# Revealable Functional Commitments: How to Partially Reveal a Secret Function*

Bharath Raj Namboothiry

Stanford University
brn@stanford.edu

**Abstract.** A revealable functional commitment allows a prover to commit to a secret polynomial size function $f$. Later, the prover has the ability to (1) prove that $y = f(x)$ for public $x, y$ and (2) open a small window into $f$'s machinery, via an encoded set of constraints - all without divulging any other information about $f$. In this way, revealable functional commitments allow the operator of a proprietary function to prove desired predicate about the function. For example, a government can commit to a bail decision algorithm, and prove that the same algorithm is being used for all defendants. They can also quell concerns about bias, and increase transparency processes by revealing windows into what their function does - while keeping most of their function secret to prevent exploitation. To build a revealable functional commitment, we introduce a *proof of reveal*, to show that a set of constraints, combined with a set of guarantees about those constraints, is consistent with a committed secret function. We show that combining a algebraic holomorphic proof (AHP), a *proof of function relation (PFR)* [4], and a proof of reveal yields a secure revealable functional commitment scheme. Additionally, we construct proof of reveals for two popular PFR-equipped AHPs, and obtain two instantiations of revealable functional commitments. Towards that end, we also develop interactive protocols that prove properties of committed polynomials, which may have independent value.

**Keywords:** Cryptography, ZK Protocols, Commitment Schemes

## 1 Introduction

We introduce a new primitive called *revealable functional commitments*. In this primitive, a committer commits to a *secret* function $f : \mathcal{X} \rightarrow \mathcal{Y}$ via a succinct hiding and binding commitment scheme. Then, the committer can open the function at any public point $(x, y)$, and prove that $f(x) = y$ without revealing anything else about the function. Additionally, the committer can reveal an arbitrarily sized public *portion* of the function, and prove that (1) the portion is contained in the function and (2) aside from designated values, the portion is self-contained.

* Special thanks to Dan Boneh and Stanford ACG for their mentorship and guidance

More specifically, a revealable function commitment is a tuple $(\mathsf{Setup}, \mathsf{Commit}, \mathsf{Eval}, \mathsf{Reveal})$. $\mathsf{Setup}(1^\lambda, N)$ is a randomized algorithm that generates public parameters $\mathsf{pp}$, which will support the commitments to functions of size $N$. $\mathsf{Commit}(\mathsf{pp}, f, r)$ is a deterministic algorithm that takes as input the function description $f$, along with randomness $r$, and outputs a hiding and binding commitment $c$. $\mathsf{Eval}$ is a interactive protocol between a prover $\mathcal{P}_E(\mathsf{pp}, f, r, x, y)$ and a verifier $\mathcal{V}_E(\mathsf{pp}, c, x, y)$, which convinces the verifier that $f(x) = y$. Finally, $\mathsf{Reveal}$ is an interactive protocol between a prover $\mathcal{P}_R(\mathsf{pp}, f, r, q)$ and a verifier $\mathcal{V}_R(\mathsf{pp}, c, q)$ which convinces the verifier that $q$ is a *valid reveal* of $f$.

Informally, to be a valid reveal, $q$ must be a description of a subset of the constraints $f$ imposes on its inputs. Additionally, values that $q$ constrains must not be constrained elsewhere, unless pre-designated by the prover. This second condition is necessary to ensure that $q$ actually discloses behavior of the function, and cannot be "worked around" via intersecting constraints. This definition is presented formally in section 3. Intuitively, the $\mathsf{Reveal}$ protocol does not disclose any information about $f$ that is not contained in $q$. The security properties we seek for the entire scheme are formally defined in section 4.

Revealable functional commitments contribute the the budding field of algorithmic fairness. Using the $\mathsf{Commit}$ method, parties can commit to proprietary algorithms, and using the $\mathsf{Eval}$ method, the public can rest assured that the same algorithm is being applied to all, as the commitment is binding. Additionally, using the $\mathsf{Reveal}$ method, the owners of the algorithm can publish portions of the function to prove to the public that the algorithm abides by agreed upon fairness or compliance criteria. Below are some examples of real world applications of this scheme:

- Bail Decisions: When a person is arrested (in the United States), they typically have a hearing where a judge decides if they should be released while they await trial and if so, how much it will cost them. Unfortunately, bail decisions are very prone to bias [1]. Recent legislature has proposed using an algorithm to make bail decisions - an algorithm that would likely have to be kept secret to prevent exploits. With a revealable functional commitment, a government can make a hiding and binding commitment to a bail function, and using the $\mathsf{Eval}$ method, defendants can ensure that the same function is being applied to all. Additionally, machinery of the bail function can be published to quell bias concerns. For example, the government can publish a hash function, and prove that (1) the name, race, and gender of a defendant are hashed in the function and (2) this information is not available to any part of the function other than the published hash machinery.
- Credit Scoring: Credit bureaus in the United States assign credit scores to customers, which are then used by various institutions for lending terms. These algorithms are company secrets, and are therefore kept proprietary. With a revealable functional commitment scheme, a credit bureau could commit to a secret credit scoring function, and the public could be assured that the same function is being used for all. Additionally, by publishing ma-

chinery, they could increase the amount of transparency into their algorithm, while keeping privileged information hidden.

### 1.1 Previous Work

This work builds off of the previous work of Boneh, Nguyen, and Ozedmir [4]. In their paper, they introduce function hiding functional commitments, which allow a prover to commit to a secret function, and then prove openings of the committed function at a public points. Their construction consists exactly of the methods $\mathsf{Setup}, \mathsf{Commit}, \mathsf{Eval}$. This work allowed for a function to be private, with the assurance that all evaluations actually came from the private function. However, in the setting of algorithmic fairness, we are concerned with more than just the consistency of the algorithm - we want to know the algorithm was not biased in the first place. Therefore, this papers construction, with the inclusion of the $\mathsf{Reveal}$ method, aims to tackle this exact problem.

## 2 Preliminaries

### 2.1 Notation and Terminology

For the reader's reference, we will begin by covering mathematical notation and terminology used in the paper. For $n \in \mathbb{N}_{>0}$ we denote $[n]$ to be the sequence $(1, 2, \ldots, n)$. We denote multisets as $\{\{\cdot\}\}$. We use $||$ to denote the concatenation operator. Generally, $\lambda$ will refer to the security parameter. We consider a function $f(n)$ as $\mathsf{poly}(n)$ if there is some $c \in \mathbb{N}$ such that $f(n) = O(n^c)$. We consider a function $g(n)$ as $\mathsf{negl}(n)$ if for all $c \in \mathbb{N}$, $g(n) = o(n^{-c})$. We refer to any such $g$ as *negligible*. We call a probability of $1 - \mathsf{negl}(n)$ an *overwhelming probability*. We call an algorithm $\mathsf{PPT}$ if it is probabilistic and runs in time polynomial in its input size.

Let $\mathbb{F}$ be a field of large prime order $p$ with a canonical ordering such that $\log(p) = \Omega(\lambda)$, and $2^k \mid (p-1)$ for some $k \in \mathbb{N}$. In the discussion of Plonk, we also require that $3 \mid (p-1)$. For $\gamma \in F^*$, let $\langle\gamma\rangle = \{y^i\}_{i \in \mathbb{N}}$. Let $\mathbb{F}^{(<d)}[X]$ denote the set of polynomials in $X$ of degree $< d$ with coefficients in $\mathbb{F}$.

Let $\{D_\lambda\}_{\lambda \in \mathbb{N}}, \{D'_\lambda\}_{\lambda \in \mathbb{N}}$ be two families of probability distributions. When it is clear from the context, we write $\{D\} = \{D'\}$ to indicate that the distributions are the same.

### 2.2 Commitment Schemes

A commitment scheme for a set of messages $\mathcal{X}$ consists of the following two algorithms:

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

A commitment scheme must satisfy the following two properties:

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

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

– *Perfect Hiding:* For all $x_1, x_2 \in \mathcal{X}$, with $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$,

$$\{\mathsf{Commit}(\mathsf{pp}, x_1, r) : r_2 \overset{\$}{\leftarrow} \mathsf{R}\} = \{\mathsf{Commit}(\mathsf{pp}, x_2, r_2) : r_2 \overset{\$}{\leftarrow} \mathsf{R}\}$$

## 2.3 Polynomial Commitment Schemes

Polynomial Commitment Schemes (PCS) [3,9,5] allow a prover to commit to a polynomial $f \in \mathbb{F}[X]$ with degree up to $d$. Afterwards, the prover can convince the verifier that for some $x, y \in \mathbb{F}$, $f(x) = y$. A PCS is defined by the following algorithms:

– $\mathsf{PC.Setup}(1^\lambda, \{d_i\}_i) \rightarrow (\mathsf{ck}, \mathsf{vk})$ : Given the security parameter and a set of degree bounds, output a commitment key ($\mathsf{ck}$) and a verifying key ($\mathsf{vk}$).
– $\mathsf{PC.Commit}(\mathsf{ck}, f \in \mathbb{F}^{<d_i}[X], d_i, r) \rightarrow c$ : Given the commitment key, a polynomial $f$ of degree less than $d_i$, and randomness $r$, output a commitment $c$ to $f$
– $\mathsf{PC.Eval}(\mathsf{ck}, f \in \mathbb{F}^{<d_i}[X], d_i, r, x \in \mathbb{F}) \rightarrow \pi$ : Given the commitment key, polynomial $f$ of degree less than $d_i$, randomness $r$ and an evaluation point $x \in \mathbb{F}$, output an evaluation proof $\pi$
– $\mathsf{PC.CheckEval}(\mathsf{vk}, c, d_i, x \in F, y \in F, \pi) \rightarrow \{0, 1\}$ : Given verifying key, commitment $c$, degree bound $d_i$, evaluation point $x$ and claimed evaluation output $y$, output decision $\in \{0, 1\}$

We consider a PCS *secure* if it is a hiding and binding commitment to $f$, with an evaluation proof that is an argument of knowledge. Additionally, evaluation proofs may be zero-knowledge. In this paper, we will use the PCS scheme from [4], which modifies [5] to make the evaluation routine honest-verifier zero-knowledge.

**Homomorphisms** For commitments respecting the same degree bound, the polynomial commitment schemes in [4,3,9,5] have commitment and randomness spaces that are linearly homomorphic. That is, if a verifier has commitments $c_1, c_2$ to polynomials $f_1, f_2$, they can derive a commitment $c_3$ to polynomial $f_3$ if $f_3$ is a linear combination of $f_1$ and $f_2$. Similarly, a prover can derive the commitment randomness $r_3$ of $c_3$ from a linear combination of the randomness of $c_1$ and $c_2$

## 2.4 Polynomial Interactive Oracle Proofs

Polynomial interactive oracle proofs [4], or $\mathsf{polyIOP}$s, are powerful tools that will be the workhorses of our construction. Before giving a definition, it is helpful to review oracle relations.

**Oracle Relations** Let $\mathcal{O} \subseteq (\mathbb{F}[X])^c$ be an oracle space for feild $\mathbb{F}$ and constant $c$. Let $\mathcal{X} \subseteq \{0,1\}^*$ be an instance space, and let $\mathcal{W} \subseteq \{0,1\}^*$ be a witness space. An *oracle relation* $\mathcal{R} \subset \mathcal{O} \times \mathcal{X} \times \mathcal{W}$ is a set of triples $(\vec{o}, x, w)$, and the language of $\mathcal{R}$, denoted $\mathcal{L}(\mathcal{R})$ is the set of pairs $(\vec{o}, x)$ where there exists $w \in \mathcal{W}$ such that $(\vec{o}, x, w) \in \mathcal{R}$. We denote $\mathcal{R}_N$ as a restriction of $\mathcal{R}$ with $\|\vec{o}\| \leq N \in \mathbb{N}$.

A $\mathsf{polyIOP}$ for an oracle relation $\mathcal{R}$ is an interactive proof between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$. In this protocol, $\mathcal{P}$ has the ability to send *polynomial oracles* that $\mathcal{V}$ can access only through evaluation queries. Concretely, a $\mathsf{polyIOP}$ is a tuple:

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

Above, $\mathcal{P}, \mathcal{V}$ are $\mathsf{PPT}$ interactive algorithms. For $(\vec{o}, x, w) \in \mathcal{R}_N$, $\mathcal{P}$ receives inputs $(\mathbb{F}, \vec{o}, x, w)$ and $\mathcal{V}$ receives $(F, x, N)$ and input oracles $\vec{o}$. $\mathcal{P}$ and $\mathcal{V}$ then interact, during which $\mathcal{P}$ can only send oracles to $\mathcal{V}$. At any point, $\mathcal{V}$ can query any of the sent or input oracles it has. After $k$ rounds of interaction, $\mathcal{V}$ outputs an accept/reject decision.

The functions $s, d$ bound the number and degrees of polynomials sent in every round. For each round $i \in [k]$, $\mathcal{V}$ can send a message $m_i \in \mathbb{F}^*$, and $\mathcal{P}$ sends back $s(i)$ oracles $o_{i,1} \ldots o_{i,s(i)}$ in $\mathbb{F}[X]$ such that $\forall j \in [s(i)]$, $\deg(o_{i,j}) < d(N, i, j)$. We denote the input oracles as $\vec{o} = (o_{0,1} \ldots o_{0,s(0)})$. This, $s(0), \{d(N, 0, j)\}_j$ encode number and degree bounds on $\vec{o}$. A pair $(\vec{o}, \tilde{P})$ is considered *admissable* if the degree bounds encoded in $d$ are satisfied by all input and sent oracles. Also, oracles are additive, meaning that if $\mathcal{V}$ has access to oracles $f, g$, it can derive and query a new oracle $f + g$.

**PolyIOP Security** A $\mathsf{polyIOP}$ can have the following properties:

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

$$Pr[\langle \mathcal{P}(\mathbb{F}, \vec{o}, x, w), \mathcal{V}^{\vec{o}}(\mathbb{F}, x, N) \rangle = 0] \leq \mathsf{negl}(\lambda)$$

  i.e. if the statement is in the relation, then the verifier should accept with probability close to 1.
- **Soundness:** For all $(\vec{o}, x) \in \mathcal{L}(\mathcal{R})$,

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

  i.e. for a statement not belonging to the language, the verifier only accepts it with negligible probability.

- **Knowledge Soundness:** A $\mathsf{polyIOP}$ has knowledge error $\epsilon$ if there exists a $\mathsf{PPT}$ extractor $\mathcal{E}$ such that for all oracles $\vec{o}$, instances $x$, and $\mathsf{PPT}$ adversaries $\tilde{\mathcal{P}}$ such that

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

  where $\mathcal{E}^{\tilde{\mathcal{P}}}$ means the extractor has blackbox rewind access to the prover $\tilde{\mathcal{P}}$ as a set of next-message functions. When $\epsilon$ is negligible, we consider the $\mathsf{polyIOP}$ knowledge-sound
- **(Perfect) Honest Verifier Zero Knowledge:** There exists a $\mathsf{PPT}$ simulator $\mathcal{S}$, for all $(\vec{o}, x, w) \in \mathcal{R}$,

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

  where the $\mathsf{View}\left(\langle \mathcal{P}(\mathbb{F}, \vec{o}, x, w), \mathcal{V}^{\vec{o}}(\mathbb{F}, x, N)\rangle\right)$ is the view of the honest verifier $\mathcal{V}$ during the interaction.

**Definition 1 (Secure PolyIOP).** *A* $\mathsf{polyIOP}$ *for relation $\mathcal{R}$ is* ***secure*** *if it is complete, sound, knowledge sound, and has (perfect) honest-verifier zero-knowledge for $\mathcal{R}$*

**Virtual Oracles** Oracles sent by the prover or given initially are referred to as *concrete* oracles. *Virtual* oracles [2] are polynomials whose evaluations can be efficiently computed from concrete oracle evaluations. Formally, let $f_1 \dots f_n \in \mathbb{F}^{(<B)}[X]$ be concrete oracles. A virtual oracle $F \in \mathbb{F}^{(<D)}[X]$ has the form:

$$F(X) := G(X, h_1(v_1(X)), \dots h_m(v_m(X))$$

where for $i \in [m]$, $h_i \in \{f_1, \dots f_n\}$, $v_i \in \mathbb{F}^{(<b)}[X], G \in \mathbb{F}[X, X_1 \dots X_m]$, and $G, \{v_i\}_i$ are public. Particularly, we are interested in the cases were $D$ is $\mathsf{negl}(\lambda)$ and $m, b, \deg(G)$, and the monomial count of $G$ are constants.

Oracles in secure $\mathsf{polyIOP}$ can be swapped with virtual oracles, as the underlying soundness depends only on the evaluations.

**Compilation** A polynomial commitment scheme can be leveraged to compile a $\mathsf{polyIOP}$ into a standard protocol. Whenever $\mathcal{P}$ sends an oracle to $\mathcal{V}$, simply have the prover sent a commitment to the polynomial. Then when $\mathcal{V}$ sends an evaluation query, $\mathcal{P}$ simply responds with the evaluation along with an evaluation proof. We use this method in our construction of a revealable functional commitment scheme

## 2.5 Algebraic Holographic Proofs

**Index Relation** Let $\mathcal{I}$ be an index space, let $\mathcal{X}$ be an instance space, and let $\mathcal{W}$ be a witness space. We call $\mathcal{R} \subseteq \mathcal{I} \times \mathcal{X} \times \mathcal{W}$ an *index relation.* Here, an index $i \in I$ explicitly represents a binary relation $\mathcal{B} \subseteq \mathcal{X} \times \mathcal{W}$ where $x \in \mathcal{X}, w \in \mathcal{W}$

represent public and witness inputs, respectively. Again, let $\mathcal{R}_N$ denote the restriction of $\mathcal{R}$ to indices $|i| \leq N \in \mathbb{N}$.

We can connect the concepts of index relations and oracle relations with an *algebraic hologoraphic proof*, or AHP, using our previous definition of $\mathsf{polyIOPs}$.

**Definition 2 (Algebraic Holographic Proof).** *A (constant round) algebraic holographic proof [5], or AHP, for an index relation $\mathcal{R} \subseteq \mathcal{I} \times \mathcal{X} \times \mathcal{W}$ is a tuple:*

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

- $\mathsf{Enc}_{\mathsf{AHP}}$ *deterministically maps an index $i \in \mathcal{I}$ to an encoded index $\vec{o} \subset \mathcal{O}$*
- $(\mathcal{P}_{\mathsf{AHP}}, \mathcal{V}_{\mathsf{AHP}}, k, s, d)$ *is a* $\mathsf{polyIOP}$ *for the oracle relation:*

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

.

**Index-Private AHPs** In our construction of revealable functional commitments, we would like AHPs that are zero-knowledge for the index as well as the witness. [4] captures this property with the following definition:

**Definition 3 (Index-private AHP).** *Let* $\mathsf{AHP}$ *be an AHP with prover* $\mathcal{P}_{\mathsf{AHP}}$ *and verifier* $\mathcal{V}_{\mathsf{AHP}}$*/ For field $\mathbb{F}$ and $(i, x, w) \in \mathcal{R}$, let*

$$View(\langle \mathcal{P}_{\mathsf{AHP}}(\mathbb{F}, i, x, w), \mathcal{V}_{\mathsf{AHP}}^{\mathsf{Enc}_{\mathsf{AHP}}(i)}(x)\rangle)$$

*be the view of* $\mathcal{V}_{\mathsf{AHP}}$. $\mathsf{AHP}$ *is* ***index-private*** *if it is complete, knowledge-sound [5], and there exists a* $\mathsf{PPT}$ *simulator $S$ such that for all $(i, x, w) \in \mathcal{R}$ and field $\mathbb{F}$, $S(F, x, 1^{|i|})$ is indistinguishable from the view of* $\mathcal{V}_{\mathsf{AHP}}$

## 2.6 Function-Hiding Functional Commitments

Our work directly builds off the work of Boneh, Nguyen, and Ozdemir on function hiding functional commitment schemes (sometimes shortened to just "functional commitment") [4]. While we will give a brief self-contained overview of their contributions in this section, we highly recommend reading their work first to contextualize our contributions.

In a function hiding functional commitment scheme, a *committer* commits to a secret function $f : \mathcal{X} \to \mathcal{Y}$ using a succinct hiding and binding commitment scheme. Later, they can prove at any public point the evaluation of $f$, while divulging no other information about the function.

More concretely a function hiding functional commitment scheme (which we will abbreviate as simply functional commitment scheme), is a triple $\{\mathsf{Setup}, \mathsf{Commit}, \mathsf{Eval}\}$. $\mathsf{Setup}(1^\lambda, N)$ is a randomized algorithm that outputs public parameters $\mathsf{pp}$ to support commitments of complexity $N$. $\mathsf{Commit}(\mathsf{pp}, f, r)$ is a deterministic algorithm that takes a description of $f \in \mathcal{F}$ and randomness

$r$, and outputs a hiding and binding commitment $c$ to $f$. $\mathsf{Eval}$ is an interactive protocol between a prover $\mathcal{P}_{\mathsf{Eval}}(\mathsf{pp}, f, r, x, y)$ and a verifier $\mathcal{V}_{\mathsf{Eval}}(\mathsf{pp}, c, x, y)$ that convinces the verifier that $f(x) = y$, and that $f$ is indeed a function (this latter part is done through a proof of function relation, defined below).

Boneh, Nguyen, and Ozdemir [4] show that using an index-private AHP and an associated *proof of function relation* (PFR), a functional commitment scheme can be constructed that is (1) complete, (2) zero knowledge, and (3) is an argument of knowledge for a function $f \in \mathcal{F}$ (we will define and rely on these properties for our own construction). Additionally, they design efficient PFRs for the commonly used Marlin [5] and Plonk [7] preprocessing zk-SNARKs, thereby achieving functional commitment schemes for both. Our work will be largely additive to these schemes, maintaining their construction while adding new functionality.

**Proof of Function Relation (PFR)** AHP's are designed for general index relations, which are not necessarily functions. Therefore, the crux of functional commitments lies in the ability to prove that a commitment is indeed to a function, without leaking any other information about the function. This is accomplished using a $\mathsf{polyIOP}$ known as a *Proof of Function Relation*, or PFR. To formalize this, we first need to translate the notion of functions into index relations with *functional sets*:

**Definition 4 (Functional Sets for Index Relations).** *Let $\mathcal{R} \subseteq I \times (\mathcal{X} \times \mathcal{Y}) \times \mathcal{W}$. A subset $\mathcal{I}_f \subseteq \mathcal{I}$ is a* ***functional set*** *if for all $i \in \mathcal{I}_f$, and for all $x \in \mathcal{X}$, there exists a unique $y \in \mathcal{Y}$ such that there is a $w \in \mathcal{W}$ such that $(i, (x, y), w) \in \mathcal{R}$. In other words, i's residual relation is a function. Furthermore, $\mathcal{I}_f$ must have a poly-time algorithm $\mathsf{Extend}$ such that for all $(i, x) \in \mathcal{I}_f \times \mathcal{X}$, $\mathsf{Extend}(i, x) \to (y, w)$ where $(i, (x, y), w) \in \mathcal{R}$.*

So, the task of a PFR becomes to prove that a given $\vec{o}$ is an encoding of an $i$ such that $i \in I_f$.

**Definition 5 (Proof of Function Relation).** *Let an $\mathsf{AHP}$ have an encoded index space $\mathcal{O} = (\mathbb{F}[X])^c$ and encoding function $\mathsf{Enc}_{\mathsf{AHP}} : \mathcal{I} \to \mathcal{O}$. Let $\Pi$ be a $\mathsf{polyIOP}$ between $\mathcal{P}_f$ and $\mathcal{V}_f$ for the following oracle relation:*

$$\mathcal{R}_{\mathsf{func}} = \{(\vec{o} \in \mathcal{O}, \perp, i) : i \in \mathcal{I}_f \wedge \vec{o} = \mathsf{Enc}_{\mathsf{AHP}}(i)$$

*We call $\Pi$ a* ***proof of function relation*** *if it is a secure $\mathsf{polyIOP}$ for $\mathcal{R}_{\mathsf{func}}$*

## 2.7 Rank-1 Constraint Systems (R1CS)

For $n \in \mathbb{N}$ constraints and $h \leq n$ instance variables, the **rank-1 constraint system** ($R1CS$) index relation is:

$$\mathcal{R}_{\mathsf{R1CS}}(n, h) := \left\{ \big((A, B, C) \in (\mathbb{F}^{n \times n})^3, x \in \mathbb{F}^h; w \in \mathbb{F}^{n-h}\big) \,\middle|\, \begin{array}{r} z := (x, w) \\ (Az) \circ (Bz) = Cz \end{array} \right\}$$

Where $\circ$ is an elementwise multiplication of two equal size vectors.

### 2.8 Arithmetic Circuits

Intuitively, an arithmetic circuit a directed acyclic graph where nodes represent gates and edges represent wires. Wires carry values in $\mathbb{F}$, and gates are binary - they add or multiply.

Formally, an arithmetic circuit $C$ with $n_i$ inputs, $n_g$ gates, and $n_o \leq n_g$ outputs is a sequence of gate tuples $(l_i, r_i, s_i)_{i \in [n_g]} \in ([n_i + n_g] \times [n_i + n_g] \times \{+, \times\})^{n_g}$, constrained by $l_i, r_i \leq i + n_i$. For gate $i$, we will call $l_i, r_i$ the left and right input wire indices, $i + n_i$ as the output wire index, and $s_i$ as the gate sign. The set of circuit input wire indices is $[n_i]$. Let $\mathcal{AC}_{n_i,n_g,n_o}$ denote the set of arithmetic circuits with $n_i$ inputs, $n_g$ gates, and $n_o$ outputs.

To evaluate $C$ on input $(x_1 \ldots x_{n_i}) \in \mathbb{F}^{n_i}$, simply compute the $n_g + n_i$ wire values, $w_1 \ldots w_{n_g+n_i}$ in order. The first $w_1 \ldots w_{n_i}$ are simply the input values. Then, each $j \in [n_g]$, $w_{j+n_i} = w_{l_j} + w_{r_j}$ if $s(j) = +$, and $w_{l_j} \times w_{r_j}$ otherwise. Then, the final $n_o$ wire values are the circuit output. By evaluating, $C$ defines a function from $\vec{x} \in \mathbb{F}^{n_i}$ to $\vec{y} \in \mathbb{F}^{n_o}$, which we denote as $\vec{y} = C(\vec{x})$

## 3 Valid Reveals

Before we begin defining revealable functional commitments, we first give some intuition of what revealing a function actually means. Functional commitments work in *encoded function spaces* [4], which are defined as follows:

**Definition 6 (Encoded Function Space).** *Let $\mathcal{X}_\lambda$ and $\mathcal{Y}_\lambda$ be input and output spaces. An* ***encoded function space*** *is a finite set $\mathcal{F}_\lambda$ of poly-$\lambda$ length strings equipped with deterministic evaluation algorithm* $\mathsf{Evaluate}_\lambda : \mathcal{F}_\lambda \times \mathcal{X}_\lambda \to \mathcal{Y}_\lambda$. *Note that each $f \in \mathcal{F}_\lambda$ must encode a function from $\mathcal{X}_\lambda$ to $\mathcal{Y}_\lambda$. Generally, $\lambda$ subscripts are omitted when unambiguous.*

### 3.1 Constraint Consistency

Importantly, we are working with functions of fixed-length descriptions, with fixed and finite input and output lengths. This means that we can interpret $f \in \mathcal{F}$ as a set of constraints on a $\mathsf{poly}(\lambda)$ length input. In fact common encodings, such as arithmetic circuits and R1CS instances, are already formatted in this way. This leads to a rather convenient way to reveal partial information about an encoded function - we can simply publish a set of constraints, and prove that it is a subset of the secret function's constraints. We refer to this property as *consistency.*

### 3.2 Lock Checking

There is another consideration we need to make if we want our protocol to have meaningful functionality. Suppose the prover's function is handling sensitive data, and the verifier would like to be convinced that a particular value $x_1$ is hashed first in the function. In this case, it is not enough in this case that the prover sends over a set of constraints that correspond to $x_1$'s hashing. If there are additional constraints outside of those that were revealed, the function could easily reproduce $x_1$'s true value elsewhere.So, the prover needs to have the ability to convince the verifier that the published set contains *all the constraints* on $x_1$. Put simply, a prover needs to prove they are being transparent about the operations performed on $x_1$. To capture this, we introduce the following vocabulary:

**Definition 7 (Locked).** *Let $U$ be a set of constraints on $x$ and let $S \subseteq U$. A variable $x_1 \in x$ is considered to be **locked** by $S$ if $x_1$ is constrained by $S$ AND there is no constraint in $U \setminus S$ that constrains $x_1$.*

So, our protocol needs to also ensure that all values the revealer claims are locked are indeed locked.

**Definition 8 (Valid Reveal, Revealable Function Encoding).** *For function encoding $f$, let $U_f$ be the exact set of constraints that $f$ imposes on $\mathcal{X}$, and let $S$ be a set of constraints on $\mathcal{X}$. Let $L$ be an encoded set of variables in $x_1, \ldots x_m$. We consider $q = (S, L) \in \mathcal{Q}$ to be a **valid reveal** of $f$ if:*

- *Consistency: $S \subseteq U_f$*
- *Lockedness: $\forall l \in L$, $l$ is locked by $S$*

*Furthermore, we consider $\mathcal{F}$ to be a **revealable function encoding** if for $f \in \mathcal{F}$, there is a poly-time decider algorithm $\mathsf{VR}(f, q) \to \{0, 1\}$ that accepts when $q$ is a valid reveal, and rejects otherwise.*

As a note, $L$ need not contain all locked values to qualify as a valid reveal. This choice is intentional, as it lets the revealer decide the amount of information to disclose.

## 4 Revealable Functional Commitments

Now we can define a revealable functional commitment scheme. Like a standard functional commitment scheme, a revealable functional commitment scheme allows the committer to commit to a secret function and then prove evaluations of that function. In addition, the committer can also prove that a published set of constraints and locks is a valid reveal. First, we can modify our definition of encoded function spaces to fit this new problem.

Let $\{\mathcal{X}_\lambda\}_{\lambda \in \mathbb{N}}$ and $\{\mathcal{Y}_\lambda\}_{\lambda \in \mathbb{N}}$ be families of input and output spaces, and let $\{\mathcal{Q}_\lambda\}_{\lambda \in \mathbb{N}}$ be a **reveal space**. Let $\{\mathcal{F}_\lambda, \mathsf{Evaluate}_\lambda, \mathsf{VR}_\lambda\}\lambda$ be a family of encoded revealable function spaces. An tuple $\{\mathcal{F}_\lambda, \mathsf{Evaluate}_\lambda, \mathsf{VR}_\lambda\}$ is an **encoded revealable function space** if $\{\mathcal{F}_\lambda, \mathsf{Evaluate}_\lambda\}$ is an encoded function space (defined in preliminaries) and $\mathcal{F}_\lambda$ is a revealable function with decider algorithm $\mathsf{VR}_\lambda : \mathcal{F}_\lambda \times \mathcal{Q}_\lambda \to \{0, 1\}$. We will omit $\lambda$ when it is clear from context.

Arithmetic circuits are an illustrative example of a revealable encoded function space. Input and output spaces are $\mathcal{X} = \mathbb{F}^{n_i}$ and $\mathcal{Y} = \mathbb{F}^{n_o}$. The encoded function space consists of $\mathsf{poly}(\lambda)$ sized directed acyclic graphs of additions and multiplications. $\mathsf{Evaluate}(f, x \in X)$ evaluates the circuit represented by the graph. The reveal space would be $\mathsf{poly}(\lambda)$ sized acyclic graphs paired with an encoding of locked wire values. $\mathsf{VR}(f, q)$ outputs 1 if and only if the revealed acyclic graph $q$ is a subcircuit of $f$, and if all locked wire values do not leave the subcircuit.
A revealable functional commitment scheme $\mathsf{RFC}$ for $\mathcal{F}$ is a tuple $(\mathsf{Setup}, \mathsf{Commit}, \mathsf{Eval}, \mathsf{Reveal})$ where:

- $\mathsf{Setup}(1^\lambda, N) \rightarrow \mathsf{pp}$: Given the security parameter and max size, randomly sample public parameters.
- $\mathsf{Commit}(\mathsf{pp}, f \in \mathcal{F}, r \in \mathsf{R}) \rightarrow c \in \mathcal{C}$: Given $\mathsf{pp}$, an encoded function $f$, and randomness $r$, deterministically produce a commitment $c$ of $f$.
- $\mathsf{Eval}(\mathcal{P}_E(\mathsf{pp}, f \in \mathcal{F}, r \in \mathsf{R}, x \in \mathcal{X}, y \in \mathcal{Y}), \mathcal{V}_E(\mathsf{pp}, c, x, y)) \rightarrow \{0, 1\}$: An interactive protocol for $\mathcal{P}_E$ to convince $\mathcal{V}_E$ that $f(x) = y$.
- $\mathsf{Reveal}(\mathcal{P}_R(\mathsf{pp}, f \in \mathcal{F}, r \in \mathsf{R}, q \in \mathcal{Q}), \mathcal{V}_R(\mathsf{pp}, c, q) \rightarrow \{0, 1\}$: An interactive protocol for $\mathcal{P}_R$ to convince $\mathcal{V}_R$ that $\mathsf{VR}(f, q) = 1$.

Here, we call $\mathsf{R}$ the *randomness space* and $\mathcal{C}$ the *commitment space*. A secure revealable functional commitment scheme should have the following properties:

- *Binding:* Computing distinct function encodings with the same commitments is infeasible
- *Hiding:* Commitments to distinct function encodings are indistinguishable
- *Completeness:* Correct evaluation proofs and reveal proofs are always accepted
- *Reveal Soundness:* Incorrect reveal proofs are accepted with negligible probability
- *Protocol zero-knowledge:* An evaluation proof divulges nothing other than the evaluation and a reveal proof divulges nothing other than the validity of the reveal
- *Knowledge soundness:* Evaluation proofs show that $\mathcal{P}_E$ knows a function encoding consistent with the evaluation and the commitment. Similarly, reveal proofs show that $\mathcal{P}_R$ knows a function encoding consistent with reveal and encoding
- *Evaluation Binding:* A malicious prover cannot construct conflicting evaluation proofs on the same input.

The hiding and binding requirements are exactly those for classical commitment schemes. Helpfully, evaluation/reveal binding is implied by binding and extractability, and can thus be omitted from the formal definition (see Supplement $A$ of [4]). We compile these properties into the following definition:

**Definition 9 (Secure Revealable Functional Commitment).** *A revealable functional commitment scheme is* ***secure*** *is it has the following properties:*

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

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

*And* $\mathsf{Reveal}$ *is a complete protocol for the following relation:*

$$\mathcal{R}_{\mathsf{reveal}}(\mathsf{pp}) = \{(c, q; f, r) : f \in \mathcal{F} \wedge \mathsf{VR}(f, q) = 1 \wedge c = \mathsf{Commit}(\mathsf{pp}, f, r)\}$$

– ***Reveal Soundness:*** $\mathsf{Reveal}$ *is a sound protocol for* $\mathcal{R}_{\mathsf{reveal}}(\mathsf{pp})$
– ***Protocol Extractablity:*** $\mathsf{Eval}$ *is an argument of knowledge for* $\mathcal{R}_{\mathsf{eval}}(\mathsf{pp})$ *and* $\mathsf{Reveal}$ *is an argument of knowledge for* $\mathcal{R}_{\mathsf{reveal}}(\mathsf{pp})$
– ***Protocol Honest Verifier Zero Knowledge:*** $\mathsf{Eval}$ ***is an honest verifier zero knowledge protocol for*** $\mathcal{R}_{\mathsf{eval}}$ ***and*** $\mathsf{Reveal}$ *is an honest verifier zero knowledge protocol for* $\mathcal{R}_{\mathsf{reveal}}$

## 5 AHP-based Construction

Our task now is to construct a revealable functional commitment scheme for a revealable encoded function space. Recall our preliminary conversation of functional sets. Let's augment this definition slightly:

**Definition 10 (Revealable Functional Set).** *A functional set* $\mathcal{I}_f$ *of an index relation* $\mathcal{R} \subset \mathcal{I} \times (\mathcal{X} \times \mathcal{Y}) \times \mathcal{W}$ *is considered a* ***revealable functional set*** *for* $\mathcal{R}$ *with reveal space* $Q$ *if* $\mathcal{I}_f$ *is a revealable function encoding. That is, there exists a decider algorithm* $\mathsf{VR}$ *such that for all* $i \in I_f, q \in \mathcal{Q}$*,* $\mathsf{VR}(i, q)$ *accepts if* $q$ *is a valid reveal of the residual* $\mathcal{X} \times \mathcal{Y}$ *function of* $i$*.*

A functional set $\mathcal{I}_f$ of an index relation $\mathcal{R} \subset \mathcal{I} \times (\mathcal{X} \times \mathcal{Y}) \times \mathcal{W}$, can naturally be viewed as a revealable encoded function space, with the following algorithms:

– $\mathsf{Evaluate}(i \in \mathcal{I}_f, x \in \mathcal{X})$ will compute $(y, w) \leftarrow \mathsf{Extend}(i, x)$ and output $y$. Since $y$ is unique, the output is deterministic.
– $\mathsf{Reveal}(i \in \mathcal{I}_f, q \in \mathcal{Q})$ will output a decision based on if $\mathsf{VR}(i, q) = 1$. A reveal is always either valid or invalid, so the output is deterministic.

Additionally, since all revealable functional sets are functional sets, we note that the notion of a PFR on a revealable functional set is well-defined.

### 5.1 Proof of Reveal

Throughout this paper, we will use the compilation property of $\mathsf{polyIOPs}$ to construct our protocols. Our focus, the $\mathsf{Reveal}$ protocol, will hence boil down a *proof of reveal*, defined as follows:

**Definition 11 (Proof of Reveal).** *Let an* $\mathsf{AHP}$ *have an encoded index space* $\mathcal{O} = (\mathbb{F}[X])^c$ *and encoding function* $\mathsf{Enc}_{\mathsf{AHP}} : \mathcal{I} \to \mathcal{O}$. *Let* $\mathcal{I}_f \subset \mathcal{I}$ *be a revealable functional set. Let* $\Pi$ *be a* $\mathsf{polyIOP}$ *between* $\mathcal{P}_r$ *and* $\mathcal{V}_r$ *for the following oracle relation:*

$$\mathcal{R}_{\mathsf{rev}} = \{(\vec{o} \in \mathcal{O}, q \in \mathcal{Q}, i \in \mathcal{I}_f) : \mathsf{VR}(i, q) = 1 \land \vec{o} = \mathsf{Enc}_{\mathsf{AHP}}(i)\}$$

*We call* $\Pi$ *a* ***proof of reveal*** *for* $I_f$ *if it is a secure* $\mathsf{polyIOP}$ *for* $\mathcal{R}_{\mathsf{rev}}$

Since encoded revealable function spaces and revealable functional sets are both subsets of their non-revealable counterparts, the $\mathsf{Setup}, \mathsf{Commit}, \mathsf{Eval}$ protocols can be used as-is from standard functional commitments [4] (we include them in the construction for completeness). The $\mathsf{Commit}$ algorithm uses a hiding polynomial commitment scheme to commit to the polynomials that encode the index. The $\mathsf{Eval}$ protocol uses the PFR (to ensure that the committed index is to a function), followed by the AHP to prove that the evaluation and witness are consistent with the committed index. Finally, $\mathsf{Reveal}$ protocol again uses the PFR, along with the proof of reveal to prove that the published reveal is valid.

**Construction 1 (Revealable Functional Commitment Scheme)**
Let $\mathcal{R} \subset \mathcal{I} \times (\mathcal{X} \times \mathcal{Y}) \times \mathcal{W}$ be the evaluation index relation. Let $\mathsf{AHP} = (\mathsf{Enc}_{\mathsf{AHP}}, \mathcal{P}_{\mathsf{AHP}}, \mathcal{V}_{\mathsf{AHP}}, k_{\mathsf{AHP}}, s_{\mathsf{AHP}}, d_{\mathsf{AHP}})$ be an AHP $\mathcal{R}_e$. Let $\Pi_f = (\mathcal{P}_f, \mathcal{V}_f, k_f, s_f, d_f)$ be a PFR for revealable functional set $\{\mathcal{I}_f \subset \mathcal{I}, \mathsf{Extend}, \mathsf{VR}\}$. Be the reveal index relation and let $\Pi_r = (\mathcal{P}_r, \mathcal{V}_r, k_r, s_r, d_r)$ be a proof of reveal for $\mathcal{I}_f$ such that $s_r(0) = s_f(0) = s_{\mathsf{AHP}}(0)$, and $\forall i \in [s_{\mathsf{AHP}}(0)]$. $d_{\mathsf{AHP}}(N, 0, i) = d_r(N, 0, i) = d_f(N, 0, i)$, where $N$ is the maximum supported index size.

Additionally, let $\mathsf{PC} = (\mathsf{PC.Setup}, \mathsf{PC.Commit}, \mathsf{PC.Eval}, \mathsf{PC.Check})$ be a polynomial commitment scheme.

We construct $\mathsf{RFC}_{\mathsf{AHP}, \mathcal{I}_f, \Pi_r, \Pi_f, \mathsf{PC}}$ as the following tuple:

– $\mathsf{Setup}(1^\lambda, N)$: Compute $D$, the union of all degree bounds, as

$$D = \bigcup_{\mathsf{piop} \in \{f, r, \mathsf{AHP}\}} \{d_{\mathsf{piop}}(N, i, j)\}_{i \in [k_{\mathsf{piop}}], j \in [s_{\mathsf{piop}}(i)]}$$

and return $\mathsf{pp} \leftarrow \mathsf{PC.Setup}(1^\lambda, D)$

– $\mathsf{Commit}(\mathsf{pp}, i \in \mathcal{I}_f, r \in \mathsf{R})$ :
  - Parse out $(r_{o_{0,1}}, \ldots r_{o_{0,s_{\mathsf{AHP}}(0)}}) = r$ and $(\mathsf{ck}, \mathsf{vk}) = \mathsf{pp}$
  - Set $\vec{o} \leftarrow \mathsf{Enc}_{\mathsf{AHP}}(i)$
  - $\forall i \in [s_{\mathsf{AHP}}(0)]$, set $c_{o_{0,i}} \leftarrow \mathsf{PC.Commit}(\mathsf{ck}, \vec{o}[i], d_{\mathsf{AHP}}(N, 0, i), r_{o_{0,i}}$
  - Return $c \leftarrow (c_{o_{0,i}}, \ldots c_{o_{0,s_{\mathsf{AHP}}(0)}})$

– $\mathsf{Eval}(\mathcal{P}_E(\mathsf{pp}, i, r, x, y), \mathcal{V}_E(\mathsf{pp}, c, x, y))$ :
  - $\mathcal{P}_E$ and $\mathcal{V}_E$ both parse $(\mathsf{ck}, \mathsf{vk}) = \mathsf{pp}$
  - $\mathcal{P}_E$ computes $\vec{o} \leftarrow \mathsf{Enc}_{\mathsf{AHP}}(i)$; $(y, w) \leftarrow \mathsf{Extend}(i, x)$; and aborts if $y \neq y'$
  - $\mathcal{P}_E$ and $\mathcal{V}_E$ run the following polynomial IOPs, using the **compilation method** (outlined below):
    * Proof of function relation: $\langle \mathcal{P}_f(\vec{o}, \perp, i), \mathcal{V}_f^{\vec{o}}(\perp) \rangle$
    * Proof for $\mathcal{R}$ (evaluation): $\langle \mathcal{P}_{\mathsf{AHP}}(i, (x, y), w), \mathcal{V}_{\mathsf{AHP}}^{\vec{o}}((x, y)) \rangle$

– $\mathsf{Reveal}(\mathcal{P}_R(\mathsf{pp}, i, r, q \in Q), \mathcal{V}_E(\mathsf{pp}, c, q))$ :
  - $\mathcal{P}_E$ and $\mathcal{V}_E$ both parse $(\mathsf{ck}, \mathsf{vk}) = \mathsf{pp}$
  - $\mathcal{P}_E$ computes $b \leftarrow \mathsf{VR}(i, q)$ and aborts if $b \neq 1$
  - $\mathcal{P}_E$ and $\mathcal{V}_E$ run the following polynomial IOPs, using the compilation method:
    * Proof of function relation: $\langle \mathcal{P}_f(\vec{o}, \perp, i), \mathcal{V}_f^{\vec{o}}(\perp) \rangle$
    * Proof of reveal: $\langle \mathcal{P}_r(\vec{o}, q, i), \mathcal{V}_r^{\vec{o}}(q) \rangle$

**Compilation Method:** Let $\mathsf{piop} \in \{\mathsf{AHP}, f, r\}$. $\mathcal{P}_\alpha$ and $\mathcal{V}_\alpha$ use the following method to "run" $\mathsf{piop}$:

- $\forall i \in [k_{\mathsf{piop}}], j \in [s_{\mathsf{piop}}(i)$, when $\mathcal{P}_{\mathsf{piop}}$ sends oracle polynomial $o_{i,j}$, $\mathcal{P}_\alpha$ computes and sends the following commitment to $\mathcal{V}_\alpha$:

$$c_{o_{i,j}} \leftarrow \mathsf{PC.Commit}(\mathsf{ck}, o_{i,j}, d_{\mathsf{piop}}(N, i, j), r_{o_{i,j}} \xleftarrow{\$} R$$

- When $\mathcal{V}_{\mathsf{piop}}$ derives an oracle $o$ that is a linear combination of other oracles it holds, $\mathcal{V}_\alpha$ derives commitment $c_o$ and $\mathcal{P}_\alpha$ derives commitment randomness $r_o$ via PCS homomorphism.
- When $\mathcal{V}_{\mathsf{piop}}$ queries any polynomial oracle $o$ with degree bound $d_o$ at $z \in \mathbb{F}$:
  - $\mathcal{V}_\alpha$ sends $z$ to $\mathcal{P}_\alpha$
  - $\mathcal{P}_\alpha$ obtains/derives $r_o$, and sends $(\pi \leftarrow \mathsf{PC.Eval}(\mathsf{ck}, o, d_o, r_o, z), y \leftarrow o(z))$ to $\mathcal{V}_\alpha$
  - $\mathcal{V}_\alpha$ obtains/derives $c_o$ and asserts $\mathsf{PC.Check}(\mathsf{vk}, c_o, d_o, z, y, \pi) = 1$

**Theorem 1.** *Let* $\mathsf{AHP}$ *be an index-private algebraic holographic proof with injective* $\mathsf{Enc}_{\mathsf{AHP}}$*, let* $\mathit{\Pi}_f$ *be a secure PFR for functional set* $\mathcal{I}_f$*, let* $\mathit{\Pi}_r$ *be a secure proof of reveal for* $\mathcal{I}_f$*, and let* $\mathsf{PC}$ *be a perfectly hiding PCS with knowledge-soundness and HVZK evaluation.* $\mathsf{RFC}_{\mathsf{AHP},\mathcal{I}_f,\mathit{\Pi}_r,\mathit{\Pi}_f,\mathsf{PC}}$ *is a secure revealable functional commitment scheme for function encodings* $i \in \mathcal{I}_f$

As previously stated, this construction is almost identical to that of theorem 2 in [4], with the addition of the Reveal method. We encourage the reader to take a look at their work for a full proof. Here, we will provide a proof sketch, and argue why their proof implies the result above.

*Proof Sketch:* The tuple $(\mathsf{Setup}, \mathsf{Commit})$ is binding and hiding as $\mathsf{Enc}_{\mathsf{AHP}}$ is injective, and $\mathsf{PC.Commit}$ is both hiding and binding.

The completeness of the protocol follows from the completeness of the subprotocols.

The reveal soundness follows from the security of $\mathit{\Pi}_r$

[4]'s argument for the evaluation extractability is as follows. An extractor $\mathcal{E}_E$ for $\mathcal{R}_{\mathsf{eval}}$ uses the PFR extractor $\mathcal{E}_f$, the AHP extractor $\mathcal{E}_{\mathsf{AHP}}$, and the PCS extractor $\mathcal{E}_{\mathsf{PC}}$. Throughout the extraction, $\mathcal{E}_E$ uses $\mathcal{E}_{\mathsf{PC}}$ to extract polynomials and commitment randomness for adversary $A$'s commitments and oracle evaluations. From these polynomials, it extracts $i \in \mathcal{I}_f$ using $\mathcal{E}_f$. It then uses $\mathcal{E}_{\mathsf{AHP}}$ to extract $w \in \mathcal{W}$. If everything succeeds, then $\mathcal{E}_E$ knows (1) $f = i \in \mathcal{I}_f$, (2) the randomness involved in $i$'s commitment (meaning the commitment to $f$ can be freshly computed and compared to $c$), (3) $w$ such that $(i, (x, y)w) \in \mathcal{R}$, implying that $f(x) = y$ - exactly the conditions it looks to satisfy. Using a similar approach, we can have an extractor $\mathcal{E}_R$ for $\mathcal{R}_{\mathsf{reveal}}$ find that $f = i \in \mathcal{I}_f$, recompute and compare a commitment to $f$, and then verify that $\mathsf{VR}(i, q) = 1$

Finally, an HVZK simulator can be built. Simply use simulators for $\mathit{\Pi}_r, \mathit{\Pi}_e, \mathsf{AHP}$ to simulate all oracle queries and evaluations. Here, the index-privacy of AHP is crucial, as it must not divulge $i$. Using the PCS evaluation simulator, we can also simulate all sent evaluation proofs. □

With this theorem in hand, the task of instantiating a revealable functional commitment is clear: create a proof of reveal for an index-private AHP equipped with a proof of function relation.

## 6 Revealable Functional Commitments from Marlin

Marlin [5] is an AHP for Rank-1 constraint systems. Previously [4], it was used to construct a functional commitment scheme, which involved constructing a proof of function relation. First, we will introduce the arithmetization that Marlin uses to translate matrices into polynomials. Then, we will give a revealable functional set for R1CS, and observe that any arithmetic circuit can be compiled into this set. Finally, we will give a proof of reveal for Marlin, thereby achieving a revealable functional commitment scheme.

We would like to point out that the key property of a selected AHP, index-privacy, is required in the construction of a revealable functional commitment scheme. [4] makes a minor modification to the protocol to achieve this. For the purposes of this paper, "Marlin" will refer to the version of the algorithm with the desired properties.

### 6.1 Arithmetization

Recall from preliminaries that an R1CS instance for $n$ constraints and $\leq n$ variables is defined by three matrices $A, B, C \in \mathbb{F}^{n \times n}$. These matrices can be encoded using two cyclic subgroups of $\mathbb{F}$. Let $\mathbb{H} = \langle h \rangle$ be a subsgroup of $\mathbb{F}$ of order $n$. A $1 \times n$ node vector $z$ (which incodes input, witness, and output variables), is said to satisfy the R1CS instance if $Az \circ Bz = Cz$. Intuitively, for $i \in [n]$, the rows $A_i, B_i, C_i$ together form a constraint on $z$. Let there be at most $m$ pairs of indices $(r, c)$ such that at least one of $A_{rc}, B_{rc}, C_{rc}$ is non-zero, where $m$ is the order of cyclic subgroup $\mathbb{K} = \langle k \rangle$ of $\mathbb{F}$. Let's order these pairs in a list $(r_i, c_i)_{i=0}^{m-1}$ (with the excess list elements being 0 if there are less than $m$ pairs. Now, we can encode the R1CS matrices with the following 5 polynomials $\mathbb{K} \to \mathbb{H} \cup \{0\}$ of degree less than $|\mathbb{K}|$:

$$\text{row}(k^i) = r_i \quad \text{col}(k^i) = col_i$$

$$\text{val}^A(k^i) = A_{r_i c_i} \quad \text{val}^B(k^i) = B_{r_i c_i} \quad \text{val}^C(k^i) = C_{r_i c_i}$$

Note that this arithmetization differs slightly from the canonical one in [5]. Typically, each matrix gets its own row, col, and val polynomials, totaling to 9 polynomials. However in practice [6], the encoding above is used. Our protocols can easily be reverted to the canonical encoding, but we chose to optimize for efficiency and succinctness.

### 6.2 A Revealable Function Set for R1CS

We unfortunately cannot make any sort of function commitment for a general R1CS instance, due to the fact that not all instances encode deterministic functions. To build a revealable functional commitment, we must first find a revealable functional set. The modification is simple: move the output variables to the end of the vector $z$[4].

**Definition 12 (Output-Final R1CS).** *For $n, t, s \in \mathbb{N}$ where $s, s+r \in [n]$, let $\mathcal{I} = (F^{n\times n})^3, \mathcal{X} = \mathbb{F}^t, \mathcal{Y} = \mathbb{F}^s, \mathsf{X} = \mathcal{X} \times \mathcal{Y}, \mathcal{W} = \mathbb{F}^{n-t-s}$. The index relation $\mathcal{R}_{\mathsf{R1CS-f}}(n,t,s) \subseteq \mathcal{I} \times \mathsf{X} \times \mathcal{W}$ is:*

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

[4] accompany this definition with a natural functional set: $t$-FT. For indices in $t$-FT, each element in $z$ beyond the input is completely determined by the previous elements.

**Definition 13 (Functional Triple ($t$-FT)).** *Let $n, t \in \mathbb{N}$ such that $t \in [n]$. Matrix $M \in \mathbb{F}^{n\times n}$ is considered $t$-**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 non-zero. Let $t$-**Diag** $\subset \mathbb{F}^{n\times n}$ be the set of such matrices.*

*$M \in \mathbb{F}^{n\times n}$ is considered $t$-**strictly lower triangular** if and only if $M$ is strictly lower triangular with the first $t$ rows equal to zero. Let $t$-**SLT** $\subset \mathbb{F}^{n\times n}$ be the set of such matrices.*

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

We can take this one step further, showing that $t$-FT is actually a revealable functional set.

**Theorem 2.** *$t$-FT is a revealable functional set for $\mathcal{R}_{\mathsf{R1CS-f}}$ with reveal space $\mathcal{Q}_{\mathsf{Marlin}} = \left((\mathbb{F}^n)^3\right)^* \times ([n]^2)^*$*

*Proof.* [4] already show that $t$-FT is a functional set for $\mathcal{R}_{\mathsf{R1CS-f}}$. So what is left to show is that it is a revealable function encoding. As we discussed previously, for $i \in [n]$, the rows $(A_i, B_i, C_i) \in (\mathbb{F}^n)^3)$ represent a constraint on $z$. So, the exact set of constraints that an instance imposes on $z$ is $U = \{(A_i, B_i, C_i)\}_{i\in[n]}$.

Suppose $S \subset U$, and consider the i'th element of $z$. For $z[i]$ to be locked by $S$, the following need to be true:

- $S$ constrains $z[i]$: There exists $(A_j, B_j, C_j) = s \in S$ where at least one row $M_j \in s$ has a non-zero element at $M_j[i]$.
- $U \setminus S$ does not constrain $z[i]$: For all $(A_k, B_k, C_k) = u \in U \setminus s$, and for all $M_k \in u$, $M_k[i]$ is zero

So, for $q = (S, L) \in \mathcal{Q}_{\mathsf{Marlin}}$, it is easy to check if $q$ is a valid reveal of $f = (A,B,C) \in t$-FT. In other words, there exists a poly-time decider algorithm $\mathsf{VR}_{tFT}(f,q) \to \{0,1\}$ that gathers $U$, checks that $S \subset U$, and then checks that for all $l \in L$, $l$ is locked by $S$.

### 6.3 Reveal Encoding

Since we aren't working with the actual matrices $A, B, C$, but rather their arithmetizations, we need to discuss how we will encode subsets of the rows and sets of locked values.

Let $R \subset \mathbb{H}$ correspond to the indices of the revealed rows. In order to reveal whole constraints, for any $r \in R$, $r$ is revealed in the $A, B$ and $C$ matrices. Let $I \subset \mathbb{K}$ be the group elements $x \in K$ where $\text{row}(x) \in R$. For an R1CS instance encoded into the five polynomials $\vec{o} = (\text{row}, \text{col}, \text{val}^A, \text{val}^B, \text{val}^C)$, a the set of constraints on rows $R$ consists of the five polynomials: $\text{row}_r, \text{col}_r, \text{val}_r^A, \text{val}_r^B, \text{val}_r^C$ where for $f \in \vec{o}$, $f_r : I \mapsto \mathbb{H} \cup 0$ is simply $f_r(x) = f(x)$

Note then that $R$ is equal to and defined by $\text{row}_r(I)$. In addition, we encode a set of locked points $L \subseteq \mathbb{K}$ using a polynomial $\text{col}_u$. A valid $\text{col}_u$ looks just like col, but with disclosed locked values zeroed out:

$$\text{col}_u(x) = \begin{cases} \text{col}(x) & \text{if } \notin L \\ 0 \text{ or } \text{col}(x) & \text{otherwise} \end{cases}$$

In order then for $q = (\text{row}_r, \text{col}_r, \text{val}_r^A, \text{val}_r^B, \text{val}_r^C, \text{col}_u)$ to be considered a valid reveal of an R1CS instance encoded as $\vec{o} = (\text{row}, \text{col}, \text{val}^A, \text{val}^B, \text{val}^C)$, on rows $R$, we simply need all of the polynomials in $q$ to be as defined (note, all of these polynomials are interpolated over $\mathbb{K}$, and thus have degree bound $|\mathbb{K}|$).

So, a proof of reveal for $t$-FT is as follows:

**Definition 14 (Proof of Reveal for $t$-FT).** *$\Pi$ is a **proof of reveal for $t$-FT** if it is a secure* polyIOP *for the following oracle relation:*

$$\begin{aligned} \mathcal{R}_{\mathsf{TFT-POR}} = \{ &\vec{o} = (row, col, val^A, val^B, val^C), \\ &q = (row_r, col_r, val_r^A, val_r^B, val_r^C, col_u), i = (A, B, C) \in t - FT) : \\ &q \text{ is a valid reveal of } i \text{ as defined above, } \vec{o} \text{ encodes } i \} \end{aligned}$$

With this, the stage is set build a proof of reveal. First, we present some necessary sub-protocols and pre-processing steps.

### 6.4 Pre-Processing

Before the reveal can take place, there is a bit of preprocessing that needs to happen in both the verifier and the prover. They will need to both interpolate the following polynomial $U : \mathbb{K} \to \mathbb{H} \cup \{0\}$:

$$\mathsf{U}(k^i) = \begin{cases} h^i & \text{if } i < |\mathbb{H}| \\ 0 & \text{otherwise} \end{cases}$$

This "universal" polynomial creates a mapping between the group members of $\mathbb{K}$ and $\mathbb{H}$, which will become useful to us in the following compliment protocol.

For the rest of this section, it is safe to assume that all provers and verifiers have interpolated this polynomial beforehand (this can be done in time $O(|\mathbb{K}|\log(|\mathbb{K}|))$ via Fast Fourier Transform (FFT) [8]). Additionally, assume every prover has sent their verifier a commitment to $\mathsf{U}$ which the verifier has checked. With these assumptions, every verifier can query $\mathsf{U}$ in constant time by asking the prover for an evaluation proof.

### 6.5 Marlin PolyIOPs

In this section, we will give a series of polyIOPs that will culminate in a proof-of-reveal for $t$-FT.

**Borrowed Protocols** For the rest of this paper, we will borrow the following polyIOPs. Proofs of their security are available in [4,7].

- **Zero over** $\mathbb{K}$: Let $F(X)$ be a virtual oracle:

  $$F(X) := G\left(X, f_{j_1}(\alpha_1 X), \ldots, f_{j_t}(\alpha_t X)\right) \in \mathbb{F}^{(<D)}[X]$$

  where $f_1 \ldots f_n$ are concrete oracles, $j_i \in [n]$, $\alpha_i \in \mathbb{F}^*$, and $D, t, \deg(G)$, as well as $G$'s monomial count are $\mathsf{negl}(\lambda)$. Then Zero over $\mathbb{K}$ is a secure polyIOP for the following relation:

  $$\mathcal{R}_{\mathsf{zero}} = \{((f_1, \ldots f_n), (\vec{\alpha}, \vec{j}, G), \bot) : f_i \in \mathbb{F}^{(<B)}[X], \vec{\alpha} \in (\mathbb{F}^*)^t, \vec{j}[n]^t, \\ \forall k \in \mathbb{K}, F(k) = 0\}$$

- **Multiset Equality over** $\mathbb{K}$ is a secure polyIOP for the following relation:

  $$\mathcal{R}_{\mathsf{ms}} = \left\{f, g \in \mathbb{F}^{(<B)}[X], \bot, \bot) : \{\{f(k) : k \in \mathbb{K}\}\} = \{\{g(k) : k \in \mathbb{K}\}\}\right\}$$

- **Group Product over** $\mathbb{K}$ is a secure polyIOP for the following relation:

  $$\mathcal{R}_{\mathsf{GrpProd}} = \left\{(f \in \mathbb{F}^{(<B)}[X], \bot, \bot) : \prod_{k \in \mathbb{K}} f(k) = 1\right\}$$

**Compliment from** $\mathbb{K}$ **to** $\mathbb{H}$ Now, we will begin constructing explicit polyIOPs. Let $\mathcal{V}$ have access to oracle polynomials $f, g$ from $\mathcal{P}$ that they know map $\mathbb{K}$ to $\mathbb{H} \cup \{0\}$, and they also know that 0 is in the image of both polynomials. First, we will design a polyIOP for the relation $\mathcal{R}_{\mathsf{comp}}$, which convinces a verifier that the image of $g$ does not intersect with the image of $f$ except at 0.

**Compliment from $\mathbb{K}$ to $\mathbb{H}$**

$$\mathcal{R}_{\mathsf{comp}} = \{((f,g) \in \mathbb{F}^{(<|\mathbb{K}|)}[X], \perp, \perp) : g(\mathbb{K}) = \mathbb{H} \setminus f(\mathbb{K}) \cup \{0\}\}$$

---

1. $\mathcal{P}$ interpolates the following polynomials $\mathbb{K} \mapsto \mathbb{H} \cup \{0\}$ and sends grants oracle access to $\mathcal{V}$:
   - $U_f(x) = \begin{cases} U(x) & \text{if } U(x) \in f(\mathbb{K}) \\ 0 & \text{otherwise} \end{cases}$
   - $U'_f(x) = \begin{cases} 1/U_f(x) & \text{if } U_f(x) \neq 0 \\ 0 & \text{otherwise} \end{cases}$
2. Run Zero over $\mathbb{K}$ to confirm that $U_f(x)(U_f(x) - U(x)) = 0$
3. Run Multiset Equality to confirm that $U_f(\mathbb{K}) = f(\mathbb{K})$
4. $\mathcal{V}$ sends a random challenge $c$ to $\mathcal{P}$
5. Run Zero over $\mathbb{K}$ to confirm that $(U_f(x) + c \cdot U'_f(x))(U_f(x)U'_f(x) - 1) = 0$
6. Run Multiset Equality to confirm that $g(\mathbb{K}) = (U \cdot (1 - (U_f \cdot U'_f)))(\mathbb{K})$

**Theorem 3.** *Compliment from $\mathbb{K}$ to $\mathbb{H}$ is a secure for the relation $\mathcal{R}_{\mathsf{comp}}$*

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

*Completeness*: Suppose that $f, g$ are functions from $\mathbb{K} \mapsto \mathbb{H} \cup \{0\}$ such that $\mathbb{K} \setminus f(\mathbb{K}) \cup \{0\} = g(\mathbb{K})$. $\mathcal{P}$ honestly generates $U_f, U'_f$ and sends their oracles to $\mathcal{V}$. With honest generation, steps 2 and 4 pass with the completeness of their respective protocols. In step 5, the right factor will be 0 when $U(x) \in f(\mathbb{K})$ and the left factor will be 0 otherwise, so it will pass with completeness of Zero. Finally, in step 6, notice that the right hand side of the equation in step 5 is exactly equal to $\mathbb{K} \setminus f(\mathbb{K}) \cup \{0\}$, so step 5 passes by completeness of Multiset Equality. Therefore, this protocol is complete.

*Soundness*: Suppose that $f, g$ are functions from $\mathbb{K} \mapsto \mathbb{H} \cup \{0\}$ such that $\mathbb{K} \setminus f(\mathbb{K}) \cup \{0\} \neq g(\mathbb{K})$. Let $\mathcal{P}$ pick any $U_f, U'_f$. To pass step 2, on every $x \in \mathbb{K}$, $U_f(x)$ has to equal either 0 or $U(x)$. Then, to pass step 3, it must be that $U_f(\mathbb{K}) = f(\mathbb{K})$. It is clear to see that these two properties force $U_f$ to be as defined in the protocol, so we can trust it. In step 5, if the right factor is equal to 0 then $U'_f(x) = 1/U_f(x)$. By Schwartz-Zippel lemma, with random challenge $c$, if left factor is equal to 0 then with high probability $U_f(x) = U'_f(x) = 0$. So, if the Zero test in step 5 passes, then $U'_f$ must be as defined, so we can trust it. Since we now trust the given polynomial oracles, we know that $(U \cdot (1 - (U_f \cdot U'_f)))(\mathbb{K}) = K \setminus f(\mathbb{K}) \cup \{0\}$, and can conclude soundness from step 6.

*HVZK*: Honest verifier zero-knowledge of this protocol follows from the subprotocols. In particular, the sent oracles reveal exactly the information about the image of $f$ that we tolerate $\mathcal{V}$ knowing. □

**Proof of Reveal for $t$-FT** Finally, we are ready to present a proof of reveal for $t$-FT.

**$t$-FT Reveal**

1. For $f \in \{\text{row}_r, \text{col}_r, \text{val}_r^A, \text{val}_r^B, \text{val}_r^C, \text{col}_u\}$, $\mathcal{P}$ interpolates the following polynomial and sends an oracle to $\mathcal{V}$
$$f_{\mathbb{K}}(x) = \begin{cases} f(x) & \text{if } x \in I \\ 0 & \text{otherwise} \end{cases}$$
2. Run Zero over $\mathbb{K}$ to confirm that $\text{row}_{r,\mathbb{K}}(x)(\text{row}_{r,\mathbb{K}}(x)) - \text{row}(x)) = 0$
3. Run Compliment from $\mathbb{K}$ to $\mathbb{H}$ to verify
$$\text{row}_{r,\mathbb{K}}(\mathbb{K}) = \mathbb{H} \setminus (\text{row} - \text{row}_{r,\mathbb{K}})(\mathbb{K}) \cup \{0\}$$
4. For $f \in \{\text{col}, \text{val}^A, \text{val}^B, \text{val}^C\}$:
   - $\mathcal{V}$ sends new random challenge $c$ to $\mathcal{P}$
   - Run Zero over $\mathbb{K}$ to confirm that $(\text{row}_{r,\mathbb{K}}(x) + cf_{r,\mathbb{K}}(x))(f(x) - f_{r,\mathbb{K}}) = 0$
5. Run Zero over $\mathbb{K}$ to confirm that $\text{col}_{u,\mathbb{K}}(x)(\text{col}_{u,\mathbb{K}} - \text{col}(x)) = 0$
6. Run Compliment from $\mathbb{K}$ to $\mathbb{H}$ to verify
$$(\text{col}_{r,\mathbb{K}} - \text{col}_{u,\mathbb{K}})(\mathbb{K}) = \mathbb{H} \setminus (\text{col} - \text{col}_{r,\mathbb{K}} + \text{col}_{u,\mathbb{K}})(\mathbb{K}) \cup \{0\}$$
7. $\mathcal{P}$ interpolates the following polynomial and sends an oracle to $\mathcal{V}$:
$$BM(x) = \begin{cases} 1 & \text{if } x \in I \\ 0 & \text{otherwise} \end{cases}$$
8. Run Zero over $\mathbb{K}$ to confirm that $BM(x)(1 - BM(x)) = 0$
9. Run Zero over $\mathbb{K}$ to confirm that $BM(x)\text{row}(x) - \text{row}_{r,\mathbb{K}}(x) = 0$
10. For $f \in \{\text{row}_r, \text{col}_r, \text{val}_r^A, \text{val}_r^B, \text{val}_r^C, \text{col}_u\}$, run Zero over $\mathbb{K}$ to confirm that $BM(x)f(x) - f_{\mathbb{K}}(x) = 0$

**Theorem 4.** *$t$-FT Reveal is a proof of reveal for $t$-FT*

*Proof.* We must show completeness, soundness and knowledge-soundness. HVZK follows from the subprotocols.

*Completeness:* Let's assume that $q$ is a valid reveal of $\vec{o}$, and that the prover is honest. We can step through every test (i.e. every rejection opportunity), and see if the protocol always passes. The Zero test from step 2 will pass, as $\text{row}_{r,\mathbb{K}}(x)$ is always either zero or $\text{row}_r(x) = \text{row}(x)$. Step 3 will pass, as the image of $\text{row}_{r,\mathbb{K}}$ is exactly $\text{row}_r(I) \cup \{0\}$, and $\mathbb{H} \setminus (\text{row} - \text{row}_{r,\mathbb{K}})$ is exactly $\text{row}_r(I)$. Next, stepping into a general case of step 4, we see that if $x \in I$ then $f(x) - f_{r,\mathbb{K}} = 0$, and if $x \notin I$ then $\text{row}_{r,\mathbb{K}}(x) = 0$ and $c * f_{r,\mathbb{K}}(x) = 0$. Next, based on the definition of a valid $\text{col}_u$ polynomial, we know that either $\text{col}_u(x) = 0$ or $\text{col}_u(x) - \text{col}(x) = 0$, so step 5 passes. Next, consider the image polynomial $\text{col}_{r,\mathbb{K}} - \text{col}_{u,\mathbb{K}}$. This is exactly $\text{col}_{r,\mathbb{K}}(\mathbb{K})$ with the unsecured elements removed. Given an honest prover, this set would be exactly $\mathbb{H} \setminus (\text{col} - \text{col}_{r,\mathbb{K}} + \text{col}_{u,\mathbb{K}})(\mathbb{K}) \cup \{0\}$, so step 6 passes. Steps 8, and 9 simply check that the polynomial $BM$ is formed as intended, using $\text{row}_{r,\mathbb{K}}$ as a tool to separate $I$ from $\mathbb{K} \setminus I$, and will certainly pass. Finally, $BM$ zeros out everything outside of $I$, we know that over $\mathbb{K}$ $BM(x)f(x) = f_{\mathbb{K}}(x)$ for all $f$ in $q$, so step 10 passes. Thus, the protocol is complete.

*Soundness:* Given a proposed valid reveal $q$ we will show that to pass all the tests, $q$ must be valid. To pass step 2, it must be that $\text{row}_{r,\mathbb{K}}$ is either zero or $\text{row}(x)$ for all $x \in \mathbb{K}$. For step 3 to pass, notice that there cannot be a $y \in \mathbb{H}$ such that $y \in \text{row}_{r,\mathbb{K}}(\mathbb{K})$ and $y \in (\text{row} - \text{row}_{r,\mathbb{K}})(\mathbb{K})$. In other words, to pass step 3, it must be that $\text{row}_{r,\mathbb{K}}$ encodes entire rows (i.e, there are no missing non-zero elements in the rows that it reveals). With these two pieces of information, we know that the reveal set $I \subset \mathbb{K}$, which is determined by $\text{row}_{r,\mathbb{K}}$, is as desired (and by consequence, that $\text{row}_{r,\mathbb{K}}$ is exactly the polynomial described in step 1). Stepping into a general case of step 4, the Zero test can only pass if for every $x \in \mathbb{K}$ either $f(x) = f_{r,\mathbb{K}}(x)$ or $\text{row}_{r,\mathbb{K}} + cf_{r,\mathbb{K}} = 0$. Looking at the latter condition, we know it is equivalent to saying $\text{row}_{r,\mathbb{K}} = 0 = f_{r,\mathbb{K}}$ with overwhelming probability. So to pass step 4, each $f_{r,\mathbb{K}}$ must be as described in step 1.

Next, step 5 straightforwardly passes from the description of $\text{col}_{u,\mathbb{K}}$. In step 6, the image of $\text{col}_{r,\mathbb{K}} - \text{col}_{u,\mathbb{K}}$ is exactly the set of secured elements (plus 0). To pass step 6, it must be that the secured elements do not ever show up in the image of $\text{col} - \text{col}_{r,\mathbb{K}} + \text{col}_{u,\mathbb{K}}$, which is exactly the property we want from these polynomials. So at this point, we know that all $f_{\mathbb{K}}$ from step 1 are as formed, and that their collection encodes a valid reveal. What is left to show is that the low-degree polynomials in the verifier's hand are actually consistent with each $f_{\mathbb{K}}$. To do this, we use a bitmask polynomial $BM$ that is 1 on $I$ and zero everywhere else. To pass step 8, BM must be 0 or 1 everywhere on $\mathbb{K}$. T0 pass step 9, $BM$ must be 0 outside of $I$ and 1 inside of $I$, so we now know that BM is as described in step 7. Finally, to pass step 10, each function $f \in q$ must match $f_{\mathbb{K}}$ on $I$, which qualifies $q$ as a valid reveal! So, we have shown the protocol is sound.

*Knowledge-Soundness:* Knowledge soundness here is easy to see, as $\vec{o}$ explicitly encodes the witness $(A, B, C)$. Given oracle access to $\vec{o}$ (which a malitious $\tilde{\mathcal{P}}$ cannot tamper with), an extractor keep querying $\vec{o}$ until they recover all of $(A, B, C)$. □

## 7 Revealable Functional Commitments from Plonk

Plonk [7] is an AHP that proves evaluations of arithmetic circuits. In this section, we work with a simplified version of the Plonk encoding, consistent with proof of function relation found in [4]. After orienting ourselves in the revealable functional commitments context, we can present a proof of reveal.

### 7.1 $\mathcal{AC}$ as a Revealable Functional Set

Arithmetic circuits yeild quire nicely to our setting. Recall that $\mathcal{AC}_{n_i,n_g,n_o}$ (sometimes abbriviated as $\mathcal{AC}$) is the set of arithmetic circuits with $n_i$ inputs, $n_g$ gates, and $n_o$ outputs. As it turns out, $\mathcal{AC}$ is already a revealable functional set. Since there are no witness inputs, each $C \in \mathcal{AC}_{n_i,n_g,n_o}$ must be a function of the public witness inputs, so we know that it is a functional set. As for revealability, we discussed earlier that subcircuits of $C$ constitute a subset of the constraints that $C$ imposes on its inputs. Additionally, there exists an efficient algorithm to verify that a circuit $C'$ is truly a subcircuit of $C$. Finally, given a set of locked wire pins in a subcircuit, it is easy to check that there are no outgoing wires that constrain those pin values outside of the subcircuit. We will formalize these notions in our discussion of the Plonk arithmetization.

### 7.2 Arithmetization and Reveal Encoding for Plonk

To encode circuits in $\mathcal{AC}_{n_i,n_g,n_o}$, Plonk assumes the existance of two multiplicative subgroups of $\mathbb{F} : \mathbb{K}_g = \langle \gamma^3 \rangle$ of order $n_g$ and $\mathbb{K} = \langle \gamma \rangle$, of order $3n_g$. The elements of $\mathbb{K}$ represent pin indicies, while the elements of $\mathbb{K}_g$ represent gates. This is intuitive, as each gate has 3 pins (two input and one output).

In the Plonk scheme, we represent a circuit $C$ as two polynomials, $(w, s)$ upon $\mathbb{K}$ and $\mathbb{K}_g$ respectively. For each gate $l \in \{0, \dots (n_g - 1)\}$:

- $s(\gamma^{3l}) = 1$ if gate $l$ is an addition gate and 0 if $l$ is a multiplication gate.
- $\gamma^{3l}, \gamma^{3l+1}, \gamma^{3l+2}$ correspond to the left pin, right pin, and output pin of gate $l$. $w$ permutes all of $\mathbb{K}$ with respect to the wiring of $C$. Therefore each cycle of $w$ represents an equivalence class of pins. A more precise definition can be found in [7, 4].

**Reveal Encoding** We define a valid reveal of a circuit $C = (w, s)$ on the group $\mathbb{K}$ is a tuple $q = (I, w_S, s_S, L \subset I)$ where $I \subset \mathbb{K}$ is *contiguous* such that:

1. $\forall x \in I^g = I \cap \mathbb{K}^g$, $s_S(x) = s(x)$
2. $\forall x \in I$, $w_S(x) = w(x)$
3. $w_S$ is a permutation on $L$

where $I^g$ is the "gate set" which is the intersection $\mathbb{K}^g \cap I$. These properties succinctly capture the notion of valid reveal for the plonk arithmetization. Since $I$ is contiguous, it is immediate that $S = (w_S, s_S)$ encodes a subcircuit of $C$.

By the first property, we know that $S$ correctly presents the gate types of the subcircuit, and by the second property, we know that it correctly represents all the wires of the subcircuit. So, use these two properties to achieve consistency. As for lockedness, the pins $\in L$ are those designated to be locked. The third property implies that there are no wires extending from $L$ to other parts of the circuit (in $\mathbb{K} \setminus L$). If there were, then there would be a cycle in $w$ that intersects $L$, and no choice of $w_S$ could simultaneously agree with $w$ on $L$ and be a permutation on $L$. So, these three properties are capture the general definition of a valid reveal, enabling a definition of a *proof of reveal for $\mathcal{AC}$*:

**Definition 15 (Proof of Reveal for $\mathcal{AC}$).** *$\Pi$ is a* ***proof of reveal for*** *$\mathcal{AC}_{n_i,n_g,n_o}$ if it is a secure* polyIOP *for the following oracle relation:*

$$\begin{aligned}\mathcal{R}_{\mathsf{AC-POR}}(I) = \{&\vec{o} = (w, s),\\ &q = (I, w_S, s_W, L), C \in \mathcal{AC}_{n_i,n_g,n_o}) :\\ &q \textit{ is a valid reveal of } i \textit{ as defined above, } \vec{o} \textit{ encodes } C\}\end{aligned}$$

With this, the stage is set build a proof of reveal. First, we present some necessary sub-protocols and pre-processing steps.

## 7.3 Pre-Processing

In these protocols, it is assumed that the following vanishing polynomials are computed in pre-processing and are queryable by the prover and verifier:

$$v_I(x) = \prod_{i\in I}(x-i) \qquad v_L(x) = \prod_{i\in I}(x-i) \qquad v_I^g(x) = \prod_{g\in I^g}(x-g)$$

Instead of having to generate these polynomials as above over a poorly-formed subset, we can generate them directly over a subgroup using a similar approach to the Marlin protocol. This is done by the prover interpolating the indicator polynomial $BM$ (as in Marlin) and sending an oracle to the verifier. We will see this in the final PLONK protocol.

We will also have the verifier and the prover calculate the following:

$$t(x) = \frac{x^{|\mathbb{K}|}-1}{v_I(x)} + v_I(x) \qquad\qquad \phi = \prod_{k\in K} t(k)$$

Finally, they will interpolate a polynomial $g$ over $\mathbb{K}$, where $g(1) = \phi$ and $g(x) = 1$ for all other $x \in \mathbb{K}$. In total, the preprocessing is bottlenecked by the calculation of the vanishing polynomials, which using FFT polynomial multiplication [8] and a divide-and-conquer approach, this pre-processing can be completed with $O(i \log^2(i))$ complexity, where $i = |I|$.

## 7.4 Marlin PolyIOPs

In this section, we will give a series of polyIOPs that will culminate in a proof-of-reveal for $\mathcal{AC}$.

**Zero over *I*** We construct a secure polyIOP to test if a virtual oracle is zero over $I \subset \mathbb{F}$. Let $F$ be a virtual oracle defined as follows:

$$F(X) := G\left(X, f_{j_1}(\alpha_1 X), \dots f_{j_t}(\alpha_t X)\right) \in \mathbb{F}^{(<D)}[X]$$

where $f_1, \dots f_n$ are concrete oracles, and $t = D = \deg(G)$. Our polyIOP shows that $\forall x \in I, F(x) = 0$. This also handily enables equality checking over any set $I$.

**Zero over** $I$ Relation:

$$\mathcal{R}_{\text{zero}} = \{((f_1, \dots f_n), (\vec{\alpha}, \vec{j}, G), \perp)) : f_i \in \mathbb{F}^{(<B)}[X], \vec{\alpha} \in (\mathbb{F}^*)^t, \vec{j} \in [n]^t$$
$$\forall x \in I, F(x) = 0, \text{ where F is defined above}\}$$

---

1. For $i \in [t]$, let $h_i = f_i$. $P$ samples random $r_i \leftarrow \mathbb{F}^{(<2)}[X]$ and computes mask $m_i(X) = r_i(\alpha_i^{-1} X) \cdot v_I(\alpha_i^{-1} X)$, and computes $h_i' = h_i + m_i$.
2. $\mathcal{P}$ computes $F'(X) = G(X, h_1'(\alpha_1 X), h_2'(\alpha_2 X), \dots, h_t'(\alpha_t X))$ and quotient $q_1 = F'/v_I$. $\mathcal{P}$ sends polynomials $\{m_i\}_i, \{r_i\}_i$ and $q_1$ with degree bounds $B$, 2 and $D \cdot B - |I|$, respectively.
3. For $j \in [t]$, $\mathcal{V}$ derives $h_i' = h_i + m_i$ through additive homomorphism. $\mathcal{V}$ randomly samples $\beta_1$, $\beta_2, c \leftarrow \mathbb{F}^* \setminus \mathbb{K}$ and sends $c$.
4. $\mathcal{P}$ computes $q_2 = r_1 + cr_2 + \dots + c^{t-1} r_t$. $\mathcal{V}$ derives concrete oracle $q_2$ through additive homomorphism.
5. Let $M(X) = m_1(\alpha_1 X) + cm_2(\alpha_2 X) + \dots + c^{t-1} m_t(\alpha_t X)$. $\mathcal{V}$ computes $v_I(\beta_1), v_I(\beta_2)$, queries $q_1(\beta_1)$ and $q_2(\beta_2)$, and queries $h_i'(\alpha_i \beta_1)$ and $m_i(\alpha_i \beta_2)$. $\mathcal{V}$ then asserts

$$M(\beta_2) - q_2(\beta_2) \cdot v_I(\beta_2) \stackrel{?}{=} 0$$
$$F'(\beta_1) - q_1(\beta_1) \cdot v_I(\beta_1) \stackrel{?}{=} 0$$

**Theorem 5.** *Zero over I is a secure polyIOP for relation* $\mathcal{R}_{zero}$

*Proof. The proof of this is almost identical to the one seen in [4], so we will omit it here.* □

**Set Product over *I*** The next polyIOP we present is to complete the same product check as Group Product over $\mathbb{K}$, but over an arbitrary subset $I$ of a group $\mathbb{K}$ rather than the entire group. With this, they will be able to show that for a function $f$, $\prod_{x\in I} f(x) = 1$.

**Set Product over** $I$ Relation:

$$\mathcal{R}_{\text{SetProd}} = \left\{ (f \in \mathbb{F}^{(<B)}[X], \perp, \perp) : \prod_{x\in I} f(x) = 1 \right\}$$

---

1. $\mathcal{P}$ computes $f'(x) = \frac{f(x)(x^{|\mathbb{K}|}-1)}{g(x)v_I(x)} + \frac{v_I(x)}{g(x)}$ and sends an oracle of $f'$ to $\mathcal{V}$
2. Run Zero over $\mathbb{K}$ to verify that:

$$v_I(x)(g(x)f'(x) - v_I(x)) - f(x)(x^{|\mathbb{K}|} - 1) \equiv 0$$

3. Run Group Product over $\mathbb{K}$ to verify that $\prod_{k\in K} f'(x) = 1$

**Theorem 6.** *Set Product over $I$ is a secure polyIOP for relation $\mathcal{R}_{SetProd}$*

*Proof. There is no witness, so it suffices to show completeness, soundness, and honest-verifier zero-knowledge. HVZK follows from the HVZK of the underlying sub-protocols*

*Completeness: By construction it is clear that steps 1, 2 are complete. Now assume that $\prod_{x\in I} f(x) = 1$. Then, it must be that*

$$\begin{aligned}
\prod_{k\in K} f'(x) &= \prod_{x\in I}\left[\frac{f(x)(x^{|\mathbb{K}|}-1)}{g(x)v_I(x)} + \frac{v_I(x)}{g(x)}\right] \cdot \prod_{x\in\mathbb{K}\setminus I}\left[\frac{f(x)(x^{|\mathbb{K}|}-1)}{g(x)v_I(x)} + \frac{v_I(x)}{g(x)}\right] \\
&= \prod_{x\in I}\left[\frac{f(x)(x^{|\mathbb{K}|}-1)}{g(x)v_I(x)}\right] \cdot \prod_{x\in\mathbb{K}\setminus I}\left[\frac{v_I(x)}{g(x)}\right] \\
&= \prod_{x\in I}[f(x)] \cdot \prod_{x\in I}\left[\frac{(x^{|\mathbb{K}|}-1)}{g(x)v_I(x)}\right] \cdot \prod_{x\in\mathbb{K}\setminus I}\left[\frac{v_I(x)}{g(x)}\right] \\
&= 1 * \prod_{x\in\mathbb{K}}\left[\frac{x^{|\mathbb{K}|}-1}{g(x)v_I(x)} + \frac{v_I(x)}{g(x)}\right] \\
&= \prod_{x\in\mathbb{K}}\left[\frac{x^{|\mathbb{K}|}-1}{v_I(x)} + v_I(x)\right] \cdot \prod_{x\in\mathbb{K}}\left[\frac{1}{g(x)}\right] \\
&= \phi \cdot \frac{1}{\phi} = 1
\end{aligned}$$

*So we know that the check in step 3 passes and have shown completeness.*

*Soundness: Suppose $\prod_{x\in I} f(x) = p$. We want to show that if the verifier $\mathcal{V}$ accepts the proof described above, then $p = 1$.*

*Firstly, we note that* $\phi = \prod_{k \in K} t(k) = \prod_{x in \mathbb{K}} \left( \frac{x^{|\mathbb{K}|}-1}{v_I(x)} + v_I(x) \right)$ *is non-zero because it is the sum of two vanishing polynomials each of which are non-zero on the corresponding set's complement.*

*If Step 3 holds if* $\prod_{x \in I} f(x) = p$*, then*

$$\prod_{k \in \mathbb{K}} f'(x) = p \cdot \phi \cdot \frac{1}{\phi} = 1$$

*implying* $p = 1$*, as required.* □

**Generalized Multiset Equality over *I*** We present a polyIOP for checking that the images of two polynomials are the same as multisets over an arbitrary subset $I$ of subgroup $\mathbb{K}$.

**Generalized Multiset Equality over** $I$ Relation:

$$\mathcal{R}_{mse} = \Big\{ ((f, g) \in \mathbb{F}^{(<B)}[X], \perp, \perp) : \{\{f(x) : x \in I\}\} = \{\{g(x) : x \in I\}\} \Big\}$$

---

1. $\mathcal{P}$ computes $h(x) = \frac{f(x)}{g(x)}$ and sends it to $\mathcal{V}$.
2. $\mathcal{P}$ and $\mathcal{V}$ run Set Product over $I$ to check that $\prod_{x \in I} h(x) = 1$.
3. Run Zero on $I$ to confirm $h(x)g(x) - f(x) = 0$.

**Theorem 7.** *Generalized Multiset Equality over I is a secure polyIOP for relation* $\mathcal{R}_{mse}$.

*Proof.* The proof of this is almost identical to the one seen in [4], so we will omit it here. □

**Permutation on *I*** We can now use the generalized multiset equality to formulate a polyIOP to check whether the image of some polynomial is a permutation of its domain.

**Permutation on** $I$ Relation:

$$\mathcal{R}_{perm} = \Big\{ ((w \in \mathbb{F}^{(<B)}), \perp, \perp) : w(I) = I \Big\}$$

---

1. Run Generalized Multiset Equality over $I$ on $(w, Id)$ where $Id$ is the identity function $Id(x) = x$ to verify $\{\{w(x) : x \in I\}\} = \{\{x : x \in I\}\}$.

**Theorem 8.** *Permutation on I is a secure polyIOP for relation* $\mathcal{R}_{perm}$.

*Proof.* The proof of this is almost identical to the one seen in [4], so we will omit it here. □

**Proof of Reveal for $\mathcal{AC}$** Finally, we are ready to present a proof of reveal for $\mathcal{AC}$:

$\mathcal{AC}$ **Reveal**

1. Use Zero over $I^g$ to verify that $s_S(X) - s(X) \equiv 0$ over $I^g$
2. Use Zero over $I$ to verify that $w_S(X) - w(X) \equiv 0$ over $I$
3. Use Permutation over $L$ to verify that $w_S$ is a permutation on $L$

**Theorem 9.** *$\mathcal{AC}$ Reveal is a proof of reveal for $\mathcal{AC}_{n_i,n_g,n_o}$*

*Proof.* Soundness, completeness, and HVZK follow from the underlying protocols. Knowledge soundness follows directly from the oracle access to $w, s$. An extractor can simply perform a trace through $\mathbb{K}$ to retrieve the entire original circuit $C$. □

Thus, we have presented all the components that instantiate a revealable functional commitment for Plonk!

## 8 Conclusion and Future Work

In this paper, we have defined the idea of a revealable functional commitment, and showed how to construct them using an algebraic holomorphic proof, a proof of function relation, and a newly-coined proof of reveal. We then explicitly construct proofs of reveal for both Marlin and Plonk, two AHP's that capture the entire space of arithmetic circuits. By combining these with their associated PFRs, we are able to get two public-coin revealable functional commitment schemes. For all protocols, verification time is logarithmic in the size of the function and linear in the input/reveal size, and proving time is quasi-linear. Furthermore, all interactive protocols we present can be turned non-interactive using the Fiat-Shamir heuristic.

We hope that future work can design revealable functional commitment schemes and/or proofs of reveals for different proof systems. We are also excited about the potential application areas for this work. Revealable functional commitments enable zero-knowledge proofs for many algorithmic predicates. Outside of those mentioned, possible applications include intellectual property law, new proofs of work for decentralized systems, and more. Additionally, next steps might include zero knowledge proofs for particular algorithmic fairness predicates.

## References

1. Arnold, D., Dobbie, W., Yang, C.S.: Racial Bias in Bail Decisions*. The Quarterly Journal of Economics **133**(4), 1885–1932 (05 2018). https://doi.org/10.1093/qje/qjy012, `https://doi.org/10.1093/qje/qjy012`
2. Ben-Sasson, E., Chiesa, A., Goldberg, L., Gur, T., Riabzev, M., Spooner, N.: Linear-size constant-query iops for delegating computation. Cryptology ePrint Archive, Paper 2019/1230 (2019), `https://eprint.iacr.org/2019/1230`, `https://eprint.iacr.org/2019/1230`
3. Boneh, D., Drake, J., Fisch, B., Gabizon, A.: Halo infinite: Recursive zk-snarks from any additive polynomial commitment scheme. Cryptology ePrint Archive, Paper 2020/1536 (2020), `https://eprint.iacr.org/2020/1536`, `https://eprint.iacr.org/2020/1536`
4. Boneh, D., Nguyen, W., Ozdemir, A.: Efficient functional commitments: How to commit to a private function. Cryptology ePrint Archive, Paper 2021/1342 (2021), `https://eprint.iacr.org/2021/1342`, `https://eprint.iacr.org/2021/1342`
5. Chiesa, A., Hu, Y., Maller, M., Mishra, P., Vesely, P., Ward, N.: Marlin: Preprocessing zksnarks with universal and updatable srs. Cryptology ePrint Archive, Paper 2019/1047 (2019), `https://eprint.iacr.org/2019/1047`, `https://eprint.iacr.org/2019/1047`
6. arkworks contributors: `arkworks` zksnark ecosystem (2022), `https://arkworks.rs`
7. Gabizon, A., Williamson, Z.J., Ciobotaru, O.M.: Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. IACR Cryptol. ePrint Arch. **2019**, 953 (2019)
8. Jia, Y.B.: Polynomial multiplication and fast fourier transform (September 2020)
9. Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-size commitments to polynomials and their applications. In: Abe, M. (ed.) Advances in Cryptology - ASIACRYPT 2010. pp. 177–194. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)