Efficient Functional Commitments:
How to Commit to Private Functions

Dan Boneh, Wilson Nguyen, and Alex Ozdemir

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

Abstract. We construct efficient functional commitments for all bounded size arithmetic circuits. A (function hiding) functional commitment scheme allows 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. Thus, functional commitments allow the operator of a secret process to prove that the process is being applied uniformly to everyone. Possible applications include bail decisions, credit scores, online ranking algorithms, and proprietary software-as-a-service.

To build functional commitments, we introduce a new type of protocol: a *proof of function relation* (PFR) to show that a committed relation is a function. We show that combining a suitable preprocessing zk-SNARK 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. These constructions build on *polynomial commitments* (a special case of functional commitments), so our work shows that polynomial commitments are "complete" for functional commitments.

1 Introduction

A *functional encryption* scheme [15, 43] is an encryption scheme where every decryption key sk_f has an associated function f. If ct is the encryption of some plaintext m, then a functional decryption key sk_f applied to ct reveals $f(m)$ and nothing else. Functional encryption is an active area of research and has many applications [2, 11, 26, 32, 35, 45].

Functional commitments [36, 37, 42] are a natural analogue in the context of cryptographic commitments. We are interested in the following notion, which we call function-hiding functional commitments: a *committer* commits to a secret function $f \in \mathcal{F}$ 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, y), the committer can prove to a verifier that the committed function f satisfies $y = f(x)$, without revealing anything else about f.

A polynomial commitment scheme (PCS) [33] is an important special case of functional commitments. Here the committer commits to a polynomial of bounded degree in $\mathbb{F}[X]$. It can later open the polynomial at any public point in $\mathbb{F}$. In this paper we generalize polynomial commitments, and construct efficient functional commitments for the set of all functions that can be expressed as an arithmetic circuit of bounded size.

In more detail, a functional commitment scheme is a triple $(\mathsf{Setup}, \mathsf{Commit}, \mathsf{Eval})$. $\mathsf{Setup}(\lambda)$ is a randomized algorithm that outputs some public parameters pp. $\mathsf{Commit}(\mathsf{pp}, f, r)$ is a deterministic algorithm that takes as input the description of a function $f \in \mathcal{F}$ 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 the function f. In addition, the evaluation protocol must satisfy an important property called *evaluation binding*: it should be infeasible for a malicious prover to 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 $\mathcal{V}_{\mathsf{E}}(\mathsf{pp}, c, x, y')$. We define these properties formally in Section 3.

Evaluation binding ensures that c is a commitment to a function: there is a unique output for every input. Functional commitments enable an organization to commit to a secret function, and the public is

assured that the organization is bound to that function. For example, in the United States, a credit bureau can commit to the secret function it uses to compute a person's credit score. Then, given a person's financial records, say Bob, the credit bureau can compute Bob's credit score, and prove to Bob that the score was computed correctly. Here Bob plays the role of the verifier. Evaluation binding is crucial: it ensures that the same function is applied to everyone. If needed, the function itself can be audited by an auditor who is trusted to examine its inner workings.

Other applications of functional commitments may include 1) *Ranking algorithms*: ranked parties want to know that the same ranking procedure is applied to everyone uniformly. 2) *Software-as-a-Service*: customers want to verify that they are receiving the service they are paying for. For example, consider a company 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 company could save money by using B and lie that they are using A. By committing to classifier A, the company can prove it is providing the same service to all of its customers. A trusted auditor could verify that the commitment is indeed a commitment to A. 3) *Price discrimination*: a company could commit to a pricing function that takes a product detail as input and outputs a price. It could then prove to every customer that the price being charged is the same for everyone.

Functional commitments are related to the notion of fairness for secret processes. A functional commitment allows the operator of a secret process to prove that the process is being applied uniformly to everyone. However, we note that while uniform application is necessary for fairness, it is not sufficient. In fact, since "fairness" is a social construct, formalizing it is an interesting problem that has been the subject of a great deal of work: [5] and [34] survey.

Constructing a functional commitment scheme: the challenge. A functional commitment scheme for bounded size arithmetic circuits can be built from a standard succinct commitment scheme and a general zero knowledge proof system, using universal circuits. We discuss this further in related work below.

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 [21], Plonk [25], or many others [1, 22, 30, 31, 38, 41, 44, 46].

Informally, a preprocessing zk-SNARK operates in two phases (following a setup step). For a function $f : \mathcal{X} \to \mathcal{Y}$ define the relation

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

Let i be a binary string, called an index, that describes the relation R_f (e.g., a description of an arithmetic circuit for f). The first phase of a preprocessing zk-SNARK is a deterministic *indexing 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. Then, in the second phase, called the online phase, the zk-SNARK prover takes as input i and a pair $(x, y) \in R_f$, and outputs a succinct non-interactive zero knowledge argument π (in the random oracle model) that $(x, y) \in R_f$. The verifier takes as input ik, (x, y), and the proof π, and outputs accept or reject. In Plonk and Marlin, the size of the indexing key and the size of the argument π depend only on the security parameter λ. The time to verify the argument π depends logarithmically on the complexity of f and linearly on the length of (ik, x, y). The time to generate the argument 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 indexing 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, and moreover, a zk-SNARK proof is an argument of knowledge for the committed function.

Evaluation binding is the bigger problem: there is no guarantee that ik represents a function. First, ik might not encode a relation R at all. Second, since zk-SNARKs support *relations*—not just functions—R might be a relation that is not a function; we explain this with two examples.

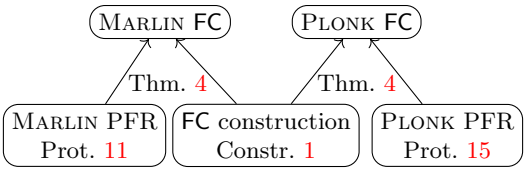


Fig. 1: Combining our proofs of function relation (PFR) with a zk-SNARK in our generic Construction 1 yields two functional commitment schemes (FC).

The Plonk indexing algorithm takes as input the description of an arithmetic circuit of bounded size and outputs an index key ik. The problem is that the Plonk indexing algorithm will output a valid index key even for a malformed arithmetic circuit: the circuit may contain gates wired together in a cycle, and some gates may take hidden inputs (witness inputs). For example, a malicious committer can commit to one of the following arithmetic circuits and claim that the committed circuit is well formed and takes a single input x:

The verifier learns nothing about the committed circuit from the commitment string. It thinks that the circuit on the left has only one input x. However the committer can set the hidden input w to any value it wants and convince the verifier that $f(x)$ is any value of the committer's choosing. For the circuit on the right, when $x = 0$, any output y corresponds to a valid addition. Thus, the committer can then convince the verifier that $f(0)$ is any value of its choosing.

A similar problem happens with Marlin. The Marlin indexing algorithm takes as input an R1CS program, namely three matrices $A, B, C \in \mathbb{F}^{n \times n}$, as explained in Section 6. It outputs an index key ik. As in the case of Plonk, there is no guarantee that ik is a commitment to an R1CS program where the output is uniquely determined by the input x. The output may depend on hidden inputs that are controlled by the adversary.

Proof of function relation (PFR). The Plonk and Marlin examples above show that to build a functional commitment scheme from a preprocessing zk-SNARK we need the zk-SNARK to satisfy a number of additional properties, described in Section 4. A key property is that the prover should be able to efficiently produce a succinct zero knowledge proof that an index key ik is a commitment to a function. We call this a *proof of function relation* or PFR. In Section 4 we show that a preprocessing zk-SNARK that has these additional properties can be used to construct a secure functional commitment scheme, as illustrated in Figure 1.

The challenge is to design an efficient zero knowledge proof of function relation (PFR) for commonly used preprocessing zk-SNARKs.

- For Plonk, we design an efficient zk-SNARK that a Plonk index key is a commitment to an arithmetic circuit whose graph is acyclic and whose inputs are explicitly declared. This is sufficient to ensure that every input has a unique output. See Section 7.
- For Marlin, we design an efficient zk-SNARK that a Marlin index key is a commitment to an R1CS program $A, B, C \in \mathbb{F}^{n \times n}$ where A and B are strictly lower triangular and C is diagonal. 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 6.

Designing an efficient zk-SNARK for these properties requires new algebraic tools and new sub-protocols to prove properties of committed polynomials. We develop these sub-protocols in Section 5. Many of them are of independent interest.

An overview. In their simplest form, a Plonk indexing key is a commitment to two polynomials, and a Marlin indexing key is a commitment to nine polynomials. We review what these polynomials are in Sections 6 and 7. Proving that these indexing keys are a commitment to a well formed function requires proofs that the committed polynomials satisfy certain complex algebraic properties. We give three examples.

- **Discrete log comparison**: (Protocol 8) 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

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

- **Representative check**: (Protocol 13) 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, $\mathsf{w}(\mathbb{K}) = \mathbb{K}$. 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. Thus, to prove that the committed circuit has no hidden inputs we design a novel zk-SNARK to prove that a certain public set $I \subseteq \mathbb{K}$ intersects every cycle of the committed w. The set I corresponds to the declared input wires to the circuit and the output wires from every gate. This is sufficient to prove in zero knowledge that the committed circuit has no hidden inputs.
- **Topological sort:** (Protocol 14) To prove that the committed arithmetic circuit in Plonk is acyclic, we use the *discrete log comparison* protocol above to design a protocol to prove that there is a topological ordering of the wires in the committed circuit.

We use these to prove that a Plonk index key is a commitment to a well formed circuit, and that a Marlin index key is a commitment to an R1CS program A, B, C where A and B are strictly lower triangular and C is diagonal.

Future work. Our work motivates the design of efficient PFRs for other popular zk-SNARKs such as Spartan [44], Fractal [22], Ligero [1], Libra [46], 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 is sufficient 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 in the underlying polynomial scheme.*

1.1 Additional related work

Previous works on functional commitments [36, 37, 42] consider a dual notion to ours: 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. We call this *input-hiding functional commitments*. 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 $U(f, x)$, where $U(f, x) = f(x)$. 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 general zk-SNARK, as observed in [36]. 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.

Input-hiding functional commitments were implicitly constructed by Gorbunov, Vaikuntanathan, and Wichs [29], although commitments produced by their commitment scheme are not succinct. The term *functional commitment* was introduced by Libert, Ramanna and Yung [36] and further developed in [37, 42]. The focus of these works is on efficient *input-hiding* functional commitments under falsifiable assumptions (zk-SNARKs require non falsifiable assumptions). Libert et al. [36] and Lipmaa and Pavlyk [37] give an input-hiding functional commitment scheme for the family of linear (or linearizable) functions. Peikert, Pepin, and Sharp [42] give a lattice construction for bounded depth boolean circuits, meaning that the size of commitments and proofs grow polynomially in the depth of the circuit.

Special cases of functional commitments. Several cryptographic primitives can be viewed as special cases of functional commitments. A verifiable random function (VRF) [40] 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. One difference is that a VRF evaluation proof need not be an argument of knowledge for the key. Other examples include vector commitments [13, 17, 18, 28, 42], accumulators [10, 13, 16] and zero knowledge sets [19, 20, 39]. A vector commitment can be viewed as a function-hiding functional commitment where the function is described as a truth table.

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 [47] 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 [27] can extend this to multiple evaluations. In fact, [27] 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 relies on fairly heavy cryptographic primitives.

2 Preliminaries

2.1 Mathematical notation

Let $[n]$ for $n \in \mathbb{N}_{>0}$ represent the sequence $1, 2, ..., n$. We 1-index sequences and matrices throughout. Let $\{\!\{\cdot\}\!\}$ denote a multiset. Thus, $\{\!\{1,1\}\!\} \neq \{\!\{1\}\!\}$. Let $\|$ be 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 $\mathsf{negl}(n)$.

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 $\left\{\gamma^i\right\}_{i\in\mathbb{N}}$. Let $\mathbb{F}^{(<d)}[X]$ denote the set of polynomials in formal variable X with coefficients from $\mathbb{F}$ with degree less than d. Fix a canonical order of $\mathbb{F}$. For fields of prime order ($\mathbb{F}_p$), this can be the order of their natural number representatives.

For $\mathbb{S} \subseteq \mathbb{F}$ and a function $f : \mathbb{F} \to \mathbb{F}$, let $g = \mathsf{LDE}_{\mathbb{S}}(f)$ be the unique polynomial in $\mathbb{F}^{(<|\mathbb{S}|)}[X]$ such that $g(s) = f(s)$ for all $s \in \mathbb{S}$. 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 $\mathbb{S}$ comes with a canonical ordering, then $\mathsf{seq}_{\mathbb{S}}(f)$ denotes the (ordered) *sequence* $(f(s) : s \in \mathbb{S})$. Consider a nonempty set $\mathbb{S}$. A *partition* $\overline{S}$ of $\mathbb{S}$ is a set of nonempty subsets $S \in \overline{S}$ of $\mathbb{S}$ such that $\mathbb{S}$ is the disjoint union of $S \in \overline{S}$. Given two partitions $\overline{A}, \overline{B}$ of $\mathbb{S}$, $\overline{A}$ is a *refinement* of $\overline{B}$ if for all $A \in \overline{A}$, there exists $B \in \overline{B}$ such that $A \subseteq B$.

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, let $\{D\} = \{D'\}$ denote that the distributions are the same.

2.2 Commitment schemes

A commitment scheme for messages $x \in \mathcal{X}$ is a tuple of algorithms $(\mathsf{Setup}, \mathsf{Commit})$ 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*. We formally define these as follows.

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

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

– **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 Interactive arguments

We use the notation $\mathsf{NAME}(\mathcal{P}(a), \mathcal{V}(b)) \to (x, y)$ to denote an interactive protocol called NAME. $\mathcal{P}$ takes input a and receives output x. $\mathcal{V}$ takes input b and receives output y. For interactive machines $\mathcal{P}$ and $\mathcal{V}$ we use $\langle \mathcal{P}(a), \mathcal{V}(b) \rangle$ to denote the random variable that is the output of their interaction.

An interactive argument Π 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. *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. *An interactive argument $(\mathcal{P}, \mathcal{V})$ for a relation $R \subseteq \mathcal{X} \times \mathcal{W}$ is an **Argument of Knowledge** (or **knowledge-sound**), if for all pairs of PPT adversaries $(\mathcal{P}_1, \mathcal{P}_2)$, there exists a PPT extractor Ext such that*

$$Pr\left[\begin{array}{c} \langle \mathcal{P}_2(\mathsf{st}), \mathcal{V}(x) \rangle = 1 \\ \Downarrow \\ (x, w) \in R \end{array} \middle| \begin{array}{l} (x, \mathsf{st}) \leftarrow \mathcal{P}_1 \\ w \leftarrow \mathsf{Ext}^{\mathcal{P}_2(\mathsf{st})}(x)^{1} \end{array}\right] \geq 1 - \mathsf{negl}(\lambda)$$

where $\mathsf{Ext}^{\mathcal{P}_2(\mathsf{st})}$ denotes that Ext has oracle access to the "interactive function" $\mathcal{P}_2(\mathsf{st})$ (see [6]).

Definition 3. *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.

Composition. Let $(\mathcal{P}, \mathcal{V})$ be an HVZK interactive argument for relation R. Define $R^t \subseteq \mathcal{X}^t \times \mathcal{W}^t$ to be the following relation: $(\vec{x}, \vec{w}) \in R^t$ if for all $i \in [t]$, $(\vec{x}_i, \vec{w}_i) \in R$. We define the interactive argument $(\mathcal{P}^t, \mathcal{V}^t)$ to be the parallel composition of $(\mathcal{P}, \mathcal{V})$ on $(\vec{x}_i, \vec{w}_i) \in R$ for all $i \in [t]$.

Theorem 2. *Let $(\mathcal{P}, \mathcal{V})$ be an HVZK interactive argument for relation R. The interactive argument $(\mathcal{P}^t, \mathcal{V}^t)$ for relation R^t is also HVZK.*

Proof Sketch Let Sim be the simulator for $(\mathcal{P}, \mathcal{V})$. We can construct a simulator Sim' for $(\mathcal{P}^t, \mathcal{V}^t)$ by running Sim on $\vec{x}_i$ for all $i \in [t]$. A standard hybrid argument shows that Sim' is a valid simulator.

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

2.4 Polynomial interactive oracle proofs

Polynomial interactive oracle proofs [21] (polyIOPs) refine interactive oracle proofs [9] (IOPs), themselves generalizations of probabilistically checkable proofs [3] (PCPs).[2] A polyIOP is an interactive proof between a prover $\mathcal{P}$ and verifier $\mathcal{V}$, where $\mathcal{P}$ sends *polynomials* that $\mathcal{V}$ can subsequently access via evaluation queries.

A *polynomial commitment scheme* [33] (PCS) enables a polyIOP to be compiled into a standard protocol [21, 38]. With a PCS, $\mathcal{P}$ can *commit* to each polynomial it sends the verifier. 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. The derived protocol has various security properties, which we discuss below. See [21] for a complete description.

Let Π be a polyIOP for a witness relation and let $\tilde{\Pi}$ be the compiled protocol. If the PCS has *extractability* (committed polynomials can be extracted from evaluation proofs), and the witness is extractable from Π's polynomials, then $\tilde{\Pi}$ is an argument of knowledge for the witness. To obtain a zero-knowledge protocol, one can exploit *bounded independence* introduced in [7] and formalized by [8, 21]. The idea is to start with a polyIOP in which the secret witness is encoded as the evaluations of a set of polynomials $f^{(1)}, \ldots, f^{(m)}$ over $K \subset \mathbb{F}$ such that $|\mathbb{F} \setminus K|$ is super-poly. Let b be an upper bound on the number of queries that the polyIOP makes to any polynomial. Protocol 1 performs *re-randomization*, and yields a polynomial $f'^{(i)}$ that agrees with $f^{(i)}$ on K, but is uniformly random for any b queries on $\mathbb{F} \setminus K$. By *re-randomizing* each $f^{(i)}$ before running the polyIOP, we can derive a zero knowledge protocol.

Protocol 1 requires that polynomial oracles are additive (i.e. given polynomial oracles f and g, an oracle for $h = f + g$ can be derived without querying f or g). This property allows $\mathcal{V}$ to derive f' without ever querying f. Many PCSs have this property, including the one in appendix D.

Protocol 1 (Randomized Low Degree Extension)
Given: A query bound b, subset $\mathbb{K} \subset \mathbb{F}$, and a polynomial $f \in \mathbb{F}^{(<|\mathbb{K}|)}[X]$.

Outputs: A polynomial $f' \in \mathbb{F}^{(<|\mathbb{K}|+\mathsf{b}+1)}[X]$ that agrees with f on $\mathbb{K}$ and whose evaluations on any size-b subset S of $\mathbb{F} \setminus \mathbb{K}$ are uniformly random and independent from $f(\mathbb{K})$.

---

1. $\mathcal{P}$ samples and sends $r \xleftarrow{\$} \mathbb{F}^{(<\mathsf{b}+1)}[X]$.
2. $\mathcal{P}$ computes and sends $m = z_{\mathbb{K}} \cdot r$.
3. $\mathcal{V}$ checks $m(x) = z_{\mathbb{K}}(x) r(x)$ at $x \xleftarrow{\$} \mathbb{F} \setminus \mathbb{K}$.
4. $\mathcal{P}$ computes $f' = m + f$. $\mathcal{V}$ derives an oracle for f' from the additive property of oracles for f and m.

Theorem 3 ((informal) polyIOP to HVZK Compilation). *Let Π be a public coin* polyIOP *with input polynomials $f_1, \ldots, f_m \in \mathbb{F}^{(<|\mathbb{K}|)}[X]$. Furthermore, let it be a sound and complete interactive argument for a witness relation R whose witness is encoded as the evaluations of $f_1, \ldots, f_m$ over $\mathbb{K}$. Finally, let Π make at most b queries to any single polynomial and queries f_i only randomly on $\mathbb{F} \setminus \mathbb{K}$. Let Π' denote the* polyIOP *that re-randomizes each f_i before running Π. Let Π'' denote the standard protocol that is the compilation of Π' using a hiding, binding, additive, extractable, and ZK-evaluation PCS. Then, Π'' is public coin, complete, knowledge-sound, and HVZK.*

Proof Sketch [21] formalizes and proves a very similar theorem with a key difference. They prove a variant of HVZK that requires a trap-door to simulate the protocol transcript. When we replace their PCS with one that has a ZK evaluation protocol, their simulator no longer requires a trap door and can instead invoke the ZK evaluation simulator. We use the PCS from [21, 33] and modify the ZK evaluation protocol from [14] to obtain a PCS with the required properties. We discuss this scheme in appendix D.

[2] Reed-Solomon encoded IOPs [8, 22] are closely related to polyIOPs.

2.5 Preprocessing arguments

A preprocessing argument for an index relation $R \subseteq \mathcal{I} \times \mathsf{X} \times \mathcal{W}$ is a tuple $(\mathsf{ppA.Setup}, \mathsf{ppA.Index}, \mathsf{ppA.Prove})$. An index $i \in \mathcal{I}$ (e.g. a circuit description) represents verifier input that is preprocessed into an index key ik during the offline phase. The index key ik and instance $x \in \mathsf{X}$ (e.g. a partial wire assignment) represent the verifier input during the online phase. A witness $w \in \mathcal{W}$ (e.g. internal wire values) represents private prover input during the online phase.

- $\mathsf{ppA.Setup}(1^\lambda) \to \mathsf{pp}$: Given the security parameter, sample system parameters. A randomized algorithm.
- $\mathsf{ppA.Index}(\mathsf{pp}, i, r) \to \mathsf{ik}$: Given parameters, an index, and randomness, compute an index key. A deterministic algorithm.
- $\mathsf{ppA.Prove}(\mathcal{P}_\mathsf{P}(\mathsf{pp}, i, r, x, w), \mathcal{V}_\mathsf{P}(\mathsf{pp}, \mathsf{ik}, x)) \to (\perp, \{0,1\})$: An interactive protocol for $\mathcal{P}_\mathsf{P}$ to convince $\mathcal{V}_\mathsf{P}$ that it knows a w such that $(i, x, w) \in R$. Both $\mathcal{P}_\mathsf{P}$ and $\mathcal{V}_\mathsf{P}$ are interactive, randomized algorithms.

This syntax is based on [21]. Departing from [21], we've added a randomness token r to the arguments of $\mathsf{ppA.Index}$. As we will see, this allows ik to be a hiding commitment to i. In Section 4, we define security for arguments with hiding index keys. For an argument with non-hiding index keys, r is null. We formally define the efficiency and security requirements for a preprocessing argument as follows.

Completeness $\mathsf{ppA.Prove}$ is a complete protocol for the following binary relation.[3]

$$R_{\text{prove}}(\mathsf{pp}) = \{(i, x; w) : (i, x, w) \in R\}$$

Knowledge Soundness $\mathsf{ppA.Prove}$ is a an Argument of Knowledge for $R_{\text{prove}}(\mathsf{pp})$.

Zero-Knowledge $\mathsf{ppA.Prove}$ is an honest verifier zero knowledge protocol for $R_{\text{prove}}(\mathsf{pp})$.

Efficiency: We must have the following efficiency properties:

- *Index Efficient*: The running time of the prover is $\mathsf{poly}(\lambda, |i|)$.
- *Succinct Proof*: The communication transcript between the prover and verifier has size $\mathsf{poly}(\lambda)$.
- *Succinct Verifier*: The running time of the verifier is $\mathsf{poly}(\lambda + |x|)$.

Public Coin and Non-interactivity Every message of the verifier must be a uniform random string of some length. With the Fiat-Shamir Transformation [23] (assuming a small number of rounds between the prover and verifier), public coin protocols can be made non-interactive using random oracles.

A preprocessing zk-SNARK is a preprocessing argument that meets additional efficiency requirements, such as succinctness, and can be made non-interactive via Fiat-Shamir [23].

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 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_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] Technically, $\mathsf{ppA.Prove}$ takes an index key, but in the preprocessing model, the verifier has the index and derives the index key deterministically.

3 Functional Commitments

We next define 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

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$: Given the security parameter, 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.
- $\mathsf{Eval}(\mathcal{P}_\mathsf{E}(\mathsf{pp}, f \in \mathcal{F}, r \in \mathsf{R}, x \in \mathcal{X}, y \in \mathcal{Y}),\ \mathcal{V}_\mathsf{E}(\mathsf{pp}, c, x, y)) \to (\perp, \{0, 1\})$: an interactive protocol for $\mathcal{P}_\mathsf{E}$ to convince $\mathcal{V}_\mathsf{E}$ that $f(x) = y$.

where R is the randomness space and $\mathcal{C}$ is the commitment space.
Functional commitments have the following informal properties:

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 honest-verifier zero-knowledge*: an evaluation proof reveals nothing other than the evaluation
5. *Extractable*: evaluation proofs for a commitment convince $\mathcal{V}_\mathsf{E}$ that $\mathcal{P}_\mathsf{E}$ knows the underlying function encoding.
6. *Evaluation Binding*: A malicious prover cannot construct valid evaluation proofs for different evaluations on the same input. Evaluation Binding is directly implied by *binding* and *extractable* properties. We defer the formal definition to appendix A.1 and omit the straight forward proof.

We state these properties formally in Definition 4. The binding and hiding properties of functional commitments are exactly those for classical commitments.

Definition 4 *Functional commitment properties:*

The tuple $(\mathsf{Setup}, \mathsf{Commit})$ *is a* ***hiding*** *and* ***binding*** *commitment scheme for message space* $\mathcal{F}$ *and randomness space* R.

Completeness: Eval *is a complete protocol for the following binary 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* $R_{eval}(\mathsf{pp})$.

As portrayed in our introduction, the Eval protocol would most likely would be run in parallel for multiple clients. Our security definitions cover security in the case of parallel composition. By Theorem 2, Eval composed in parallel t times is also an honest verfier zero knowledge protocol. We also note that the security of sequential composition is implied by the security of parallel composition.

4 Functional commitments from preprocessing arguments

Our goal is to construct an efficient functional commitment scheme from a preprocessing argument. To do so, we require the preprocessing argument to satisfy a few non-standard properties. First, index keys must be a hiding and binding commitment to the index. Second, the proof protocol must be knowledge sound and zero-knowledge for the index as well as the witness. Third, and most notably, the preprocessing argument must support a novel kind of protocol we call a *proof of function relation* or a PFR.

Let us define these properties. Recall that a preprocessing argument ppA for an index relation $R \subseteq \mathcal{I} \times \mathsf{X} \times \mathcal{W}$ is a triple $(\mathsf{ppA.Setup}, \mathsf{ppA.Index}, \mathsf{ppA.Prove})$.

Definition 5. *A preprocessing argument* ppA *is* ***committing*** *if the pair of algorithms* $(\mathsf{ppA.Setup}, \mathsf{ppA.Index})$ *is a hiding and binding commitment scheme for the message space* $\mathcal{I}$*.*

The next two properties require that the proving protocol $\mathsf{ppA.Prove}$ be a zero-knowledge argument of knowledge for the index i as well as the witness w.

Definition 6. ppA *has* ***Index-extended knowledge soundness*** *if* $\mathsf{ppA.Prove}$ *is an Argument of Knowledge for the binary relation.*

$$R_{prove}(\mathsf{pp}) = \{(\mathsf{ik}, x \ ; \ i, r, w) : (i, x, w) \in R \wedge \mathsf{ik} = \mathsf{ppA.Index}(\mathsf{pp}, i, r)\}.$$

ppA *is an* ***index-extended honest-verifier zero-knowledge*** *if* $\mathsf{ppA.Prove}$ *is an HVZK protocol for* $R_{prove}(\mathsf{pp})$*.*

4.1 Proof of function relation (PFR)

Preprocessing arguments support proofs about relations that are not necessarily functions. Thus, we need a protocol to prove that a committed relation is a function: every input should have a unique output. To capture this, we first define the concept of a *functional set*, which is the subset of indices that encode functions.

Definition 7 (Functional Sets for Index Relations). *Let* $R \subseteq \mathcal{I} \times \mathsf{X} \times \mathcal{W}$ *be an index relation where* $\mathsf{X} = \mathcal{X} \times \mathcal{Y}$*. 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 an efficient 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, there must exist a unique y for this; thus, $\mathsf{Evaluate}$ returns a deterministic result.

A *proof of function relation* for a functional set $\mathcal{I}_f$ is a protocol Π between $\mathcal{P}_f$ and $\mathcal{V}_f$ which is a zero-knowledge argument of knowledge for the following relation:

$$R_{\mathsf{func}}(\mathcal{I}_f, \mathsf{pp}) = \{(\mathsf{ik}; i, r) : i \in \mathcal{I}_f \wedge \mathsf{ik} = \mathsf{ppA.Index}(\mathsf{pp}, i, r)\}$$

More precisely, Π should be extractable and zero-knowledge as in Definition 8.

Definition 8 ***Proof of function relation***

A proof of function relation for preprocessing argument ppA *and functional index set* $\mathcal{I}_f$ *is an interactive protocol*

$$\Pi(\mathcal{P}_f(\mathsf{pp}, i, r), \mathcal{V}_f(\mathsf{pp}, \mathsf{ik})) \to (\perp, \{0, 1\})$$

which is extractable and zero-knowledge.
Complete*:* Π *is a complete protocol for* $R_{\mathsf{func}}(\mathcal{I}_f, \mathsf{pp})$*.*
Extractable*:* Π *is an argument of knowledge for* $R_{\mathsf{func}}(\mathcal{I}_f, \mathsf{pp})$*.*

***Honest-verifier zero-knowledge**: Π is an HVZK protocol for $R_{\mathsf{func}}(\mathcal{I}_f, \mathsf{pp})$.*

We will construct our proofs of function relation as polyIOPs, where the relation's index is encoded as the evaluation of a collection of polynomials over a multiplicative subgroup. We will compile these protocols using Theorem 3.

4.2 Functional commitments from preprocessing arguments

Construction 1
Let $\mathsf{ppA} = (\mathsf{ppA.Setup}, \mathsf{ppA.Index}, \mathsf{ppA.Prove}, \mathsf{ppA.Verify})$ be a committing preprocessing argument—for an index relation $R \subseteq \mathcal{I} \times (\mathcal{X} \times \mathcal{Y}) \times \mathcal{W}$—and let Π be a proof of function relation for functional set $\mathcal{I}_f$ (an encoded function space), equipped with Extend. Let $\mathsf{FC}_{\mathsf{ppA},\mathcal{I}_f,\Pi}$ be the following tuple:

- $\mathsf{Setup}(1^\lambda)$: output $\mathsf{pp} \leftarrow \mathsf{ppA.Setup}(1^\lambda)$
- $\mathsf{Commit}(\mathsf{pp}, i, r)$: output $\mathsf{ik} \leftarrow \mathsf{ppA.Index}(\mathsf{pp}, i, r)$
- $\mathsf{Eval}(\mathcal{P}_\mathsf{E}(\mathsf{pp}, i, r, x, y), \mathcal{V}_\mathsf{E}(\mathsf{pp}, \mathsf{ik}, x, y))$:
  - $\mathcal{V}_\mathsf{E}$ and $\mathcal{P}_\mathsf{E}$: run $\Pi(\mathcal{P}_f(\mathsf{pp}, i, r), \mathcal{V}_f(\mathsf{pp}, \mathsf{ik}))$, and $\mathcal{V}_\mathsf{E}$ asserts that the output is 1.
  - $\mathcal{P}_\mathsf{E}$: $(y', w) \leftarrow \mathsf{Extend}(i, x)$, abort if $y \neq y'$.
  - $\mathcal{V}_\mathsf{E}$ and $\mathcal{P}_\mathsf{E}$: run $\mathsf{ppA.Prove}(\mathcal{P}_\mathsf{P}(\mathsf{pp}, i, r, (x, y), w), \mathcal{V}_\mathsf{P}(\mathsf{pp}, \mathsf{ik}, (x, y)))$ and $\mathcal{V}_\mathsf{E}$ asserts that the output is 1.

Note that Eval can be optimized so that the proof of function relation Π is run only once across many invocations of Eval.

Theorem 4. *Let* ppA *be a committing preprocessing argument with index-extended knowledge-soundness and honest-verifier zero-knowledge. Furthermore, let Π be a honest-verifier zero-knowledge argument of knowledge for function relations. Then $\mathsf{FC}_{\mathsf{ppA},\mathcal{I}_f,\Pi}$—as defined in Construction 1—is a secure functional commitment for function encodings $i \in \mathcal{I}_f$.*

Proof. Completeness follows directly from the completeness of ppA and Π. Binding and hiding follow directly from the binding and hiding properties of ppA.Index.

The functional commitment extractor builds on the proof of function relation extractor Ext_f and the preprocessing argument index-extended extractor $\mathsf{Ext}_\mathsf{ppA}$. First, it runs Ext_f to get $i \in \mathcal{I}_f$ and $r \in \mathsf{R}$ such that $\mathsf{ik} = \mathsf{ppA.Index}(\mathsf{pp}, i, r)$. Then, it runs $\mathsf{Ext}_\mathsf{ppA}$ to get to get $i' \in \mathcal{I}$, $w \in \mathcal{W}$, and $r' \in \mathsf{R}$ such that $\mathsf{ik} = \mathsf{ppA.Index}(\mathsf{pp}, i', r')$ and $(i', (x, y), w) \in R$. Per binding, $i = i'$, except with negligible probability. Thus, $(i, (x, y), w) \in R$, and we have extracted function encoding $i \in \mathcal{I}_f$ and randomness r consistent with the commitment ik, the input x, and the output y.

The honest-verifier zero-knowledge property for Eval follows from the same properties of Π and ppA. The simulator Sim_f for Π is used to simulate the verifier's view in the first part of the evaluation protocol. Then, since $\mathcal{P}_\mathsf{E}$ finds $(i, (x, y), w) \in R$, the simulator $\mathsf{Sim}_\mathsf{ppA}$ for ppA can be used to simulate $\mathcal{V}_\mathsf{E}$'s view in the second part of the evaluation protocol.

5 Polynomial Property Tests

Our goal is to construct functional commitment schemes from the Plonk and Marlin preprocessing arguments. The main technical step is to construct a suitable proof of function relation, a PFR, for the Plonk and Marlin index keys. Towards this goal, in this section we describe a number of polynomial IOPs (polyIOPs) for various properties of polynomials. These protocols provide the building blocks for our proofs of function relation presented in Sections 6 and 7.

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 $\mathbb{K}$.

Unless otherwise noted, all our 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, these queries are at random $x \in \mathbb{F} \setminus \mathbb{K}$. Fourth, they have perfect completeness. Five, using a technique from [21], our protocols can be updated to work when $\mathbb{K}$ is a coset of a multiplicative subgroup. Fourth, the prover's running time is always quasi-linear in the degree of the provided polynomials and verifier time is logarithmic in the degree (which is $< B = m + \mathsf{b}$ for some constant query bound b). Finally, all polynomials have degree less than bound $B = \mathsf{poly}(\lambda)$ such that $B/|\mathbb{F}| = \mathsf{negl}(\lambda)$.

Notation: By $R(a_1, \ldots, a_n) = \{f_1, \ldots, f_t \in \mathbb{F}^{(<B)}[X] : \phi(f_1, \ldots, f_t)\}$ we denote a relation over polynomials $f_1, \ldots, f_t$ that is parameterized by $a_1, \ldots, a_n$. The polynomials are known to $\mathcal{P}$, but $\mathcal{V}$ only has oracle access to them. The parameters $a_1, \ldots, a_n$ are known to both the prover and verifier.

5.1 Prior work

Protocol 2 (Equality over $\mathbb{K}$) Previous work [21, 25, 38] gives a polyIOP for the relation

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

The protocol only requires $\mathcal{V}$ to *query* f and g. Thus, it can be applied to polynomials which were not directly sent by the prover.[4] For example, it can be used to check that $f(X) \cdot f(X) = f(X)$ over $\mathbb{K}$. This protocol requires $\mathcal{V}$ to evaluate the vanishing polynomial, $z_{\mathbb{K}}(X) = \prod_{k \in \mathbb{K}}(X - k)$. When $\mathbb{K}$ is a subgroup or coset, this requires logarithmic time in the size of $\mathbb{K}$. When $\mathbb{K}$ is an arbitrary set, the time is linear.

Protocol 3 (Permutation composition over $\mathbb{K}$) Previous work [25] also gives a polyIOP for the relation

$$\{f, g, \mathsf{w} \in \mathbb{F}^{(<B)}[X] : \forall k \in \mathbb{K},\ f(k) = g(\mathsf{w}(k))\} \tag{1}$$

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

We note that Protocol 3 cannot be implemented by directly applying Protocol 2 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. [25] develop an elegant protocol for proving (1) where the prover only manipulates polynomials of degree $|\mathbb{K}|$. Technically, the protocol is for a slightly different relation:

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

for γ that generates $\mathbb{K}$ and $\mathsf{w}(\mathbb{K}) = [|\mathbb{K}|]$. However, adapting their protocol to our relation (1) is straightforward, see appendix C.

Protocol 4 (Subset over $\mathbb{K}$) Previous work [24] gives a polyIOP for relation

$$\{f, t \in \mathbb{F}^{(<B)}[X] : f(\mathbb{K}) \subseteq t(\mathbb{K})\} \tag{2}$$

With a minor change (discussed in appendix B.1), the polyIOP in [24] can be adapted to check the relation

$$\{f, t_1, ..., t_c \in \mathbb{F}^{(<B)}[X] : f(\mathbb{K}) \subseteq t_1(\mathbb{K}) \cup ... \cup t_c(\mathbb{K})\}$$

for some small constant $c \geq 1$ and polynomials t_1, ..., t_c such that $\|_{i=1}^{c} \mathrm{seq}_{\mathbb{K}}(t_i)$ has all repeat elements adjacent to one another. The requirement for t to have a structured image was implicitly assumed in [24]. We will compose this protocol with Protocol 5 to check the relation

$$\{f, t_1, ..., t_c \in \mathbb{F}^{(<B)}[X] : f(\mathbb{K}) \subseteq t_1(\mathbb{K}) \cup ... \cup t_c(\mathbb{K}) \ \wedge \ 0 \notin f(\mathbb{K})\} \tag{3}$$

We will refer to this protocol that checks (3) as Protocol 4. While this protocol suffices for our use case, we also describe a polyIOP for relation (2) in appendix B.5 which avoids any structure assumption on the image.

[4] Other works [21] call such polynomials "virtual oracles".

5.2 Non-zero over $\mathbb{K}$

We present a polyIOP that shows that for all $k \in \mathbb{K}$, $f(k) \neq 0$.

Protocol 5 (Non-zero over $\mathbb{K}$)
Relation: $\{f \in \mathbb{F}^{(<B)}[X] : \forall k \in \mathbb{K}, f(k) \neq 0\}$

---

1. Define $\forall k \in \mathbb{K}$, $g(k) = (f(k))^{-1}$. $\mathcal{P}$ interpolates and sends g to $\mathcal{V}$.
2. $\mathcal{P}$ and $\mathcal{V}$ run Protocol 2 to check for all $k \in \mathbb{K}$, $f(k)g(k) = 1$

Completeness and *soundness* are both immediate.

Note that this protocol can be used to test, for any $y \in \mathbb{F}$, that for all $k \in \mathbb{K}$, $f(k) \neq y$. Informally, the prover and verifier run the non-zero test on $f'(X) = f(X) - y$.

5.3 Multiset equality over $\mathbb{K}$

We present a polyIOP for verifying that the images of two polynomials are equal as multisets.[5] That is, that $\{\{f(k) : k \in \mathbb{K}\}\} = \{\{g(k) : k \in \mathbb{K}\}\}$.

Protocol 6 (Multiset Equality over $\mathbb{K}$)
Relation: $\left\{f, g \in \mathbb{F}^{(<B)}[X] : \{\{f(k) : k \in \mathbb{K}\}\} = \{\{g(k) : k \in \mathbb{K}\}\}\right\}$

---

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 $\mathcal{V}$.

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

3. $\mathcal{V}$ checks if $z(1) \stackrel{?}{=} 1$
4. $\mathcal{P}$ and $\mathcal{V}$ run Protocol 2 to check:

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

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.

5.4 Geometric sequence

We present a polyIOP that shows the sequence $\text{seq}_{\mathbb{K}}(f)$ is the concatenation of geometric sequences[6] 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

[5] This protocol can be modified to have perfect completeness as done in [25].
[6] This protocol can be generalized to other inductively defined sequences.

sequences that share the same multiplicative factor $r \in \mathbb{F}^*$. Let $c_1, c_2, ..., c_n \in \mathbb{N}$, where $c_1 + ... + c_n = m$, represent the number of terms in each geometric sequence. The relation $\mathcal{R}(r, \vec{a}, \vec{c})$ contains a polynomial f such that the sequence $f(1), f(\gamma), f(\gamma^2), ..., f(\gamma^{m-1})$ is identical to the sequence

$$a_1, a_1 r, ..., a_1 r^{c_1-1}, a_2, a_2 r, ..., a_2 r^{c_2-1}, ..., a_n, a_n r, ..., a_n r^{c_n-1}$$

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

Protocol 7 (Geometric Sequence Test)
Relation: $\mathcal{R}(r, \vec{a}, \vec{c}) = \left\{ f \in \mathbb{F}^{(<B)}[X] : \mathrm{seq}_{\mathbb{K}}(f) = \|_{i=1}^{n} (a_i, a_i r, \ldots, a_i r^{c_i-1}) \right\}$

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

1. $\mathcal{V}$ checks $\forall i \in [n], f(\gamma^{p_i}) \stackrel{?}{=} a_i$
2. $\mathcal{P}$ and $\mathcal{V}$ run Protocol 2 to check

$$\forall k \in \mathbb{K}, (f(\gamma \cdot k) - r \cdot f(k)) \cdot \prod_{i \in [n]} (k - \gamma^{p_i + c_i - 1}) \stackrel{?}{=} 0$$

Soundness Consider an arbitrary $i \in [n]$, we will prove by induction that the sequence $(f(\gamma^j) : p_i \leq j \leq p_i + c_i - 1)$ is identical to the geometric sequence $a_i, a_i r, ..., a_i r^{c_i-1}$. By the first identity, we know that $f(\gamma^{p_i}) = a_i$. If $c_i = 1$, then we are done. Otherwise, $\gamma^{p_i} \neq \gamma^{p_i + c_i - 1}$.

Let us consider j such that $p_i \leq j < p_i + c_i - 1$ and $\gamma^j \neq \gamma^{p_i + c_i - 1}$. Let $k = \gamma^j$. We must have that the right factor of the second identity is not zero, which implies the left factor must be zero. Then, this implies $f(\gamma \cdot k) = r \cdot f(k)$; in other words, the element next in the sequence must be the current multiplied by r. Thus, by induction, we must have the output sequence of $(f(\gamma^j) : p_i \leq j \leq p_i + c_i - 1)$ be the required geometric sequence. Since we considered an arbitrary $i \in [n]$, we must have that $\mathrm{seq}_{\mathbb{K}}(f)$ is the exact sequence required.

5.5 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 8 is a key protocol for our PFRs constructed in Sections 6.4 and 7.3.

Protocol 8 (Comparison Protocol)

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

where $m = |\mathbb{K}|$, $c \in \mathbb{N}$ is the least $c \geq 1 = O(1)$ such that $n < cm$, $\omega = \Delta^2$, and $\mathrm{ord}(\Delta) \geq 2n$ is even.

---

1. $\mathcal{P}$ interpolates and sends $s \leftarrow \mathsf{LDE}_{\mathbb{K}}(f/g)$
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 $\|_{i=1}^{c} \mathrm{seq}_{\mathbb{K}}(h_i)$ is the following sequence: $1, \Delta, \Delta^2, ..., \Delta^{n-1}$, $0, ..., 0$ with $cm - n$ zeroes.
4. $\mathcal{P}$ and $\mathcal{V}$ run Protocol 2 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 Protocol 7 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 Protocol 4 three times with $(p, h_1, ..., h_c)$ for $p \in \{f', g', s'\}$.
7. $\mathcal{P}$ and $\mathcal{V}$ run Protocol 5 to test $s(X) - 1$ is non-zero over $\mathbb{K}$.

Soundness By the soundness of the geometric sequence test, we have that $h_1(\mathbb{K}) \cup .... \cup h_c(\mathbb{K}) = \{\Delta^{e-1} : e \in [n]\} \cup \{0\}$. By soundness of the subset over $\mathbb{K}$ check (step 6), we have that: $f'(\mathbb{K}), g'(\mathbb{K}), s'(\mathbb{K}) \subseteq \bigcup_{i\in[c]} h_i(\mathbb{K}) \setminus \{0\} = \{\Delta^{e-1} : e \in [n]\}$.

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 $\mathbb{K}$ (step 4(a)), we know that $a \equiv b + c \pmod{\text{ord}(\Delta)}$; given the range of a, b, and c and $\text{ord}(\Delta) \geq 2n$, this implies that $a = b + c$ (as integers). Since $\Delta^2 = \omega$, $(f')^2 = f$, $(g')^2 = g$, and $(s')^2 = s$ (steps 4(b-d)), we have that $\log_\omega(f(k)) = a$, $\log_\omega(g(k)) = b$, $\log_\omega(s(k)) = c$. This immediately shows that $f(\mathbb{K}), g(\mathbb{K}) \subseteq \{1, \omega, \ldots, \omega^{n-1}\}$, which is one of our requirements. Furthermore, $\log_\omega(f(k)) = \log_\omega(g(k)) + \log_\omega(s(k))$. Since $\log_\omega(s(k)) = c \geq 0$ and since step 7 gives that $\log_\omega(s(k)) \neq 0$, we have $\log_\omega(f(k)) > \log_\omega(g(k))$. Since we considered an arbitrary $k \in \mathbb{K}$, this must hold for all $k \in \mathbb{K}$ as required.

6 Functional commitments from Marlin

Marlin [21] is a preprocessing argument for R1CS. In this section, we adapt it into a functional commitment. We begin with a review of the relevant parts of Marlin. Next, in Section 6.3, we describe how to modify Marlin to obtain the additional properties from Section 4. Then, in Section 6.4, we give a proof-of-function relation (Section 4.1) for the Marlin index key. Finally, Theorem 4 yields a functional commitment.

Relation and index Marlin is a preprocessing argument for $R_{R1CS}(n, h)$ as in Definition 9. Thus, an index is three matrices: $(A, B, C) \in (\mathbb{F}^{n\times n})^3$.

Definition 9 (Rank-1 Constraint System (R1CS)).
Let $\circ$ denote component-wise product. For $n \in \mathbb{N}$ constraints and $h \leq n$ instance variables, the rank-1 constraint system (R1CS) family of index relations is

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

Preliminaries Let $\mathbb{H}$ be a multiplicative subgroup of $\mathbb{F}$ with $|\mathbb{H}| = n$, $\langle\omega\rangle = \mathbb{H}$. Furthermore, let there exist order-$2n$ $\Delta \in \mathbb{F}$ such that $\omega = \Delta^2$. Let each index matrix contain at most $m = |\mathbb{K}| = O(n)$ non-zero entries, where $\mathbb{K}$ is a multiplicative subgroup of $\mathbb{F}$, generated by γ.

Index key The Marlin index key comprises of commitments to polynomials encoding the index matrices. For $M \in \{A, B, C\}$, let M be represented as a list of triples $(r_i, c_i, v_i)_{i=0}^{m-1} \in (\mathbb{N} \times \mathbb{N} \times \mathbb{F})^m$. Each triples represents a row number ($0 \leq r_i < n$), a column number ($0 \leq c_i < n$), and a value $v_i = M_{r_i,c_i}$. 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 $\text{val}_M(\gamma^i) = v_i$,[7] for $i \in \{0, \ldots, m-1\}$. The Marlin index key comprises of (non-hiding) commitments to row_M, col_M, and val_M for all $M \in \{A, B, C\}$.

6.1 A functional set for Marlin

Let $s, t \in \mathbb{N}_{>0}$ such that $s + t = h$. Unfortunately, for $\mathcal{X} = \mathbb{F}^t$, $\mathcal{Y} = \mathbb{F}^s$, and $\mathsf{X} = \mathcal{X} \times \mathcal{Y}$, the set of all indices $\mathcal{I}$ (for the relation $R_{R1CS}(n, h)$) is not a functional set. In this section, we describe a restriction of R1CS that captures bounded sized arithmetic circuits, and excludes non-functions. To simplify our functional set definition, we modify the relation slightly, moving the output instance variables to the *end* of the vector z.

[7] 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 [21]. The difference is unimportant to our protocols.

Definition 10 (Output-final R1CS).
Let $\circ$ denote component-wise product. For $n, t, s \in \mathbb{N}$ such that $t, s, s+t \in [n]$. Let $\mathcal{I} = (\mathbb{F}^{n\times n})^3$, $\mathcal{X} = \mathbb{F}^t$, $\mathcal{Y} = \mathbb{F}^s$, $\mathsf{X} = \mathcal{X} \times \mathcal{Y}$, $\mathcal{W} = \mathbb{F}^{n-t-s}$. We define the following family of index relations $R_{R1CS\text{-}f}(n, t, s) \subseteq \mathcal{I} \times \mathsf{X} \times \mathcal{W}$,

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

With this order, we can describe a functional set of indices $\mathcal{I}_f$, where each element of z beyond inputs is uniquely determined by the previous elements.

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.*

A matrix $M \in \mathbb{F}^{n\times n}$ 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.*

The following theorem shows that if we restrict $R_{\text{R1CS-f}}(n, t, s)$ indices to matrices $(A, B, C) \in t$-FT, then the residual binary relation over $\mathcal{X} \times \mathcal{Y}$ is a function.

Theorem 5. *For $R_{R1CS\text{-}f}(n, t, s)$, t-FT $\subseteq \mathcal{I}$ is a functional set.*

Proof. Let (A, B, C) be a functional triple. We want to show that

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

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:

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

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.

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 Compiler:

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 $s_i = 0$, 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 $s_i = 1$, 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 $i \leftarrow (A, B, C)$

6.2 Compiling arithmetic circuits to $R_{\text{R1CS-f}}$

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 from arithmetic circuits to $R_{\text{R1CS-f}}$.

Construction 2 is that compiler. 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.

Theorem 6. *For any bounded sized arithmetic circuit $C \in \mathcal{AC}_{n_{\mathsf{i}},n_{\mathsf{g}},n_{\mathsf{o}}}$, $\mathsf{Compiler}(C) \in t$-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 $(\mathsf{Compiler}(C), ((1, x), y), w) \in R_{R1CS\text{-}f}(n_{\mathsf{g}} + n_{\mathsf{i}} + 1, n_{\mathsf{i}} + 1, n_{\mathsf{o}})$*

Proof. The proof is straightforward and is omitted.

6.3 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 4.

Restricting the index encoding We will only work with C matrices which are t-diagonal (Definition 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: $M_{t,t}, M_{t+1,t+1}, \ldots, M_{n-1,n-1}, 0, 0, \ldots, 0$.

This encoding captures any t-diagonal C. Since this encoding is a restriction of Marlin's original encoding, it requires no protocol modifications.[8]

Additional properties To obtain the properties required by Theorem 4, we'll modify Marlin further. Our modifications (Construction 3) assume Marlin is compiled with an additive, hiding, and extractable PCS with ZK-Eval.

Construction 3 Extended Marlin *differs from* Marlin *in two ways:*

[8] Additionally, we adapt Marlin to target $R_{\text{R1CS-f}}$ instead of R_{R1CS}. This can be done in a straight forward manner.

1. *Modify* $\mathsf{ppA.Index}(\mathsf{pp}, i, r) \to \mathsf{ik}$*, use randomness* r *to create hiding commitments to the polynomials that encode* i*:* $\{row_M, col_M, val_M\}_{M \in \{A,B,C\}}$.
2. *Compile* $\mathsf{polyIOP}$ $\mathsf{ppA.Prove}(\dots)$ *using rerandomization to a standard protocol as described in Theorem 3.*

Theorem 7. Extended Marlin *satisfied the conditions of Theorem 4.*

Proof. The committing property of ($\mathsf{ppA.Setup}$, $\mathsf{ppA.Index}$) follows immediately from that of the PCS.

At a high-level, Extended Marlin achieves index-extended extractability (Definition 6) and honest-verifier zero-knowledge (Definition 6) through the compilation described in Theorem 3.

6.4 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 $\mathsf{polyIOP}$ that shows that polynomials $\{\text{row}_M, \text{col}_M, \text{val}_M\}_{M \in \{A,B,C\}}$ represent matrices $(A, B, C) \in t$-FT. This requires A and B to be t-strictly lower triangular and C to be t-diagonal.

t-strictly lower triangular For $M \in \{A, B\}$, we want to show that row_M and col_M encode matrix $M \in t$-SLT. To do so, Protocol 9 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 a *discrete-log comparison* protocol (Protocol 8) 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 a *geometric sequence* protocol (Protocol 7). Then, we show that image contains the image of row_M using a *subset* protocol (Protocol 4).

Protocol 9 (t-SLT Test)
Relation: $\{\text{row}_M, \text{col}_M, \text{val}_M \in \mathbb{F}^{(<B)}[X] : \text{Encode } M \in t\text{-SLT}\}$.

---

1. $\mathcal{P}$ interpolates and sends polynomial h such that $\text{seq}_{\mathbb{K}}(h)$ is:
$$\omega^t, \omega^{t+1}...., \omega^{n-1}, 0, 0, ..., 0$$
2. $\mathcal{P}$ and $\mathcal{V}$ run Protocol 7 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 Protocol 4 between row_M and h.
4. $\mathcal{P}$ and $\mathcal{V}$ run Protocol 8 between row_M and col_M, with parameters $(\Delta, n = |\mathbb{H}|)$ such that $\text{ord}(\Delta) = 2n, \Delta^2 = \omega$.

Soundness By the soundness of Protocol 7, $h(\mathbb{K}) = \{\omega^e : t \leq e \leq n-1\} \cup \{0\}$. By the soundness of Protocol 4, we have $\text{row}_M(\mathbb{K}) \subseteq \{\omega^e : t \leq e \leq n-1\}$. By the soundness of Protocol 8, we have $\text{row}_M(\mathbb{K}), \text{col}_M(\mathbb{K}) \subseteq \mathbb{H}$ and $\forall k \in \mathbb{K}$, $\log_\omega(\text{row}_M(k)) > \log_\omega(\text{col}_M(k))$. Therefore, $M \in t$-SLT.

***t*-diagonal** We want to verify that $(\text{row}_C, \text{col}_C, \text{val}_C)$ encodes a matrix $C \in t\text{-Diag}$. We can restrict the prover to known row_C and col_C polynomials. To do so, we can test for a polynomial h whose $\text{seq}_{\mathbb{K}}(h)$ is: $\omega^t, \omega^{t+1}...., \omega^{n-1}, 1, 1, ..., 1$.

Using an equality test over $\mathbb{K}$ (Protocol 2), we can test row_C and col_C are equal to this polynomial over $\mathbb{K}$. 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 $\text{seq}_{\mathbb{K}}(h_2)$ is: $0, 0, ..., 0, 1, 1, ..., 1$. The following is a polyIOP to test that an encoding represents a matrix M that is t-Diag.

Protocol 10 (t-Diag Test)

$$\textit{Relation: } \Big\{ \text{row}_M, \text{col}_M, \text{val}_M \in \mathbb{F}^{(<B)}[X] : \exists \vec{v} \in (\mathbb{F}^*)^{n-t}, \text{seq}_{\mathbb{K}}(\text{val}_M) = \vec{v} \| \vec{0} \\ \wedge \text{seq}_{\mathbb{K}}(\text{row}_M) = \text{seq}_{\mathbb{K}}(\text{col}_M) = (\omega^t, \omega^{t+1}, \ldots, \omega^{n-1}, 1, 1, \ldots, 1) \Big\}$$

These properties imply that $M \in t\text{-Diag}$.

---

1. $\mathcal{P}$ interpolates and sends two polynomials h_1, h_2 such that
   - $\text{seq}_{\mathbb{K}}(h_1)$ is the following: $\omega^t, \omega^{t+1}...., \omega^{n-1}, 0, 0, ..., 0$ in which there are $m - (n - t)$ zeroes.
   - $\text{seq}_{\mathbb{K}}(h_2)$ 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 Protocol 7 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 Protocol 7 on h_2 with initial values $0, 1$, multiplicative factor 1, and $c_1 = n - t, c_2 = m - (n - t)$.
4. Define $h := (h_1 + h_2)$. $\mathcal{P}$ and $\mathcal{V}$ run Protocol 2 between pairs (h, row_M) and $(\text{row}_M, \text{col}_M)$.
5. $\mathcal{P}$ and $\mathcal{V}$ run Protocol 2 to check for all $k \in \mathbb{K}$: $\text{val}_M(k) \cdot h_2(k) \stackrel{?}{=} 0$.
6. $\mathcal{P}$ and $\mathcal{V}$ run Protocol 5 to check for all $k \in \mathbb{K}$: $\text{val}_M(k) + h_2(k) \stackrel{?}{\neq} 0$.

Soundness Soundness of Protocol 7 and Protocol 2 implies that $\text{row}_M = \text{col}_M$ over $\mathbb{K}$ and that $\text{seq}_{\mathbb{K}}(\text{row}_M)$ is $\omega^t, \omega^{t+1}...., \omega^{n-1}, 1, 1, ..., 1$. This implies all nonzero entries are restricted to coordinates $(\omega^t, \omega^t), (\omega^{t+1}, \omega^{t+1}), ..., (\omega^{n-1}, \omega^{n-1}), (1, 1)$. By the soundness of Protocol 7, Protocol 2, and Protocol 5, 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 $(1, 1)$. This implies $M \in t\text{-Diag}$.

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 $(A, B, C) \in t\text{-FT}$.

Protocol 11 (t-FT Test)
Relation: $R_{t\text{-FT}} = \big\{(\text{row}_M, \text{col}_M, \text{val}_M)_{M \in \{A,B,C\}} \in \mathbb{F}^{(<B)}[X] : \text{Encode } (A, B, C) \in t\text{-FT}\big\}$.

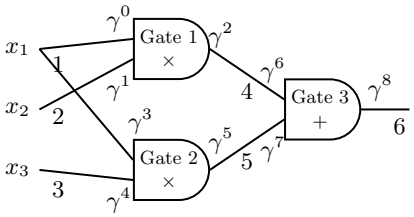
---

1. $\mathcal{P}$ and $\mathcal{V}$ run Protocol 9 on $(\text{row}_M, \text{col}_M, \text{val}_M)$ for $M \in \{A, B\}$
2. $\mathcal{P}$ and $\mathcal{V}$ run Protocol 10 on $(\text{row}_C, \text{col}_C, \text{val}_C)$

Theorem 8. *Protocol 11 is a* polyIOP *that is a sound and complete interactive argument for* $R_{t\text{-}FT}$

Proof. *Completeness* and *Soundness* are immediate from Protocol 9 and Protocol 10. □

The next corollary describes the proof of function relation derived from Protocol 11.



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

$x \in \mathbb{K}$	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. 2: A circuit and PLONK arithmetization for $n_\mathsf{g} = 3$, $n_\mathsf{i} = n_\mathsf{p} = 3$, $n_\mathsf{o} = 1$.

Corollary 1 (PFR for Marlin). *Let Π be the resulting protocol when Protocol 11 is compiled with a hiding, binding, additive, extractable, and zero-knowledge evaluation PCS with succinct commitments as described in Theorem 3. Then Π is a secure (i.e., complete, knowledge-sound, and zero-knowledge) PFR for* EXTENDED MARLIN. *$\mathcal{V}$'s runtime is logarithmic in m (the number of non-zero matrix entries), $\mathcal{P}$'s runtime is quasilinear in m, and proof size depends only on the security parameter.*

7 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 [25]. We follow [12], but represent circuit inputs slightly differently.

The PLONK *index relation* PLONK proves evaluations of arithmetic circuits. Figure 2 will be our running example: a circuit that computes $x_1x_2 + x_1x_3$. PLONK partitions the n_i circuit inputs into $n_\mathsf{p} \leq n_\mathsf{i}$ 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 2). 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 2 shows G, s, $\mathsf{Pins}^{\mathsf{In}}$, and $\mathsf{Pins}^{\mathsf{Out}}$ for our example, as well as valid choices of w and J.

7.1 Plonk Argument (Proving Protocol)

We briefly describe the PLONK argument from [25][12]. 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 (Protocol 3) 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 $\vec{y}$.

7.2 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 4, we'll modify PLONK as in EXTENDED MARLIN to hide the index (s and w) and compile to a standard protocol.

7.3 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 E) 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 Protocol 12. Condition 2 is checked with Protocol 13 and Protocol 14. Condition 3 is checked with Protocol 14. Protocol 15 is the complete proof of function relation.

Protocol 12 (w is a permutation on $\mathbb{K}$)
Relation: $\left\{\mathsf{w} \in \mathbb{F}^{(<B)}[X] : \mathsf{w}(\mathbb{K}) = \mathbb{K}\right\}$

1. Use Protocol 6 on $\mathsf{w}, g(X) = X$ to show $\{\!\{\mathsf{w}(k) : k \in \mathbb{K}\}\!\} = \{\!\{k : k \in \mathbb{K}\}\!\}$.

Completeness and soundness are immediate.

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

Protocol 13 (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 $\mathbb{K}$.

$$\textit{Relation: } \mathcal{R}(J, G) = \{\mathsf{w} \in \mathbb{F}^{(<B)}[X] : \mathsf{w} \text{ is a permutation over } \mathbb{K} \text{ and } \forall W \in \overline{W},\ W \cap (J \cup G) \neq \emptyset\}$$

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

$$\begin{cases} f(k) = 1 & \text{for } k \in I = J \cup G, \text{ and} \\ f(k) = \alpha \cdot f\big(\mathsf{w}^{-1}(k)\big) & \text{for } k \in \mathbb{K} \setminus I. \end{cases} \tag{4}$$

   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 Protocol 12 to show $\mathsf{w}(\mathbb{K}) = \mathbb{K}$.
4. Use Protocol 3 to show $f(X) = p(\mathsf{w}(X))$ over $\mathbb{K}$.
5. Use Protocol 2 to show that for all $k \in \mathbb{K}$,

$$\big(f(k) - \alpha \cdot p(k)\big) \cdot z_G(k) \cdot z_J(k) = 0, \tag{5}$$

   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 Protocol 5 to show that for all $k \in \mathbb{K}$, $f(k) \neq 0$.

Completeness We need to show that (5) 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 (4).

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 Protocol 2, 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 Protocol 3, 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 Protocol 5 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 $i \in W_i$.

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 $v : \mathbb{K} \to N = \{\beta^1, \ldots, \beta^{n_{\mathsf{g}}+n_{\mathsf{p}}}\}$. 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 $\beta^{i+n_{\mathsf{p}}}$. 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 Protocol 14 describes these conditions in detail.

Protocol 14 (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 $\mathbb{K}$.

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

---

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 i^{th} element of J, where J has an agreed upon ordering.
2. Use Protocol 2 to check $v(X) = u(X)$ over J.
3. Use Protocol 7 $\mathrm{seq}_G(v) = (\beta^{i+n_{\mathsf{p}}} : i \in [n_{\mathsf{g}}])$
4. Use Protocol 3 to show $v(\mathsf{w}(X)) = v(X)$ on $\mathbb{K}$.
5. Use Protocol 8 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 Protocol 8 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 Protocol 5 to check $v(X) - 1$ is nonzero over $\mathbb{K}$.

Soundness Step 2 directly implies the first property in $\mathcal{R}(J,G)$. From the soundness of Protocol 7, 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 Protocol 8, 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 (Protocol 5), we have $v(\mathbb{K}) \subseteq N$. From steps 2, 3 (Protocol 7), we must have $v(\mathbb{K}) = N$.

Protocol 15 (Plonk proof-of-function relation)
Relation: $R_{\mathcal{AC}} = \{(\mathsf{w}, s \in \mathbb{F}^{(<B)}[X]; C \in \mathcal{AC}_{n_{\mathsf{p}}, n_{\mathsf{g}}, n_{\mathsf{o}}}) : \mathsf{w}$ and s 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 Protocol 2 to show $s(X)s(X) = s(X)$ over $\mathbb{K}_{\mathsf{g}}$.
3. Invoke Protocol 12, Protocol 13, Protocol 14.

Theorem 9. *Protocol 15 is a* polyIOP *that is a sound and complete interactive argument for* $R_{\mathcal{AC}}$.

Proof. We prove completeness and soundness.

Completeness Completeness follows immediately from the completeness of protocols Protocol 2, Protocol 12, Protocol 13, Protocol 14.

Soundness We show that the properties required by our previous circuit extractor are guaranteed:

1. Protocol 12 shows that w is a permutation.
2. Protocol 13 shows
   - For all $W_i \in \overline{W}$, there exists $i \in I$ such that $i \in W_i$.

[10] A similar argument was made for the wire value assignment in PLONK [25].

– $|\overline{W}| \leq |I|$ as a corollary.

Protocol 14 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. Protocol 14 shows $\overline{W}$ can be topologically sorted.
4. Protocol 2 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. □

The next corollary describes the proof of function relation derived from Protocol 15.

Corollary 2 (PFR for Plonk). *Let Π be the standard protocol obtained by compiling Protocol 15 with a hiding, binding, additive, extractable and ZK-Eval PCS with succinct commitments as described in Theorem 3. Then, Π is a secure (i.e., complete, knowledge-sound, and zero-knowledge) PFR for* Plonk *where the prover time is quasilinear in n_{g}, verifier time is logarithmic in n_{g} and linear in the length of $J = |x|$, and the proof size depends only on the security parameter.*

8 Conclusion

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 4). In Section 6 we construct a proof of function relation (PFR) for a subset of Marlin index keys which capture all arithmetic circuits. In Section 7 we construct a PFR for Plonk index keys. Both protocols are public coin, send a constant number of polynomials, and make a constant number of queries. By combining these PFRs with their preprocessing arguments 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, proof size is constant, and 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 zk-SNARKs.

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*, 2017.

[2] P. Ananth and V. Vaikuntanathan. Optimal bounded-collusion secure functional encryption. *TCC 2019*, 2019.

[3] S. Arora and S. Safra. Probabilistic checking of proofs; A new characterization of NP. *33rd FOCS*, 1992.

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

[5] S. Barocas, M. Hardt, and A. Narayanan. *Fairness in machine learning*. https://fairmlbook.org/, 2017.

[6] M. Bellare and O. Goldreich. On defining proofs of knowledge. *CRYPTO'92*, 1993.

[7] E. Ben-Sasson, A. Chiesa, A. Gabizon, and M. Virza. Quasi-linear size zero knowledge from linear-algebraic PCPs. *TCC 2016-A*, 2016.

[8] E. Ben-Sasson, A. Chiesa, M. Riabzev, N. Spooner, M. Virza, and N. P. Ward. Aurora: Transparent succinct arguments for R1CS. *EUROCRYPT 2019*, 2019.
[9] E. Ben-Sasson, A. Chiesa, and N. Spooner. Interactive oracle proofs. *TCC 2016-B*, 2016.
[10] 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.
[11] N. Bitansky and V. Vaikuntanathan. Indistinguishability obfuscation from functional encryption. *56th FOCS*, 2015.
[12] D. Boneh. Building a SNARK. https://cs251.stanford.edu/lectures/lecture17.pdf, 2020.
[13] D. Boneh, B. Bünz, and B. Fisch. Batching techniques for accumulators with applications to IOPs and stateless blockchains. *CRYPTO 2019*, 2019.
[14] D. Boneh, J. Drake, B. Fisch, and A. Gabizon. Halo infinite: Proof-carrying data from additive polynomial commitments. *CRYPTO 2021*, 2021.
[15] D. Boneh, A. Sahai, and B. Waters. Functional encryption: Definitions and challenges. *TCC 2011*, 2011.
[16] J. Camenisch and A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. *CRYPTO 2002*, 2002.
[17] M. Campanelli, D. Fiore, N. Greco, D. Kolonelos, and L. Nizzardo. Incrementally aggregatable vector commitments and applications to verifiable decentralized storage. *ASIACRYPT 2020*, 2020.
[18] D. Catalano and D. Fiore. Vector commitments and their applications. *PKC 2013*, 2013.
[19] D. Catalano, D. Fiore, and M. Messina. Zero-knowledge sets with short proofs. *EUROCRYPT 2008*, 2008.
[20] M. Chase, A. Healy, A. Lysyanskaya, T. Malkin, and L. Reyzin. Mercurial commitments with applications to zero-knowledge sets. *EUROCRYPT 2005*, 2005.
[21] A. Chiesa, Y. Hu, M. Maller, P. Mishra, N. Vesely, and N. P. Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. *EUROCRYPT 2020*, 2020.
[22] A. Chiesa, D. Ojha, and N. Spooner. Fractal: Post-quantum and transparent recursive proofs from holography. *EUROCRYPT 2020*, 2020.
[23] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. *CRYPTO'86*, 1987.
[24] A. Gabizon and Z. J. Williamson. plookup: A simplified polynomial protocol for lookup tables. Cryptology ePrint Archive, Report 2020/315, 2020. https://eprint.iacr.org/2020/315.
[25] A. Gabizon, Z. J. Williamson, and O. Ciobotaru. PLONK: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953, 2019. https://eprint.iacr.org/2019/953.
[26] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. *54th FOCS*, 2013.
[27] S. Goldwasser, Y. T. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich. Reusable garbled circuits and succinct functional encryption. *45th ACM STOC*, 2013.
[28] S. Gorbunov, L. Reyzin, H. Wee, and Z. Zhang. Pointproofs: Aggregating proofs for multiple vector commitments. *ACM CCS 2020*, 2020.
[29] S. Gorbunov, V. Vaikuntanathan, and D. Wichs. Leveled fully homomorphic signatures from standard lattices. *47th ACM STOC*, 2015.
[30] J. Groth. On the size of pairing-based non-interactive arguments. *EUROCRYPT 2016*, 2016.
[31] J. Groth and M. Maller. Snarky signatures: Minimal signatures of knowledge from simulation-extractable SNARKs. *CRYPTO 2017*, 2017.
[32] A. Jain, A. Korb, N. Manohar, and A. Sahai. Amplifying the security of functional encryption, unconditionally. *CRYPTO 2020*, 2020.
[33] A. Kate, G. M. Zaverucha, and I. Goldberg. Constant-size commitments to polynomials and their applications. *ASIACRYPT 2010*, 2010.
[34] M. P.-S. Kim. *A Complexity-Theoretic Perspective on Fairness*. Stanford University, 2020.
[35] I. Komargodski and G. Segev. From minicrypt to obfustopia via private-key functional encryption. *Journal of Cryptology*, 33(2):406–458, 2020.

[36] B. Libert, S. C. Ramanna, and M. Yung. Functional commitment schemes: From polynomial commitments to pairing-based accumulators from simple assumptions. *ICALP 2016*, 2016.
[37] H. Lipmaa and K. Pavlyk. Succinct functional commitment for a large class of arithmetic circuits. *ASIACRYPT 2020*, 2020.
[38] 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*, 2019.
[39] S. Micali, M. O. Rabin, and J. Kilian. Zero-knowledge sets. *44th FOCS*, 2003.
[40] S. Micali, M. O. Rabin, and S. P. Vadhan. Verifiable random functions. *40th FOCS*, 1999.
[41] B. Parno, J. Howell, C. Gentry, and M. Raykova. Pinocchio: Nearly practical verifiable computation. *2013 IEEE Symposium on Security and Privacy*, 2013.
[42] C. Peikert, Z. Pepin, and C. Sharp. Vector and functional commitments from lattices. *TCC*, 2021.
[43] A. Sahai and B. R. Waters. Fuzzy identity-based encryption. *EUROCRYPT 2005*, 2005.
[44] S. Setty. Spartan: Efficient and general-purpose zkSNARKs without trusted setup. *CRYPTO 2020*, 2020.
[45] B. Waters. A punctured programming approach to adaptively secure functional encryption. *CRYPTO 2015*, 2015.
[46] T. Xie, J. Zhang, Y. Zhang, C. Papamanthou, and D. Song. Libra: Succinct zero-knowledge proofs with optimal prover computation. *CRYPTO 2019*, 2019.
[47] A. C.-C. Yao. How to generate and exchange secrets (extended abstract). *27th FOCS*, 1986.

Appendix

A Definitions

A.1 Evaluation binding

Definition 12. ***Evaluation Binding*** *For all pairs of* PPT *adversaries* $(\mathcal{P}_1, \mathcal{P}_2)$,

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

B polyIOP for image subset

B.1 Multiple Polynomial Subset over $\mathbb{K}$

We will apply a straight forward adaptation to the polyIOP discussed in [24] to check the relation

$$\{f, t_1, ..., t_c \in \mathbb{F}^{(<B)}[X] : f(\mathbb{K}) \subseteq t_1(\mathbb{K}) \cup ... \cup t_c(\mathbb{K})\}$$

We will refer to section 3 of [24] for notation and construction. 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:

$$\begin{aligned} F(\beta, \gamma) := (1+\beta)^n \prod_{i\in[n]} (\gamma + f_i) \cdot \prod_{j\in[c]} \left[\textstyle\prod_{i\in[d-1]} (\gamma(1+\beta) + t_{j,i} + \beta t_{j,i+1})\right] \\ \cdot \prod_{i\in[c-1]} \left[\gamma(1+\beta) + t_{j,d} + \beta t_{j+1,1}\right] \\ G(\beta, \gamma) := \prod_{i\in[n+cd-1]} (\gamma(1+\beta) + s_i + \beta s_{i+1}) \end{aligned}$$

The remaining proof and protocol (polyIOP) from section 3 of [24] follow with only minor modifications.

B.2 Shadow over $\mathbb{K}$

Definition 13. *Shadow over* $\mathbb{K}$
Let f, g *be two polynomials.* f *is in the* ***shadow*** *of* g *over* $\mathbb{K}$ *if for all* $k \in \mathbb{K}$,

$$f(k) \in \{g(k), 0\}$$

We present a polyIOP that shows that f is in the shadow of g.

Protocol 16 (Shadow over $\mathbb{K}$)
Relation: $\{f, g \in \mathbb{F}^{(<B)}[X] : \forall k \in \mathbb{K}, f(k) \in \{g(k), 0\}\}$

---

1. $\mathcal{P}$ interpolates $s(X)$ defined below and sends s to $\mathcal{V}$.

$$\forall k \in \mathbb{K},\ s(k) = \begin{cases} 1 & f(k) = g(k) \\ 0 & o.w. \end{cases}$$

2. $\mathcal{P}$ and $\mathcal{V}$ run Protocol 2 to check for all $k \in \mathbb{K}$:

$$s(k) \stackrel{?}{=} s(k)s(k),\ f(k) \stackrel{?}{=} g(k)s(k)$$

Completeness and *soundness* are immediate.

B.3 Zero-knowledge point equality

For $a \in \mathbb{F}$, we present a polyIOP that shows that for polynomials f and g, $f(a) = g(a)$, without revealing $f(a)$.

Protocol 17 (Point Equality)
Relation: $\mathcal{R}_{\text{point}}(a) = \{f, g \in \mathbb{F}^{(<B)}[X] : f(a) = g(a)\}$

---

1. $\mathcal{P}$ computes $h := f - g$, with degree equal to the maximum degree of f and g, and sends h to $\mathcal{V}$.
2. For random $c \xleftarrow{\$} \mathbb{F} \setminus \mathbb{K}$, $\mathcal{V}$ checks $h(c) \stackrel{?}{=} f(c) - g(c)$ and $h(a) \stackrel{?}{=} 0$

Completeness and *soundness* are both immediate.

B.4 De-duplication

We present a polyIOP that shows that for polynomials f and g, $\text{seq}_{\mathbb{K}}(f)$ is $\text{seq}_{\mathbb{K}}(g)$ with duplicate neighbors set to zero.

Definition 14. *De-Duplication over* $\mathbb{K}$
For polynomials f and g, g is the ***de-duplication*** *of f over $\mathbb{K}$ if and only if $f(1) = g(1)$ and for all $i \in [m-1]$*

$$g(\gamma^i) = \begin{cases} 0 & f(\gamma^{i-1}) = f(\gamma^i) \\ f(\gamma^i) & o.w. \end{cases}$$

Protocol 18 (De-Duplication over $\mathbb{K}$)
Relation: $\{f, g \in \mathbb{F}^{(<B)}[X] : g \text{ is the de-duplication of } f \text{ over } \mathbb{K}\}$

---

1. $\mathcal{V}$ sends a random challenge $c \xleftarrow{\$} \mathbb{F}$ to $\mathcal{P}$.
2. $\mathcal{P}$ and $\mathcal{V}$ run Protocol 17 and Protocol 2 to check

$$f(1) \stackrel{?}{=} g(1) \tag{6}$$

$$\forall k \in \mathbb{K}, (f(k) - g(k)) \cdot (g(k) + c(f(\gamma^{-1}k) - f(k))) \stackrel{?}{=} 0 \tag{7}$$

Soundness We prove only a limited notion of soundness: every unique element in the image of f over $\mathbb{K}$ must be contained in the image of g over $\mathbb{K}$. This is the notion we need for future uses of de-duplication. More formally, we show that the protocol proves that $f(\mathbb{K}) \subseteq g(\mathbb{K})$. For any element $y \in f(\mathbb{K})$, there exists a minimal $k \in \mathbb{K}$ such that $f(k) = y$ (in other words, $k \leq k' \in \mathbb{K}$ for all $k' \in \mathbb{K}$ such that $f(k') = y$). There are two cases:

- $k = 1$: We know that $y = f(1) = g(1) \in g(\mathbb{K})$ by the point equality check.
- $k \neq 1$: Since k is minimal, we must have that $f(\gamma^{-1}k) \neq y = f(k)$. Then, over the randomness of c, $g(k) + c(f(\gamma^{-1}k) - f(k))$ is non-zero with all but negligible probability. Therefore, to satisfy Equation 7, we must have $g(k) - f(k) = 0$. This implies $y = g(k) \in g(\mathbb{K})$.

B.5 Subset over $\mathbb{K}$

We present a polyIOP for verifying that the images of two polynomials f and g satisfy a subset relation. We impose the extra requirement that $0 \notin f(\mathbb{K})$.

Protocol 19 (Subset over $\mathbb{K}$)
Relation: $\{f, g \in \mathbb{F}^{(<B)}[X] : f(\mathbb{K}) \subseteq g(\mathbb{K}) \wedge 0 \notin f(\mathbb{K})\}$.

---

1. $\mathcal{P}$ interpolates the following polynomials and sends them to $\mathcal{V}$.
   - f' such that $\text{seq}_{\mathbb{K}}(f')$ is the sorted version of $\text{seq}_{\mathbb{K}}(f)$.
   - f'' be the de-duplication of f' over $\mathbb{K}$.
   - g' such that
     - g' is in the shadow of g over $\mathbb{K}$.
     - $g'(\mathbb{K}) \setminus \{0\} = f(\mathbb{K})$
     - Every element in $\text{seq}_{\mathbb{K}}(g')$ is unique except for 0.
2. $\mathcal{P}$ and $\mathcal{V}$ run Protocol 5 to check that f is non-zero on $\mathbb{K}$.
3. $\mathcal{P}$ and $\mathcal{V}$ run Protocol 6 to check that f' and f are equal as multisets.
4. $\mathcal{P}$ and $\mathcal{V}$ run Protocol 18 (de-duplication) to check that $f'(\mathbb{K}) \subseteq f''(\mathbb{K})$.
5. $\mathcal{P}$ and $\mathcal{V}$ run Protocol 6 to check that f'' and g' are equal as multisets.
6. $\mathcal{P}$ and $\mathcal{V}$ run Protocol 16 to check that g' is in the shadow of g over $\mathbb{K}$.

Soundness Consider $y \in f(\mathbb{K})$. Per the first multiset equality protocol's soundness, $y \in f'(\mathbb{K})$. Per the soundness of de-duplication, $y \in f''(\mathbb{K})$. Per the second multiset equality protocol's soundness, $y \in g'(\mathbb{K})$. Then, per the soundness of the shadow protocol, y is 0 or in $g(\mathbb{K})$. However, the non-zero protocol's soundness guarantee's that y is non-zero, so $y \in g(\mathbb{K})$.

C polyIOP for permutation composition

We present a polyIOP for the relation

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

where w is a permutation on $\mathbb{K}$. Our protocol is an adaptation of a similar protocol from [25].

Protocol 20 (Permutation Composition over $\mathbb{K}$)

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

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 Protocol 6 to show that the evaluations of $\mathsf{w}(X) + \beta f(X)$ and $X + \beta g(X)$ are equal as multisets.

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 $\mathbb{K}$.

D Additive PCS Scheme with ZK Eval

MARLIN [21] modifies the PCS from [33] to obtain an additive, hiding, binding, and extractable PCS with constant sized commitments. We will use this scheme to obtain a PCS which preserves those properties and has a HVZK evaluation protocol. To do this, we will modify the ZK evaluation protocol from [14].

Protocol 21 (HVZK Eval)
Given: An additive, hiding, and binding PCS = (Setup, Commit, Verify, Eval). Public inputs (pp, C_f, z, y) and private prover inputs $(f \in \mathbb{F}^{(<d)}[X], \mathsf{open}_f)$, where open_f is the commitment randomness.

---

1. $\mathcal{P}$ samples $r(X) \xleftarrow{\$} \mathbb{F}^{(<d)}[X]$ and computes $C_r = \mathsf{Commit}(r(X), \mathsf{open}_r)$. $\mathcal{P}$ sends C_r and $r(z)$ to the $\mathcal{V}$.
2. $\mathcal{V}$ sends random challenge $c \xleftarrow{\$} \mathbb{F}$ to $\mathcal{P}$.
3. We now invoke the additive property of the PCS. $\mathcal{P}$ computes $s := r + c \cdot f$ and $\mathsf{open}_s := \mathsf{open}_r + c \cdot \mathsf{open}_f$. $\mathcal{P}$ and $\mathcal{V}$ derive $C_s := C_r + c \cdot C_f$.
4. $\mathcal{P}$ and $\mathcal{V}$ run Eval on public inputs $(\mathsf{pp}, C_s, z, r(z) + c \cdot y)$ and private prover input (s, open_s). $\mathcal{V}$ accepts if Eval accepts.

Lemma 1. *(Informal) If* Eval *is public coin, complete, and knowledge sound, then Protocol 21 is public coin, complete, knowledge sound, and HVZK.*

Proof Sketch Small modifications to proof shown in [14] will suffice. To preserve knowledge soundness (instead of witness extended emulation), we can replace the extractor with the one from [4].

E Partition Lemma

Lemma 2. *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 $S' \subseteq T$. 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 $\mathcal{S} = \mathcal{T}$.