (Permutation Composition over $\mathbb{K}$) is a secure* $\mathsf{polyIOP}$ *(Definition 4) for relation $\mathcal{R}_{pcomp}$.*

Proof. Since there is no witness, it suffices to show completeness, soundness, and HVZK.

Completeness Since $f = g \circ \mathsf{w}$, for all $k \in \mathbb{K}$, the multisets $\{\!\{(\mathsf{w}(k), f(k)) : k \in \mathbb{K}\}\!\}$ and $\{\!\{(k, g(k)) : k \in \mathbb{K}\}\!\}$ are equal. Thus, $\{\!\{\mathsf{w}(k) + \beta f(k) : k \in \mathbb{K}\}\!\}$ and $\{\!\{k + \beta g(k) : k \in \mathbb{K}\}\!\}$ are equal.

Soundness By the soundness of the multiset check, we know

$$\{\!\{\mathsf{w}(k) + \beta f(k) : k \in \mathbb{K}\}\!\} = \{\!\{k + \beta g(k) : k \in \mathbb{K}\}\!\}$$

Since β was chosen at random, we have

$$\{\!\{(\mathsf{w}(k), f(k)) : k \in \mathbb{K}\}\!\} = \{\!\{(k, g(k)) : k \in \mathbb{K}\}\!\}$$

except with negligible probability. Thus, since w is a permutation, we have that $f = g \circ \mathsf{w}$ over $\mathbb{K}$.

Zero Knowledge Let tr_M be the simulated transcript generated by the Multiset Equality over $\mathbb{K}$. This Sim outputs simulated transcript $(\beta' \xleftarrow{\$} \mathbb{F}^* \setminus \mathbb{K}, \mathsf{tr}_M)$.

F.3 Expanded Discrete-log Comparison

We present a polyIOP to check the following about polynomials f and g for multiplicative subgroup $\mathbb{H} = \langle\omega\rangle$:

$$f(\mathbb{K}), g(\mathbb{K}) \subseteq \langle\omega\rangle = \mathbb{H} \qquad \text{and} \qquad \forall k \in \mathbb{K},\ \log_\omega(f(k)) > \log_\omega(g(k))$$

Protocol 13 (Expanded Discrete-log Comparison)

$$\mathcal{R}_{\text{edlog}<} = \left\{ ((f, g \in \mathbb{F}^{(<B)}[X], (\Delta, n), \bot) : \begin{array}{c} f(\mathbb{K}), g(\mathbb{K}) \subseteq \{1, \omega^1, \ldots, \omega^{n-1}\} \wedge \\ \forall k \in \mathbb{K}, \log_\omega(f(k)) > \log_\omega(g(k)) \end{array} \right\}$$

where $m = |\mathbb{K}|$, $\omega = \Delta^2$, $\text{ord}(\Delta) \geq 2n$ is even, and there exists a constant $c \in \mathbb{N}_{>0}$ such that $n < cm$.

---

1. $\mathcal{P}$ interpolates and sends $s \in \mathbb{F}^{(<|\mathbb{K}|)}[X]$ that agrees with f/g on $\mathbb{K}$.
2. For $b \in \{f, g, s\}$, $\mathcal{P}$ interpolates and sends b' such that for all $k \in \mathbb{K}$ $b'(k) = \Delta^{\log_\omega(b(k))}$. Let these polynomials be called f', g', and s'.
3. $\mathcal{P}$ interpolates and sends $h_1, ..., h_c$ such that $\|_{i=1}^{c} \text{seq}_{\mathbb{K}}(h_i)$ is the following sequence: $1, \Delta, \Delta^2, ..., \Delta^{n-1}, 0, ..., 0$ with $cm - n$ zeroes.
4. $\mathcal{P}$ and $\mathcal{V}$ run Zero over $\mathbb{K}$ four times to check the following equalities over $\mathbb{K}$: $f' = s' \cdot g'$, $f = (f')^2$, $g = (g')^2$, $s = (s')^2$
5. $\mathcal{P}$ and $\mathcal{V}$ run Geometric Sequence Test on $h_1, ..., h_c$ with multiplicative factor Δ. For $i \in [c-1]$, the initial value is 1 and $c_1 = m$. Let $r = cm - n$. For $i = c$, the initial values are $1, 0$ and $c_1 = m - r, c_2 = r$.
6. $\mathcal{P}$ and $\mathcal{V}$ run Multiple Polynomial Subset over $\mathbb{K}$ three times with $(p, h_1, ..., h_c)$ for $p \in \{f', g', s'\}$.
7. $\mathcal{P}$ and $\mathcal{V}$ run Nonzero over $\mathbb{K}$ to test $s(X) - 1$ is non-zero over $\mathbb{K}$.

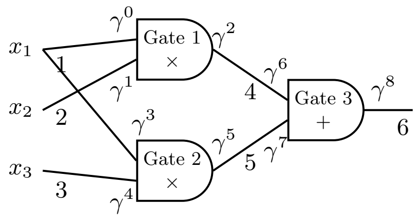
Theorem 17. *Protocol 13 (Expanded Discrete-log Comparison) is a secure* polyIOP *(Definition 4) for relation $\mathcal{R}_{edlog<}$.*

Proof. We omit the proof as it is almost identical to the proof of Theorem 9.

G Partition Lemma

Lemma 4. *Let $\mathcal{S}$ and $\mathcal{T}$ be partitions of a finite universe U such that $\mathcal{S}$ is a refinement of $\mathcal{T}$. If for all pairs $S_1 \neq S_2 \in \mathcal{S}$, there cannot exist $T \in \mathcal{T}$ such that $S_1, S_2 \subseteq T$, then $\mathcal{S} = \mathcal{T}$.*

Proof. Consider an arbitrary $S \in \mathcal{S}$. Since $\mathcal{S}$ is a refinement of $\mathcal{T}$, we have there exist $T \in \mathcal{T}$ such that $S \subseteq T$. We would like to show that $S = T$. Assume for the sake of contradiction, that there exist $t \in T$ such that $t \notin S$. Since $\mathcal{S}$ is a partition, this implies $t \in S' \neq S \in \mathcal{S}$. By refinement, we know there exists a $T' \in \mathcal{T}$ such that $S' \subseteq T'$. Since $\mathcal{T}$ is a partition, $T' = T$. Thus, we have $S' \subseteq T$.



$C = ((1, 2, \times), (1, 3, \times), (4, 5, +))$

$x \in \mathbb{K}$	γ^0	γ^1	γ^2	γ^3	γ^4	γ^5	γ^6	γ^7	γ^8
$\mathsf{w}(x) \in \mathbb{F}^{(<9)}[X]$	γ^3	γ^1	γ^6	γ^0	γ^4	γ^7	γ^2	γ^5	γ^8
$s(x) \in \mathbb{F}^{(<3)}[X]$	1	–	–	1	–	–	0	–	–

cycles of w: $(\gamma^0\,\gamma^3)(\gamma^1)(\gamma^4)(\gamma^2\,\gamma^6)(\gamma^5\,\gamma^7)(\gamma^8)$
$J = \{\gamma^0, \gamma^1, \gamma^4\}$ (input representatives)
$G = \{\gamma^2, \gamma^5, \gamma^8\}$ (all gate output wires)
$\{\mathsf{Pins}_i^{\mathsf{In}}\}_{i\in[n_\mathsf{i}]} = \{\{\gamma^0, \gamma^3\}, \{\gamma^1\}, \{\gamma^4\}\}$
$\{\mathsf{Pins}_i^{\mathsf{Out}}\}_{i\in[n_\mathsf{g}]} = \{\{\gamma^2, \gamma^6\}, \{\gamma^5, \gamma^7\}, \{\gamma^8\}\}$

Fig. 1: A circuit and Plonk arithmetization for $n_\mathsf{g} = 3$, $n_\mathsf{i} = n_\mathsf{p} = 3$, $n_\mathsf{o} = 1$.

By the statement condition, this is a contradiction and $S = T$. Thus, we have for all $S \in \mathcal{S}$, there exists a $T \in \mathcal{T}$ such that $S = T$. This implies $\mathcal{S} \subseteq \mathcal{T}$.

Consider an arbitrary $T \in \mathcal{T}$ and $t \in T$. Since $\mathcal{S}$ is a partition, there exists $S \in \mathcal{S}$ such that $t \in S$. We know there exists $T' = S$. Since $\mathcal{T}$ is a partition, we have $T = T'$. This implies $T \in \mathcal{S}$ and $\mathcal{S} = \mathcal{T}$.

H Functional commitments from Plonk

In this section, we build a PFR for Plonk: a different preprocessing argument. First, we review the index relation and arithmetization of Plonk [20]. We follow [9], but represent circuit inputs slightly differently.

The Plonk *index relation* Plonk proves evaluations of arithmetic circuits. Figure 1 will be our running example: a circuit that computes $x_1x_2 + x_1x_3$. Plonk partitions the n_i circuit inputs into $n_\mathsf{p} \le n_\mathsf{i}$ public inputs (part of the relation instance) and $n_\mathsf{i} - n_\mathsf{p}$ witness inputs (part of the relation witness). Plonk is a preprocessing argument for the index relation with

- indices $\mathcal{I} = C \in \mathcal{AC}_{n_\mathsf{i}, n_\mathsf{g}, n_\mathsf{o}}$: the circuit being evaluated
- instances $\mathsf{X} = (\vec{x}, \vec{y}) \in \mathbb{F}^{n_\mathsf{p}} \times \mathbb{F}^{n_\mathsf{o}}$: public inputs and outputs
- witnesses $\mathcal{W} = \vec{w} \in \mathbb{F}^{n_\mathsf{i} - n_\mathsf{p}}$: witness inputs

defined by

$$R_{n_\mathsf{p}, n_\mathsf{i}, n_\mathsf{g}, n_\mathsf{o}} = \{(C, (\vec{x}, \vec{y}); \vec{w}) : \vec{y} = C(\vec{x} \parallel \vec{w})\}$$

Plonk *preliminaries* Plonk assumes the existence of two multiplicative subgroups of $\mathbb{F}$: $\mathbb{K}_\mathsf{g}$ of order n_g and $\mathbb{K}$ of thrice that order, such that γ generates $\mathbb{K}$ and $\gamma_\mathsf{g} = \gamma^3$ generates $\mathbb{K}_\mathsf{g}$. $\mathbb{K}_\mathsf{g}$ will represent gates indices, while $\mathbb{K}$ will represent pin indices. Each gate has two input pins and one output pin.

The Plonk *index key* In Plonk, the circuit C is encoded as two polynomials. Recall (Section 2.6) that C is defined by a tuple (l_i, r_i, s_i) for each gates $i \in [n_\mathsf{g}]$. The first polynomial is the *selector polynomial*: $s(X)$, which is the unique polynomial of degree less than n_g such that for all $i \in [n_\mathsf{g}]$, $s(\gamma_\mathsf{g}^{i-1})$ is 0 when $s_i = +$, and 1 otherwise.

The second is the *wiring polynomial*. Associate with gate $i \in [n_\mathsf{g}]$: a left input pin $\gamma^{3(i-1)}$, a right input pin $\gamma^{3(i-1)+1}$, and an output pin $\gamma^{3(i-1)+2}$ (see Figure 1). The union of all gate pins is $\mathbb{K}$, and the wiring polynomial $\mathsf{w}(X)$ permutes $\mathbb{K}$. Informally, w must equate pins that are wired together in C. The permutation $\mathsf{w}(X)$ naturally defines an equivalence relation over $\mathbb{K}$; this relation equates elements in the same cycle of the unique cycle decomposition of $\mathsf{w}(X)$.

For a more formal definition of w, we define the sets pins that are wired together. For input $i \in [n_\mathsf{i}]$, define

$$\mathsf{Pins}_i^{\mathsf{In}} = \{\gamma^{3(j-1)} : j \in [n_\mathsf{g}], l_j = i\} \cup \{\gamma^{3(j-1)+1} : j \in [n_\mathsf{g}], r_j = i\}$$

to be the pins wired to input i. For gate $i \in [n_\mathsf{g}]$, define $\mathsf{Pins}_i^{\mathsf{Out}} =$

$$\{\gamma^{3(i-1)+2}\} \cup \{\gamma^{3(j-1)} : j \in [n_\mathsf{g}], l_j = i + n_\mathsf{i}\} \cup \{\gamma^{3(j-1)+1} : j \in [n_\mathsf{g}], r_j = i + n_\mathsf{i}\}$$

to be the pins wired to gate i's output. For any circuit, these sets partition $\mathbb{K}$. To be a wiring polynomial, w must induce the same partition of $\mathbb{K}$. That is, if $\overline{W}$ are the cycles of w, then the following must hold:

$$\overline{W} = \left\{\mathsf{Pins}_i^{\mathsf{In}}\right\}_{i \in [n_\mathsf{i}]} \cup \left\{\mathsf{Pins}_i^{\mathsf{Out}}\right\}_{i \in [n_\mathsf{g}]}$$

Finally, let $J = \{j_i : j_i \in \mathsf{Pins}_i^{\mathsf{In}}\}_{i=1}^{n_\mathsf{p}}$ be a set of representative input pins. Plonk's index key comprises the set J and commitments to $\mathsf{w}(X)$ and $s(X)$.

The right side of Figure 1 shows G, s, $\mathsf{Pins}^{\mathsf{In}}$, and $\mathsf{Pins}^{\mathsf{Out}}$ for our example, as well as valid choices of w and J.

H.1 Plonk Argument (Proving Protocol)

We briefly describe the Plonk argument from [20][9]. The prover interpolates a wire value polynomial $p(X) : \mathbb{K} \to \mathbb{F}$ which maps pins to the values they carry. This represents a candidate wire value assignment by the prover. The prover sends $p(X)$ to the verifier. The Plonk permutation argument (Permutation Composition over $\mathbb{K}$) is used to convince the verifier that for all $k \in \mathbb{K}$, $p(k) = p(\mathsf{w}(k))$. Informally, this argument shows that pins wired together carry the same value. The verifier must also confirm that the wire value assignment respects the gate types. The prover and verifier run a zero check for all $k \in \mathbb{K}$, $(1-s(k))[p(k)+p(\gamma \cdot k)] + s(k)p(k)p(\gamma \cdot k) - p(\gamma^2 \cdot k) = 0$. This shows that the output pin value of every gate is either the sum or product of the input pin values, depending on the gate type. Thus, $p(X)$ is a valid assignment of the wire values in the circuit. The verifier must also query $p(X)$ to check that the wire values assigned to the circuit input pins J match $\vec{x}$ and to circuit output pins (in our example, γ^8) match $\vec{y}$.

H.2 Extending Plonk

Modifying the index encoding The set of input pin representatives J may reveal relationships between the circuit inputs. To avoid this, the circuit indexing algorithm can be augmented to add prefix dummy gates which copy values from a fixed set of publicly known, prefix pins J' to the underlying input pins J. Thus, we can remove J from the PLONK index key and treat it as a fixed set of pins.

Additional properties To obtain the properties required by Theorem 2, we modify PLONK to meet our definition of zero knowledge. This can be done by performing the zero check with Zero over $\mathbb{K}$.

H.3 Proof of function relation

Functional Set for PLONK We require a functional set for the relation $R_{n_\mathsf{p},n_\mathsf{i},n_\mathsf{g},n_\mathsf{o}}$. Informally, we restrict PLONK to have no witness inputs; thus, the circuit outputs must be a function of the public inputs. Formally, we fix $n_\mathsf{i} = n_\mathsf{p}$, restricting to the relation $R_{n_\mathsf{p},n_\mathsf{p},n_\mathsf{g},n_\mathsf{o}}$. The functional set is then $\mathcal{AC}_{n_\mathsf{p},n_\mathsf{g},n_\mathsf{o}}$.

Our approach For s, it suffices to show its image is in $\{0, 1\}$ on $\mathbb{K}_\mathsf{g}$.

It is harder to prove that w is a wiring polynomial. Informally, we must convince the verifier that each equivalence class induced by the wiring polynomial w contains a gate output or a circuit input declared in J—but not both. Then, we show that these equivalence classes can be sorted, such that each gate's input classes come before that gate's output class.

In a little more detail, let $\overline{W} = \{W_i\}_{i=1}^{n_\mathsf{g}+n_\mathsf{i}}$ be the partition of $\mathbb{K}$ induced by w and let J be a set of n_p input pins, as defined above. Let $G = \{\gamma^2 p : p \in \mathbb{K}_\mathsf{g}\}$ be the subset of pins which are the output of a gate. Let $I = J \cup G$. Let α generate $\mathbb{F}^*$, with an order divisible by 2, and let $\beta = \alpha^2$. Let $N = \{\beta^i : i \in [n_\mathsf{g} + n_\mathsf{p}]\}$. Our protocol will show:

1. w is a permutation (so partition $\overline{W}$ is well-defined),
2. There exists a bijection $B : I \to \overline{W}$ such that for all $i \in I$, $i \in B(i)$.
3. The $\overline{W}$ can be topographically sorted, with inputs first.
   More precisely one can extract a surjective map $v : \mathbb{K} \to N$ such that
   (a) For all $W_i \in \overline{W}$, for all elements $w, w' \in W_i$, we have $v(w) = v(w')$.
   (b) $\mathrm{seq}_J(v) = (\beta^i : i \in [n_\mathsf{p}])$
   (c) For all $\gamma^{3(i-1)+2} \in G$ (thus, $i \in [n_\mathsf{g}]$), $v(\gamma^{3(i-1)+2}) = \beta^{i+n_\mathsf{p}}$.
   (d) For each gate i, the discrete log of the image of the left and right input pins is less than that of the output pin. More formally, $\log_\beta(v(\gamma^{3(i-1)})) < \log_\beta(v(\gamma^{3(i-1)+2}))$ and $\log_\beta(v(\gamma^{3(i-1)+1})) < \log_\beta(v(\gamma^{3(i-1)+2}))$.
4. $s(\mathbb{K}_\mathsf{g}) \subseteq \{0, 1\}$.

Soundness and extractability

Claim. If the above conditions hold, one can extract from w, s, and v a sequence of n_{g} tuples $C = ((l_i, r_i, s_i))_{i=1}^{n_{\mathsf{g}}}$ such that C is a valid arithmetic circuit on n_{p} inputs, and w is a wiring permutation for it.

Proof. First, we extract a candidate n_{p}-input, n_{g}-gate circuit C from the polynomials s and v. Then, we show that w is a wiring polynomial for the candidate circuit C. Define C as follows. For gate $i \in [n_{\mathsf{g}}]$, define that gate by[9]

$$\left(l_i = \log_\beta\left(v(\gamma^{3(i-1)})\right),\ r_i = \log_\beta\left(v(\gamma^{3(i-1)+1})\right),\ s_i = s\left(\gamma_{\mathsf{g}}^i\right)\right)$$

We argue that C is a valid arithmetic circuit with n_{p} inputs. By construction, C has n_{g} gates and n_{p} inputs. It suffices to show that for each $i \in [n_{\mathsf{g}}]$, l_i, r_i are strictly less than $i + n_{\mathsf{p}}$ and $s_i \in \{0, 1\}$. The former is implied by conditions 3(c-d). The latter is implied by condition 4. Thus, C is valid.

We will show that w is a wiring polynomial for C. It suffices to show that $\overline{W}$ (the partition induced by w) and $\{\mathsf{Pins}_i^{\mathsf{In}}\}_{i\in[n_{\mathsf{i}}]} \cup \{\mathsf{Pins}_i^{\mathsf{Out}}\}_{i\in[n_{\mathsf{g}}]}$ are equivalent.

By definition of $\mathsf{Pins}_i^{\mathsf{In}}$, we know that $\{\mathsf{Pins}_i^{\mathsf{In}}\}_{i\in[n_{\mathsf{p}}]} = \{\{\gamma^{3(j-1)} : j \in [n_{\mathsf{g}}], l_j = i\} \cup \{\gamma^{3(j-1)+1} : j \in [n_{\mathsf{g}}], r_j = i\}\}_{i\in[n_{\mathsf{p}}]}$. By the construction of C and 3(b-c), this implies $\{\{\mathsf{Pins}_i^{\mathsf{In}}\}\}_{i\in[n_{\mathsf{p}}]} = \{\{h \in \mathbb{K} : v(h) = \beta^i\}\}_{i\in[n_{\mathsf{p}}]}$. Similarly, it can be shown that $\{\{\mathsf{Pins}_i^{\mathsf{Out}}\}\}_{i\in[n_{\mathsf{g}}]} = \{\{h \in \mathbb{K} : v(h) = \beta^{i+n_{\mathsf{p}}}\}\}_{i\in[n_{\mathsf{g}}]}$. Thus, by the definition of N, we have $\{\mathsf{Pins}_i^{\mathsf{In}}\}_{i\in[n_{\mathsf{i}}]} \cup \{\mathsf{Pins}_i^{\mathsf{Out}}\}_{i\in[n_{\mathsf{g}}]} = \{\{h \in \mathbb{K} : v(h) = b\}\}_{b\in N}$. Let us call this set $\overline{V}$. Since v is a surjective map, each member of $\overline{V}$ is non-empty, so $\overline{V}$ is a partition of $\mathbb{K}$. By condition 3(a), we have for all $W_i \in \overline{W}$, there exists $V \in \overline{V}$ such that $W_i \subseteq V$; that is, that $\overline{W}$ refines $\overline{V}$. By condition 3(b-c), we know that v maps elements of I to distinct elements in N. Therefore, there cannot exist $i_1, i_2 \in I$ such that $v(i_1) = v(i_2)$. Thus, by condition 2, there cannot exist $W_1 \neq W_2$ such that $W_1, W_2 \subseteq V$ for some $V \in \overline{V}$. By the partition lemma (appendix G) this fact and the fact that $\overline{W}$ refines $\overline{V}$ implies that $\overline{W} = \overline{V}$. Thus, w is a valid wiring polynomial for C.

Finally, condition 2 and the definition of I guarantee that J represents all n_{p} input wires.

The proof of function relation, in detail Our proof of function relation simply checks the conditions listed above. Condition 1 is checked with Permutation on $\mathbb{K}$. Condition 2 is checked with Representative Check and Topological Sort. Condition 3 is checked with Topological Sort. See Plonk Proof-of-Function Relation for the full protocol description.

Protocol 14 (Permutation on $\mathbb{K}$)
$\mathcal{R}_{\text{perm}} = \left\{((\mathsf{w} \in \mathbb{F}^{(<B)}[X]), \bot, \bot) : \mathsf{w}(\mathbb{K}) = \mathbb{K}\right\}$

[9]The extractor can efficiently compute these discrete logarithms because N is small.

1. Use Multiset Equality over $\mathbb{K}$ on $\mathsf{w}, g(X) = X$ to show $\{\!\{\mathsf{w}(k) : k \in \mathbb{K}\}\!\} = \{\!\{k : k \in \mathbb{K}\}\!\}$.

Theorem 18. *Protocol 14 (Permutation on $\mathbb{K}$) is a secure* polyIOP *(Definition 4) for relation $\mathcal{R}_{perm}$.*

Proof. *Completeness*, *Soundness*, and *Zero Knowledge* are immediate.

Protocol 15 (Representative Check)
Let $G = \{\gamma^2 p : p \in \mathbb{K}_{\mathsf{g}}\}$, let $J \subseteq \mathbb{K}$, and let $\overline{W}$ be the partition of $\mathbb{K}$ induced by the cycles of w, where w is a permutation polynomial over $\mathbb{K}$.

$$\mathcal{R}_{\text{rep}} = \{((\mathsf{w} \in \mathbb{F}^{(<B)}[X]), (J, G), \perp) : \mathsf{w}(\mathbb{K}) = \mathbb{K} \text{ and } \forall\, W \in \overline{W},\ W \cap (J \cup G) \neq \emptyset\}$$

In other words, $J \cup G$ intersects every cycle of w over $\mathbb{K}$.

1. $\mathcal{P}$ interpolates and sends a polynomial $f \in \mathbb{F}^{(<B)}[X]$ that satisfies
$$\begin{cases} f(k) = 1 & \text{for } k \in I = J \cup G, \text{ and} \\ f(k) = \alpha \cdot f\big(\mathsf{w}^{-1}(k)\big) & \text{for } k \in \mathbb{K} \setminus I. \end{cases} \tag{7}$$
Such an f can be constructed whenever the set I intersects every cycle in w (i.e., I intersects every set in $\overline{W}$).
2. $\mathcal{P}$ interpolates and sends $p \in \mathbb{F}^{(<B)}[X]$ where $\forall k \in \mathbb{K}$: $f(k) = p(\mathsf{w}(k))$.
3. Use Permutation on $\mathbb{K}$ to show $\mathsf{w}(\mathbb{K}) = \mathbb{K}$.
4. Use Permutation Composition over $\mathbb{K}$ to show $f(X) = p(\mathsf{w}(X))$ over $\mathbb{K}$.
5. Use Zero over $\mathbb{K}$ to show that for all $k \in \mathbb{K}$,
$$\big(f(k) - \alpha \cdot p(k)\big) \cdot z_G(k) \cdot z_J(k) = 0, \tag{8}$$
where $z_G(X) = (X^{|\mathbb{K}_{\mathsf{g}}|} - \gamma^{2 \cdot |\mathbb{K}_{\mathsf{g}}|})$ and $z_J(X) = \prod_{j \in J}(X - j)$ are the vanishing polynomials on G and J respectively.
6. Use Nonzero over $\mathbb{K}$ to show that for all $k \in \mathbb{K}$, $f(k) \neq 0$.

Theorem 19. *Protocol 15 (Representative Check) is a secure* polyIOP *(Definition 4) for relation $\mathcal{R}_{rep}$.*

Proof. Since there is no witness, it suffices to show completeness, soundness, and zero-knowledge.

Completeness We need to show that (8) holds. It suffices to show that for all $k \in \mathbb{K} \setminus I$ we have $f(k) - \alpha \cdot p(k) = 0$. Since $f(k) = p(\mathsf{w}(k))$ over $\mathbb{K}$, we have $p(k) = f(\mathsf{w}^{-1}(k))$. Thus, it suffices to show that $f(X) - \alpha \cdot f(\mathsf{w}^{-1}(X)) = 0$ for $k \in \mathbb{K} \setminus I$. Indeed this holds by definition of f in (7).

Soundness Assume for the sake of contradiction that there exists a $W_i \in \overline{W}$ such $I \cap W_i = \emptyset$. Consider an arbitrary element $w \in W_i$. From the soundness of Zero over $\mathbb{K}$, we know that $f(w) = \alpha \cdot p(w)$ (since the vanishing polynomials for $I = J \cup G$ must be nonzero). By the soundness of Permutation Composition over $\mathbb{K}$, we have $f(X) = p(\mathsf{w}(X))$ over $\mathbb{K}$, which implies $p(X) = f(\mathsf{w}^{-1}(X))$ over $\mathbb{K}$. Thus, we have $f(w) = \alpha \cdot f(\mathsf{w}^{-1}(w))$. Since we considered an arbitrary $w \in W_i$, for all $w \in W_i$, we have $f(w) = \alpha \cdot f(\mathsf{w}^{-1}(w))$.

Pick an arbitrary $w_0 \in W_i$. Let $(w_0, w_1, ..., w_j)$ for $j = |W_i| - 1$ represent the cycle that contains w_0. Since α generates $\mathbb{F}^*$ and Nonzero over $\mathbb{K}$ is sound, we have $f(w_0) = \alpha^a$ for some $a < |\mathbb{F}^*|$. Inductively, $f(w_i) = \alpha^{a+i}$ for all $i \leq j$. Since $f(w) = \alpha \cdot f(\mathsf{w}^{-1}(w))$, we must have $f(w_0) = \alpha \cdot f(w_j)$ implies $\alpha^a = \alpha^{a+j+1}$. However, this implies $\alpha^{j+1} = \alpha^{|W_i|} = 1$, but since $|W_i| \ll |\mathbb{F}^*|$, this is a contradiction. Therefore, we must have for all $W_i \in \overline{W}$, there exists an $i \in I$ such that $i \in W_i$.

Zero Knowledge is immediate.

Checking topological order Informally, we must check that the wires encoded by w can be ordered such that each gate's inputs come before its outputs. We show this with a mapping $v : \mathbb{K} \to N = \{\beta^1, \dots, \beta^{n_\mathsf{g}+n_\mathsf{p}}\}$. The mapping must send the representation of input $i \in [n_\mathsf{p}]$ (from J) to β^i and the output pin for gate $i \in [n_\mathsf{g}]$ to β^{i+n_p}. Furthermore, it must assign the same value to any pins that w connects (i.e., has in the same cycle). Finally, it must assign a lesser β power to each gate's input pins than it assigns to the output pin. The relation for Topological Sort describes these conditions in detail.

Protocol 16 (Topological sort)
Let $G = \{\gamma^2 p : p \in \mathbb{K}_\mathsf{g}\}$, let $J \subseteq \mathbb{K}$, and let $\overline{W}$ be the partition of $\mathbb{K}$ induced by the cycles of w, where w is a permutation polynomial over $\mathbb{K}$.

$$\begin{aligned}\mathcal{R}_{\text{TS}} = \Big\{&((v, \mathsf{w} \in \mathbb{F}^{(<B)}[X]), (J, G), \perp) : \text{seq}_J(v) = (\beta^i : i \in [n_\mathsf{p}]) \wedge \\ &\forall\, \gamma^{3(i-1)+2} \in G,\ v(\gamma^{3(i-1)+2}) = \beta^{i+n_\mathsf{p}} \wedge \\ &\forall\, W_i \in \overline{W}, \forall\, w, w' \in W_i, v(w) = v(w') \ \wedge\ v(\mathbb{K}) = N \wedge \\ &\forall\, i \in [n_\mathsf{g}], \log_\beta(v(\gamma^{3(i-1)})) < \log_\beta(v(\gamma^{3(i-1)+2})) \wedge \\ &\log_\beta(v(\gamma^{3(i-1)+1})) < \log_\beta(v(\gamma^{3(i-1)+2}))\Big\}\end{aligned}$$

---

1. $\mathcal{P}$ and $\mathcal{V}$ interpolate $u(X) \in \mathbb{F}^{(<B)}[X]$ such that $u(j_i) = \beta^i$ for $i \in [n_\mathsf{p}]$, where j_i is the ith element of J, where J has an agreed upon ordering.
2. Use Zero over J to check $v(X) = u(X)$ over J.
3. Use Geometric Sequence Test $\text{seq}_G(v) = (\beta^{i+n_\mathsf{p}} : i \in [n_\mathsf{g}])$
4. Use Permutation Composition over $\mathbb{K}$ to show $v(\mathsf{w}(X)) = v(X)$ on $\mathbb{K}$.
5. Use Expanded Discrete-log Comparison with parameters $(\alpha, n_\mathsf{g}+n_\mathsf{p}+1)$ to show $\log_\beta(v(X)) < \log_\beta(v(\gamma^2 X))$ over $\mathbb{K}_\mathsf{g}$.

6. Use Expanded Discrete-log Comparison with parameters $(\alpha, n_{\mathsf{g}}+n_{\mathsf{p}}+1)$ to show $\log_\beta(v(\gamma X)) < \log_\beta(v(\gamma^2 X))$ over $\mathbb{K}_{\mathsf{g}}$.
7. Use Nonzero over $\mathbb{K}$ to check $v(X) - 1$ is nonzero over $\mathbb{K}$.

Theorem 20. *Protocol 16 (Topological Sort) is a secure* polyIOP *(Definition 4) for relation $\mathcal{R}_{TS}$.*

Proof. Since there is no witness, it suffices to show completeness, soundness, and zero-knowledge.

Completeness By construction, steps 2, 3, and 4 must succeed. Since $v(w) = v(w')$. for all $w, w' \in W_i$ for all $W_i \in \overline{W}$, step 5 succeeds. Since, for each gate i, $v(\gamma^{3(i-1)}) < v(\gamma^{3(i-1)+2})$ and $v(\gamma^{3(i-1)+1}) < v(\gamma^{3(i-1)+2})$, the comparisons succeed in steps 5 and 6.

Soundness Step 2 directly implies the first property in $\mathcal{R}(J, G)$. From the soundness of Geometric Sequence Test, we have the second property. Step 4 directly implies[10] that $v(w) = v(w')$ for all $w, w' \in W_i$ and for all $W_i \in \overline{W}$. For any $\gamma^{3(i-1)} \in \mathbb{K}_{\mathsf{g}}$, step 5 shows that $v(\gamma^{3(i-1)}) < v(\gamma^{3(i-1)+2})$ and step 6 shows that $v(\gamma^{3(i-1)+1}) < v(\gamma^{3(i-1)+2})$. From the soundness of Expanded Discrete-log Comparison, we have $v(\mathbb{K}_{\mathsf{g}}), v(\gamma\mathbb{K}_{\mathsf{g}}), v(\gamma^2\mathbb{K}_{\mathsf{g}}) \subseteq N \cup \{1\}$. Because these are the cosets of $\mathbb{K}_{\mathsf{g}}$ in $\mathbb{K}$ and by step 7 (Nonzero over $\mathbb{K}$), we have $v(\mathbb{K}) \subseteq N$. From steps 2, 3 (Geometric Sequence Test), we must have $v(\mathbb{K}) = N$.

Zero Knowledge is immediate.

Protocol 17 (Plonk proof-of-function relation)
$R_{\mathcal{AC}} = \{((\mathsf{w}, s \in \mathbb{F}^{(<B)}[X]), (n_{\mathsf{p}}, n_{\mathsf{g}}, n_{\mathsf{o}}), C \in \mathcal{AC}_{n_{\mathsf{p}},n_{\mathsf{g}},n_{\mathsf{o}}}) : \mathsf{w} \text{ and } s \text{ encode } C\}$

1. $\mathcal{P}$ interpolates and sends $v \in \mathbb{F}^{(<B)}[X]$ such that for $i \in [n_{\mathsf{g}}]$,
   - $v(\gamma^{3(i-1)}) \mapsto \beta^{l_i}$
   - $v(\gamma^{3(i-1)+1}) \mapsto \beta^{r_i}$
   - $v(\gamma^{3(i-1)+2}) \mapsto \beta^{i+n_{\mathsf{p}}}$
2. Use Zero over $\mathbb{K}_{\mathsf{g}}$ to show $s(X)s(X) = s(X)$ over $\mathbb{K}_{\mathsf{g}}$.
3. Invoke Permutation on $\mathbb{K}$ on w, Representative Check on w, Topological Sort on w and v.

Theorem 21. *Protocol 17 (Plonk proof-of-function relation) is a secure* polyIOP *(Definition 4) for relation $R_{\mathcal{AC}}$. Thus is a secure PFR for the* Extended Plonk.

Proof. Completeness, soundness, and zero-knowledge follow from the same properties of the subprotocols.

[10] A similar argument was made for the wire value assignment in Plonk [20].

Knowledge Soundness The extractor $\mathcal{E}$ takes as argument the oracles w and s. We show that the properties required by our previous circuit extractor are guaranteed:

1. Permutation on $\mathbb{K}$ shows that w is a permutation.
2. Representative Check shows
   - For all $W_i \in \overline{W}$, there exists $i \in I$ such that $i \in W_i$.
   - $|\overline{W}| \leq |I|$ as a corollary.

   Topological Sort shows from the third property of $\mathcal{R}(J, G)$ that $|v(\mathbb{K})| \leq |\overline{W}|$. The first and second properties imply $|I| \leq |v(\mathbb{K})|$. Thus, $|I| \leq |\overline{W}|$. Thus, from the corollary above, we must have $|I| = |\overline{W}|$. Therefore, there exists a bijection $B : I \to \overline{W}$ such that for all $i \in I$, $i \in B(i)$.
3. Topological Sort shows $\overline{W}$ can be topologically sorted.
4. Zero over $\mathbb{K}$ shows $s(\mathbb{K}_{\mathsf{g}}) \subseteq \{0, 1\}$, since the only roots of $X^2 - X = (X-1) \cdot X$ are zero and one.

Thus, by our previous argument, a circuit encoded by w and s can be extracted.
□