Polynomial IOPs for Linear Algebra Relations

Alan Szepieniec
alan@nervos.org

Nervos Foundation

Abstract. This paper proposes new Polynomial IOPs for arithmetic circuits. They rely on the monomial coefficient basis to represent the matrices and vectors arising from the arithmetic constraint satisfaction system, and build on new protocols for establishing the correct computation of linear algebra relations such as matrix-vector products and Hadamard products. Our protocols give rise to concrete proof systems with succinct verification when compiled down with a cryptographic compiler whose role is abstracted away in this paper. Depending only on the compiler, the resulting SNARKs are either transparent or rely on a trusted setup.

Keywords: SNARK · Polynomial IOP · Zero-Knowledge

1 Introduction

Succinct Non-Interactive Arguments of Knowledge (SNARKs) enable a resource-constrained verifier to cryptographically verify the authentic computations of an untrusted prover. The technology is particularly well-suited to the cryptocurrency setting, where participants are typically anonymous, untrusted, and where the success of the network depends on the capability of lightweight nodes to verify the network's consensus (however that is defined). In this setting, there is a large monetary incentive for malicious behavior.

Despite the flurry of rapid related and unrelated developments by diverse parties, two trends are emerging as good practice in this domain.

1. *Functional separation in the compilation pipeline.* The compilation process for general purpose zero-knowledge proofs is separated into multiple steps with clear boundaries. At the input of this pipeline is a computation, represented either as program source code or as a circuit. A technique called arithmetization turns this computation into a constraint system involving native operations over a finite field. The next step transforms this constraint system into an abstract proof system between two parties, prover P and verifier V, that are interactive Turing machines with access to unrealistic or unrealizable resources such as PCP oracles. The abstract proof systems in this step typically achieve statistical or even perfect security. In the last step, the *cryptographic compilation*, the unrealistic resources are replaced by cryptographic approximations that achieve the same functionality at the expense of introducing computational hardness assumptions for security.

2. *Polynomial IOP formalism.* The abstract information-theoretical proof system in the step before cryptographic compilation could in principle rely on a variety of unrealistic resources, and build a sound proof system from their mathematical properties. However, for the purpose establishing soundness, the Schwartz-Zippel lemma is an indispensable tool. The strategy is to reduce the satisfaction of arithmetic constraints arising from the constraint system to series of identities of *low-degree polynomials*. By evaluating these polynomials in random points, their equality is tested probabilistically. If the left and right hand sides of an equation represent identical polynomials, they are identical everywhere, but if they are unequal they are different *almost* everywhere. The Schwartz-Zippel lemma provides an exact concrete quantification of the security lost due to this probabilistic approximation. A *Polynomial IOP* is the abstract proof system tailored to this strategy. In this formalism, the prover sends low degree polynomials to the verifier, and rather than reading the entire list of coefficients, the verifier queries these polynomials in a given point through an oracle interface. The cryptographic compiler uses a *polynomial commitment scheme* to simulate this unrealistic resource.

These trends are visible in the rise of universal SNARKs with universal and updatable structured reference strings (SRS's) such as Sonic [11], PLONK[8], and Marlin [7]. The common idea here is to use the cryptographic pairing-based mathematics only to realize *polynomial commitment scheme*, typically the KZG scheme [10]. Since the SRS is used only for the KZG scheme, it is independent of the preceding abstract proof system and the circuit it encodes; this independence is precisely what enables updates to the SRS and its adaptation to any circuit. PLONK and Marlin independently formalize this abstraction and introduce the terms *Polynomial Protocol* and *Algebraic Holographic Proof (AHP)*, respectively. This paper adopts the terminology of Bünz *et al.* [6], who introduce a new polynomial commitment scheme (and hence a cryptographic compiler) based on groups of unknown order and in the process explore the landscape of protocols it can apply to.

These trends are *also* visible in the rise of IOPs based on Reed-Solomon codes [1,4,3]. The underlying abstract protocols here are not explicitly Polynomial IOPs. However, their common feature is the reliance on Reed-Solomon codewords as the proof oracles. Since Reed-Solomon codewords are obtained by evaluating polynomials in a domain of points whose cardinality is larger than the polynomials' degree, these proof oracles uniquely identify the originating low-degree polynomials. As a result, a Reed-Solomon IOP is a Polynomial IOP in disguise.

Despite the spontaneous convergence onto Polynomial IOPs as a useful formalism, there seems to be little agreement about the optimal interface between Polynomial IOPs and the arithmetic constraint systems that they realize. Arithmetic constraint systems typically express constraints using matrix algebra: in terms of vectors, and matrix multiplication, but also *Hadamard products*, which is a fancy word for the element-wise products of pairs of equal-length vectors. The set of operations that Polynomial IOPs natively offer are somewhat different. As a result, how the Polynomial IOP represents the objects in the arithmetic constraint system and how it simulates the equations that constrain them, are the key questions in the design process of Polynomial IOPs. The various answers to these questions are what set the various Polynomial IOPs for arithmetic circuits apart.

– Marlin and Aurora represent the objects of the arithmetic constraint system as the Reed-Solomon codewords of polynomials. Standard techniques establish the correct computation of a Hadamard product of such codewords. The computation of a linear transform applied to such a codeword is reduced to checking the sum of a related codeword.
– PLONK represents the input and output wires to addition and multiplication gates as the value of a polynomial in a domain of points. The consistency of each gate, and of wires between gates, is verified by checking that a polynomial derived from the description of that circuit evaluates to zero over the same domain.
– Sonic represents the vectors of left, right, and output wires of a series of multiplication gates as the coefficient vectors of three polynomials. The consistency of these multiplication gates, and of a linear transform, is established by checking several properties of bivariate polynomials. The paper furthermore explains under which conditions these bivariate polynomials can be simulated with univariate ones.

Contributions. In this paper we propose a new Polynomial IOP for arithmetic circuits called Claymore[1]. Succinct verification is achieved with an untrusted preprocessing phase. When compiled down using any polynomial commitment scheme, the result is a concrete zk-SNARK with universal updatable structured reference string, or transparent setup, depending only on the nature of the polynomial commitment scheme.

The arithmetic constraint system chosen to represent the arithmetic circuit is the Hadamard Product Relation (HPR), in which the witness consists of three vectors representing the left, right, and output wires of a list of multiplication gates. Naïvely, one would expect each of these vectors to be represented by one polynomial each. However, one of our optimizations represents two of the witness vectors (the left and right wires) by a single polynomial, simply by concatenating the vectors. Another optimization drops the polynomial representing the third witness vector (of output wires) from the transcript altogether. The net result is fewer polynomials in the transcript of the Polynomial IOP.

We note that Sonic realizes a similar constraint satisfaction relation by reducing both the multplication and linear constraints into one large equation. In Claymore, the multiplication gate consistency and linear consistency are achieved in two separate steps, both of which rely on a collection of subprotocols for linear algebra relations that we develop along the way. The separate steps are later merged as an explicit optimization.

[1] A type of Scottish sword.

While the concrete cryptographic compilation is abstracted away, it is possible to make arguments regarding the size of concrete proofs based on the communication complexity of the Polynomial IOP. In this respect, Claymore stands out as the number of polynomials transmitted in the online phase is only 5 – down 16.7% from the runner-up PLONK, whose number stands at 6. As a result, if the number of polynomials in the transcript is the dominant factor of proof size, then Claymore will result in the smallest proofs.

The price to pay for this brevity is two-fold. First, we can only show that zero-knowledge is achieved for a variant of the protocol that has one more polynomial, putting it back on par with PLONK. In other words, the 5 polynomial variant does not seem to be compatible with zero-knowledge. Second, the degree of the largest-degree polynomial is much larger. For Claymore this degree scales with roughly $O(n^2)$, where n is the size of the witness; whereas the alternatives achieve $O(n)$ scaling. Depending on the concrete cryptographic compiler, this behavior is either a minor inconvenience, or a complete blockade, for SNARKifying large enough circuits. We discuss some techniques (that do scale well) for dealing with matrices that are sparse but exhibit a particular kind of structure.

Motivation. The motivation for this work is chiefly theoretical. By studying the interface between arithmetic circuits and Polynomial IOPs in isolation of other constraints and demands, we develop a protocol that achieves its target functionality exactly. As a result of this focus, our protocol is arguably simpler than other protocols that achieve nominally the same thing. Complexity is the friend of mistakes, and our protocol may therefore be the preferred option for this reason even in circumstances where it is inferior in terms of performance.

However, it is by no means clear that Claymore does perform worse in the general case, because this comparison is highly dependent on the parameters of the situation. We illustrate two use cases where Claymore is likely to perform comparably well, if not outright outshine its competition.

Claymore performs extremely well for shallow arithmetic circuits with dense linear transformations. An example of such circuits are the verification circuits of lattice-based and MQ-based signature schemes, which typically involve operations on large matrices and vectors over a small finite field. As a result, a Claymore-SNARK is an outstanding candidate for achieving post-quantum signature aggregation — or indeed, post-quantum signatures with various fancy properties that zero-knowledge proofs enable.

The total number of polynomials in both the proof transcript and the structured or uniform reference string is smallest for Claymore, even when considering the zero-knowledge variant. An example where this number is important is recursive proof composition because the verifier performs operations on all polynomials. By reducing this number, Claymore potentially shrinks the verification circuit, and potentially lowers the threshold for incremental verification.

If it is true that Claymore results in the smallest proofs, then it stands to reason that Claymore will be the SNARK of choice in settings where the bandwidth is the most critical optimization target. Blockchains naturally satisfy this description when they have a fixed block rate and size; as a result, Claymore will allow the network to prove more.

2 Preliminaries

2.1 Indexed Relations

Owing to their convenience, we use *indexed relations* [7]. An indexed relation is a set $\mathcal{R}$ of tuples $(\mathbb{i}, \mathbb{x}, \mathbb{w})$, whose three components are called the *index*, *instance*, and *witness*, respectively. The separation between index and instance captures the intuition that some properties of concrete proofs for $\mathcal{R}$ should be computable from $\mathbb{i}$ even before $\mathbb{x}$ is known. For instance, $\mathbb{i}$ can be the description of an arithmetic circuit, $\mathbb{x}$ the values of the output wires, and $\mathbb{w}$ an assignment of values to all wires that makes the all gates consistent. The projection $\{(\mathbb{i}, \mathbb{x}) \mid (\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}\}$ of triples in $\mathcal{R}$ onto the first two components is the *indexed language corresponding to* $\mathcal{R}$ and is denoted by $\mathcal{L}(\mathcal{R})$.

2.2 Constraint Systems

A constraint system is a representation of a computation in terms of equations with unknown variables. When there is an assignment to the unknown variables that satisfies all equations, we say the constraint system is *satisfiable*, and this assignment is the *witness*. The *index* determines all fixed constants in the equations, and the *instance* determines known variables that can vary independently of the index but are ultimately known by all parties involved.

The following constraint system is adapted from Bootle *et al.* [5].

Definition 1 (Hadamard Product Relation (HPR)). *Let $\mathbb{F}$ be a finite field. A triple $(\mathbb{i}, \mathbb{x}, \mathbb{w})$ where $\mathbb{i} = (m, n, M)$ with $m, n \in \mathbb{N}$, and $M \in \mathbb{F}^{m\times(1+3n)}$, where $\mathbb{x} = \mathbf{x} \in \mathbb{F}^m$, and where $\mathbb{w} = (\mathbf{w_l}, \mathbf{w_r}, \mathbf{w_o}) \in \mathbb{F}^n \times \mathbb{F}^n \times \mathbb{F}^n$; satisfies the* Hadamard Product Relation *iff both*

$$\mathbf{x} = M \begin{pmatrix} 1 \\ \hline \mathbf{w_l} \\ \hline \mathbf{w_r} \\ \hline \mathbf{w_o} \end{pmatrix} \tag{1}$$

and

$$\mathbf{w_l} \circ \mathbf{w_r} = \mathbf{w_o} \ , \tag{2}$$

where $\circ$ denotes the Hadamard (i.e., entry-wise) product; and in this case we write $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}_{\mathsf{HPR}}$.

2.3 Interactive Proof Systems

Definition 2 (Interactive Proof System). *Let $\mathcal{R}$ be an indexed relation with corresponding relation language $\mathcal{L}(\mathcal{R})$. An* interactive proof system *is a pair* (P, V) *of stateful interactive Turing machines such that: the input to* P *is* $(\mathbb{i}, \mathbb{x}, \mathbb{w})$*, the input to* V *is* $(\mathbb{i}, \mathbb{x})$*;* P *and* V *exchange* $\mathsf{r} = \mathsf{r}(|\mathbb{i}|)$ *messages in total; and in the last step of the protocol* V *outputs a single bit* $b \in \{\top, \bot\}$*. The system satisfies two more properties:*

- Completeness — V *accepts members of* $\mathcal{L}(\mathcal{R})$*:* $(\mathbb{i}, \mathbb{x}) \in \mathcal{L}(\mathcal{R}) \Rightarrow b = \top$*.*
- Soundness *(with* soundness error σ*)* — V *rejects non-members of* $\mathcal{L}(\mathcal{R})$ *except with probability at most* σ *taken over the all random coins involved:* $\Pr[(\mathbb{i}, \mathbb{x}) \notin \mathcal{L}(\mathcal{R}) \Rightarrow b = \bot] \geq 1 - \sigma$*.*

Soundness becomes a moot point when for the given index $\mathbb{i}$ every instance $\mathbb{x}$ has a matching witness $\mathbb{w}$ such that $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}$. In this case a stronger notion called *knowledge soundness* [2] is preferred, which informally requires that any adversary that successfully convinces the verifier can be made to leak a witness by an extractor machine that has the same interface as the verifier but can additionally reset the adversary to an earlier point in time without forgetting the observed transcripts. In our context, all witnesses are encoded into oracles, and the prover displays knowledge of them simply by providing the oracles to the verifier. As a result, at our level of abstraction, knowledge soundness follows automatically from soundness. When the oracles are simulated by a concrete cryptographic tool, knowledge soundness becomes an important consideration that is not automatically satisfied. However, this cryptographic instantiation is beyond the scope of this paper.

A proof system is zero-knowledge [9] if, informally, an authentic transcript could have been produced by an adversary who is ignorant of the witness. More formally, the distribution of authentic transcripts must be sampleable with public information only.

Definition 3 (Honest-Verifier Zero-Knowledge). *Let $\mathcal{R}$ be an indexed relation and let* (P, V) *be a proof system for $\mathcal{R}$. Let $tr \leftarrow \langle \mathsf{P}(\mathbb{i}, \mathbb{x}, \mathbb{w}), \mathsf{V}(\mathbb{i}, \mathbb{x}) \rangle$ denote the assignment to the variable tr of the transcript arising from the interaction between* P *with input* $(\mathbb{i}, \mathbb{x}, \mathbb{w})$ *and* V *with input* $(\mathbb{i}, \mathbb{x})$*. The proof system* (P, V) *is* honest-verifier zero-knowledge *if there exists a polynomial-time Turing machine* S *such that the distribution $\mathcal{D}_0$ of authentic transcripts $tr \leftarrow \langle \mathsf{P}(\mathbb{i}, \mathbb{x}, \mathbb{w}), \mathsf{V}(\mathbb{i}, \mathbb{x}) \rangle$, is identical to the distribution $\mathcal{D}_1$ of simulated transcripts $tr \leftarrow \mathsf{S}(\mathbb{i}, \mathbb{x})$. When $\mathcal{D}_0$ and $\mathcal{D}_1$ are distinct, we consider the statistical distance and use the term* Statistical *Honest-Verifier Zero-Knowledge.*

2.4 Polynomial IOP

Informally, a Polynomial IOP is an abstract proof system, where the prover sends polynomials and the verifier, instead of reading the polynomials in their entirety, is allowed to query the polynomial as oracles in select points.

Definition 4 (Polynomial IOP). *Let $\mathcal{R}$ be an indexed relation with corresponding indexed language $\mathcal{L}(\mathcal{R})$, $\mathbb{F}$ some finite field, and $\mathsf{d} \in \mathbb{N}$ a degree bound. A* Polynomial IOP for $\mathcal{R}$ with degree bound d *is a pair of interactive machines* (P, V)*, satisfying the following description.*

- (P, V) *is an interactive proof for $\mathcal{L}(\mathcal{R})$ with r rounds, and with soundness error σ.*
- P *sends polynomials $f_i(X) \in \mathbb{F}[X]$ of degree at most d to V.*
- V *is an oracle machine with access to a list of oracles, which contains one oracle for each polynomial it has received from the prover.*
- *When an oracle associated with a polynomial $f_i(X)$ is queried on a point $z_j \in \mathbb{F}$, the oracle responds with the value $f_i(z_j)$.*
- V *sends challenges $\alpha_k \in \mathbb{F}$ to P.*
- V *is public coin.*

While we decided in favor of the shorter variant here, it is possible to modify the definition so as to allow a unique degree bound d_i for every polynomial $f_i(X)$, which is determined as a function of $\mathbb{i}$. The present simplification does not degrade generality as long as $\mathsf{d} \geq \max_i \mathsf{d}_i$. To see this, observe that P can always send $X^{\mathsf{d}-\mathsf{d}_i} \cdot f_i(X)$ instead of $f_i(X)$, in which case V should divide the obtained oracle response by $z^{\mathsf{d}-\mathsf{d}_i}$. It is easy to see that this transformation retains completeness. Soundness is retained because $f_i^*(X) \neq X^{\mathsf{d}-\mathsf{d}_i} \cdot f_i(X)$ agrees with $X^{\mathsf{d}-\mathsf{d}_i} \cdot f_i(X)$ in at most d points. As a result, if an identity involving $f_i(X)$ is tested, it will hold for $f_i^*(X)$ with probability at most $\mathsf{d}/|\mathbb{F}|$.

With a minor extension, Polynomial IOPs can appropriately capture preprocessing. This extension introduces third machine, the *indexer* I. As its name suggest, I reads only $\mathbb{i}$, and it outputs a list of polynomials to which V has oracle access.

Definition 5 (Polynomial IOP with Preprocessing). *Let $\mathcal{R}$ be an indexed relation with corresponding language $\mathcal{L}(\mathcal{R})$. A* Polynomial IOP with Preprocessing *is a tuple of interactive machines* $(\mathsf{I}, \mathsf{P}, \mathsf{V})$ *such that* (P, V) *is a Polynomial IOP for $\mathcal{L}(\mathcal{R})$ and such that*

- I *takes $\mathbb{i}$ for input and outputs a list of polynomials of degree at most d;*
- V *has oracle access to these polynomials in addition to the polynomials it receives from* P*.*

Some of the Polynomial IOPs in this paper are designed for modular composition. As a result, V does not begin with an empty list of polynomial oracles. In order to define the relations that these Polynomial IOPs realize, we denote by $[f_i(X)]$ a polynomial $f_i(X)$ that was sent to V by I or P at some earlier stage and to which V has oracle access.

3 Polynomial IOPs for Linear Algebra Relations

The protocols introduced in this paper rely on protocols propoed with varying degrees of formality by Bünz *et al.* [6]. For the sake of completeness, we provide a formal treatment of these building blocks in the appendix. Here we summarize the main qualities needed for the subsequent constructions.

- Flip establishes that the coefficient vector of one polynomial is the reverse of the coefficient vector of another polynomial.
- $\mathsf{CheckCoefficient}$ establishes that the coefficient at a given index of a given polynomial is equal to a given scalar.
- $\mathsf{InnerProduct}$ establishes that the inner product of the coefficient vectors of two polynomials is equal to a given scalar.

3.1 Modular Reduction

We start with a protocol that will be used as a subprotocol in the sequel. This protocol establishes that one polynomial, $r(X)$, is the remainder after division of a second polynomial $f(X)$, by a third, $d(X)$. This third polynomial is assumed to be known, but the protocol can be naturally amended to allow V only oracle access to $[d(X)]$. Formally, the relation is given by

$$\mathcal{R}_{\text{reduce}} = \left\{ (\mathbb{i}, \mathbb{x}, \mathbb{w}) \;\middle|\; \begin{array}{l} \mathbb{i} = (\mathsf{d}_f, \mathsf{d}_r) \\ \mathbb{x} = ([f(X)], [r(X)], d(X)) \\ \mathbb{w} = (f(X), r(X)) \\ \exists q(X) \in \mathbb{F}[X] \,.\, f(X) = q(X) \cdot d(X) + r(X) \\ \mathsf{deg}(f) \leq \mathsf{d}_f \\ \mathsf{deg}(r) \leq \mathsf{d}_r \end{array} \right\} . \quad (3)$$

```
description: decides $\mathcal{L}(\mathcal{R}_{\text{reduce}})$
inputs: $\mathbb{i} : (\mathsf{d}_f, \mathsf{d}_r)$
        $\mathbb{x} : ([f(X)], [r(X)], d(X))$
        $\mathbb{w} : (f(X), r(X))$
begin
    $\mathsf{P}$ computes $q$ such that $f(X) = q(X) \cdot d(X) + r(X)$
    $\mathsf{P}$ sends $q(X)$ of degree at most $\mathsf{d}_f - \mathsf{deg}(d)$ to $\mathsf{V}$
    $\mathsf{V}$ samples $z \overset{\$}{\leftarrow} \mathbb{F}\backslash\{0\}$ and queries $[f(X)]$, $[q(X)]$, and $[r(X)]$ in $z$
    $\mathsf{V}$ receives $y_f = f(z)$, $y_q = q(z)$, and $y_r = r(z)$
    $\mathsf{V}$ tests $y_f \overset{?}{=} y_q \cdot d(z) + y_r$
```

Protocol 1: $\mathsf{ModReduce}$

Theorem 1 (Security of $\mathsf{ModReduce}$). *Protocol $\mathsf{ModReduce}$ of Protocol 1 is a Polynomial IOP for $\mathcal{L}(\mathcal{R}_{\text{reduce}})$ with completeness and soundness with soundness error $\sigma = \mathsf{d}_f/|\mathbb{F}|$.*

Proof. completeness follows from construction: dividing $f(X)$ by $d(X)$ gives quotient $q(X)$ and remainder $r(X)$. Therefore, $f(X) = q(X) \cdot d(X) + r(X)$ is an identity of polynomials and guaranteed to hold everywhere including in the point z.

For soundness, observe that when $r(X) \not\equiv f(X) \operatorname{\mathsf{mod}} d(X)$ then $d(X)$ does not divide $f(X) - r(X)$. As a result, $f(X) \neq q(X) \cdot d(X) + r(X)$ is an inequality of polynomials with degree $\mathsf{deg}(d) + \mathsf{deg}(q) = \mathsf{d}_f$. Due to the Schwartz-Zippel lemma, the left and right hand sides can evaluate to the same value in at most d_f choices for z. The probability of V accepting when $r(X) \not\equiv f(X) \operatorname{\mathsf{mod}} d(X)$ is therefore $\sigma = \mathsf{d}_f/|\mathbb{F}|$.

What is left to argue is that P fails to convince V when the congruence $r(X) \equiv f(X) \operatorname{\mathsf{mod}} d(X)$ holds, but $r(X)$ is not equal to the remainder after division of $f(X)$ by $d(X)$. The representatives of the congruence class of $r(X)$ are apart by polynomials of degree at least $\mathsf{deg}(d)$, there is only one representative of degree at most $\mathsf{d}_r < \mathsf{deg}(d)$. The index value d_r therefore already constrains $r(X)$ to a unique polynomial. □

3.2 Matrix-Vector Product

The next protocol involves two polynomials that represent vectors in the monomial coefficient basis and one that represents a matrix. Specifically Let $\boldsymbol{a} \in \mathbb{F}^n$ and $\boldsymbol{b} \in \mathbb{F}^m$ and $M \in \mathbb{F}^{m \times n}$ with the element in row i and column j (both indices starting at zero) indexed as $M_{[i,j]}$. These objects are represented as polynomials with $\boldsymbol{a}_{[i]}$ being ith element of $\boldsymbol{a}$ and simultaneously the coefficient of the monomial X^i in $f_{\boldsymbol{a}}(X)$, and similarly for $\boldsymbol{b}, \boldsymbol{b}_{[i]}$, and $f_{\boldsymbol{b}}(X)$. The matrix is encoded in row-first order, specifically $f_M(X) = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} M_{[i,j]} X^{in+j}$. The protocol establishes that $\boldsymbol{b} = M\boldsymbol{a}$. Formally, the relation is given by

$$\mathcal{R}_{\mathrm{mvp}} = \left\{ (\mathbb{i}, \mathbb{x}, \mathbb{w}) \left| \begin{array}{l} \mathbb{i} = (m, n) \\ \mathbb{x} = ([f_{\boldsymbol{a}}(X)], [f_{\boldsymbol{b}}(X)], [f_M(X)]) \\ \mathbb{w} = (f_{\boldsymbol{a}}(X), f_{\boldsymbol{b}}(X), f_M(X)) \\ f_{\boldsymbol{a}}(X) = \sum_{i=0}^{n-1} \boldsymbol{a}_{[i]} X^i \text{ for some } \boldsymbol{a} \in \mathbb{F}^n \\ f_{\boldsymbol{b}}(X) = \sum_{i=0}^{m-1} \boldsymbol{b}_{[i]} X^i \text{ for some } \boldsymbol{b} \in \mathbb{F}^m \\ f_M(X) = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} M_{[i,j]} X^{in+j} \text{ for some } M \in \mathbb{F}^{m \times n} \\ \boldsymbol{b} = M\boldsymbol{a} \end{array} \right. \right\} . \tag{4}$$

Theorem 2 (Security of MVP). *Protocol* MVP *of Protocol 2 is a Polynomial IOP for $\mathcal{L}(\mathcal{R}_{\mathrm{mvp}})$ with completeness and soundness with soundness error $\sigma = \frac{mn+m+2n-3}{|\mathbb{F}|-1}$.*

```
description: decides $\mathcal{L}(\mathcal{R}_{\mathrm{mvp}})$
inputs: $\mathbb{i} : (m, n)$
        $\mathbb{x} : ([f_{\boldsymbol{a}}(X)], [f_{\boldsymbol{b}}(X)], [f_M(X)])$
        $\mathbb{w} : (f_{\boldsymbol{a}}(X), f_{\boldsymbol{b}}(X), f_M(X))$
begin
    V samples $\alpha \xleftarrow{\$} \mathbb{F}$ and sends $\alpha$ to P
    P computes $r \leftarrow f_M(X) \bmod X^n - \alpha$
    P sends $r(X)$ of degree at most $n-1$ to V
5   P and V run ModReduce with $\mathbb{i}^{(1)} = (mn-1, n-1)$,
     $\mathbb{x}^{(1)} = ([f_M(X)], [r(X)], X^n - \alpha)$, and $\mathbb{w}^{(1)} = (f_M(X), r(X))$
    V queries $[f_{\boldsymbol{b}}(X)]$ in $\alpha$ and receives $y_{\boldsymbol{\alpha}^\intercal \boldsymbol{b}} = f_{\boldsymbol{b}}(\alpha)$
7   P and V run InnerProduct with $\mathbb{i}^{(2)} = n-1$, $\mathbb{x}^{(2)} = ([r(X)], [f_{\boldsymbol{a}}(X)], y_{\boldsymbol{\alpha}^\intercal \boldsymbol{b}})$,
     and $\mathbb{w}^{(2)} = (r(X), f_{\boldsymbol{a}}(X))$
```

Protocol 2: MVP

Proof. Let $\boldsymbol{\alpha}^\intercal = (\alpha^0, \alpha^1, \cdots)$ and $\boldsymbol{r}^\intercal = \boldsymbol{\alpha}^\intercal M$, and consider the equations

$$\boldsymbol{b} = M\boldsymbol{a} \tag{5}$$

$$\boldsymbol{\alpha}^\intercal \boldsymbol{b} = \boldsymbol{\alpha}^\intercal M \boldsymbol{a} \tag{6}$$

$$\sum_{i=0}^{m-1} \alpha^i \boldsymbol{b}_{[i]} = \boldsymbol{r}^\intercal \boldsymbol{a} \tag{7}$$

$$f_{\boldsymbol{b}}(\alpha) = \sum_{i=0}^{n-1} \boldsymbol{r}_{[i]} \boldsymbol{a}_{[i]} \tag{8}$$

$$y_{\boldsymbol{\alpha}^\intercal \boldsymbol{b}} = \mathsf{coeffs}(r(X)) \cdot \mathsf{coeffs}(f_{\boldsymbol{a}}(X)) \tag{9}$$

$$(\mathbb{i}^{(2)}, \mathbb{x}^{(2)}) = (n-1, ([r(X)], [f_{\boldsymbol{a}}(X)], y_{\boldsymbol{\alpha}^\intercal \boldsymbol{b}})) \in \mathcal{L}(\mathcal{R}_{\mathsf{InnerProduct}}) \ . \tag{10}$$

Furthermore, observe that the coefficient vector of $r(X)$ matches $\boldsymbol{r}$, by substituting X^n by α in the expression for $f_M(X)$:

$$\sum_{i=0}^{m-1} \sum_{j=0}^{n-1} M_{[i,j]} X^{in+j} \xrightarrow{X^n \mapsto \alpha} r(X) = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} M_{[i,j]} \alpha^i X^j \tag{11}$$

$$= \sum_{j=0}^{n-1} \left(\sum_{i=0}^{m-1} M_{[i,j]} \alpha^i \right) X^j \tag{12}$$

$$= \sum_{j=0}^{n-1} \boldsymbol{r}_{[j]} X^j \ . \tag{13}$$

Completeness follows from the implications (5) $\Rightarrow$ (6) $\Leftrightarrow$ (7) $\Leftrightarrow$ (8) $\Rightarrow$ (9) $\Rightarrow$ (10).

For soundness, there are 3 events that can cause V to accept despite $\boldsymbol{b} \neq M\boldsymbol{a}$:

1. (5) $\not\Leftarrow$ (6). The probability of this event is at most $\frac{m-1}{|\mathbb{F}|-1}$ due to the Schwartz-Zippel lemma.

2. (8) $\not\Leftarrow$ (9) because $r(X)$ is not the remainder of $f_M(X)$ after division by $X^n - \alpha$. The probability of this event is at most $\frac{mn-1}{|\mathbb{F}|}$, the soundness error of $\mathsf{ModReduce}$.
3. (9) $\not\Leftarrow$ (10), because $y_{\boldsymbol{\alpha}^\intercal \boldsymbol{b}}$ is not the inner product of the coefficient vectors of $r(X)$ and $f_{\boldsymbol{a}}(X)$. The probability of this event is at most $\frac{2n-1}{|\mathbb{F}|}$, the soundness error of $\mathsf{InnerProduct}$.

It follows that the soundness error of MVP is bounded by $\sigma = \frac{mn+m+2n-3}{|\mathbb{F}|-1}$. □

3.3 Matrix-Matrix Product

The following protocol for verifying a matrix-matrix product relies on the fact that a matrix identity $AB = C$ satisfies $\boldsymbol{\alpha}^\intercal AB\boldsymbol{\beta} = \boldsymbol{\alpha}^\intercal C\boldsymbol{\beta}$ for all compatible vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. In the protocol, V supplies scalars $\alpha, \beta \in \mathbb{F}\backslash\{0\}$, which determine the vectors $\boldsymbol{\alpha} = (\alpha^i)_i$ and $\boldsymbol{\beta} = (\beta^i)_i$ of the appropriate dimensions. Formally, the relation is given by

$$\mathcal{R}_{\text{mmp}} = \left\{ (\mathbb{i}, \mathbb{x}, \mathbb{w}) \left| \begin{array}{l} \mathbb{i} = (m, n, o) \\ \mathbb{x} = ([f_A(X)], [f_B(X)], [f_C(X)]) \\ \mathbb{w} = (f_A(X), f_B(X), f_C(X)) \\ f_A(X) = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} A_{[i,j]} X^{in+j} \text{ for some } A \in \mathbb{F}^{m\times n} \\ f_B(X) = \sum_{i=0}^{n-1} \sum_{j=0}^{o-1} B_{[i,j]} X^{io+j} \text{ for some } B \in \mathbb{F}^{n\times o} \\ f_C(X) = \sum_{i=0}^{m-1} \sum_{j=0}^{o-1} C_{[i,j]} X^{io+j} \text{ for some } C \in \mathbb{F}^{m\times o} \\ AB = C \end{array} \right. \right\} . \tag{14}$$

Theorem 3 (Security of MMP). *Protocol MMP of Protocol 3 is a Polynomial IOP for $\mathcal{L}(\mathcal{R}_{\text{mmp}})$ with completeness and soundness with soundness error $\sigma = \frac{mn+mo+no+2m+3n+5o-11}{|\mathbb{F}|-1}$.*

Proof. Consider the sequence of equations

$$AB = C \tag{15}$$
$$\boldsymbol{\alpha}^\intercal AB = \boldsymbol{\alpha}^\intercal C \tag{16}$$
$$\boldsymbol{\alpha}^\intercal AB\boldsymbol{\beta} = \boldsymbol{\alpha}^\intercal C\boldsymbol{\beta} \tag{17}$$
$$\mathsf{coeffs}(f_{\boldsymbol{\alpha}^\intercal A}(X))^\intercal B\boldsymbol{\beta} = \boldsymbol{\alpha}^\intercal C\boldsymbol{\beta} \tag{18}$$
$$\mathsf{coeffs}(f_{\boldsymbol{\alpha}^\intercal A}(X))^\intercal \mathsf{coeffs}(f_{B\boldsymbol{\beta}}(X)) = \boldsymbol{\alpha}^\intercal C\boldsymbol{\beta} \tag{19}$$
$$\gamma = \boldsymbol{\alpha}^\intercal C\boldsymbol{\beta} \tag{20}$$
$$\gamma = \boldsymbol{\alpha}^\intercal \mathsf{coeffs}(f_{C\boldsymbol{\beta}}(X)) \tag{21}$$
$$\gamma = f_{C\boldsymbol{\beta}}(\alpha) \ . \tag{22}$$

Completeness follows from the implication (15) $\Rightarrow$ (16) $\Rightarrow$ (17) $\Rightarrow$ (18) $\Rightarrow$ (19) $\Rightarrow$ (20) $\Rightarrow$ (21) $\Leftrightarrow$ (22).

For soundness, consider when the reverse implications fail.

```
description: decides $\mathcal{L}(\mathcal{R}_{\text{mmp}})$
inputs: $\mathbb{i}$: $m, n, o$
        $\mathbb{x}$: $[f_A(X)], [f_B(X)], [f_C(X)]$
        $\mathbb{w}$: $f_A(X), f_B(X), f_C(X)$
begin
    V samples $\alpha, \beta \xleftarrow{\$} \mathbb{F}\backslash\{0\}$ and sends $\alpha, \beta$ to P
    P computes $f_{\boldsymbol{\alpha}^\mathsf{T} A} \leftarrow f_A(X) \mathsf{~mod~} (X^n - \alpha)$
    P sends $f_{\boldsymbol{\alpha}^\mathsf{T} A}(X)$ of degree at most $n+1$ to V
    P and V run ModReduce with $\mathbb{i}^{(1)} = (mn-1, n-1)$,
      $\mathbb{x}^{(1)} = ([f_A(X)], [f_{\boldsymbol{\alpha}^\mathsf{T} A}(X)], X^n - \alpha)$ and $\mathbb{w}^{(1)} = (f_A(X), f_{\boldsymbol{\alpha}^\mathsf{T} A}(X))$
    P computes $f_{B\boldsymbol{\beta}} \leftarrow \sum_{i=0}^{n-1} \left( \sum_{j=0}^{o-1} B_{[i,j]} \beta^j \right) X^i$ and $f_{\boldsymbol{\beta}} \leftarrow \sum_{i=0}^{o-1} \beta^i X^i$, and
      sends $f_{B\boldsymbol{\beta}}(X)$ of degree at most $o-1$ to V
    P and V run MVP with $\mathbb{i}^{(2)} = (n, o)$, $\mathbb{x}^{(2)} = ([f_{\boldsymbol{\beta}}(X)], [f_{B\boldsymbol{\beta}}(X)], [f_B(X)])$
      and $\mathbb{w}^{(2)} = (f_{\boldsymbol{\beta}}(X), f_{B\boldsymbol{\beta}}(X), f_B(X))$ and with V computing queries to
      $[f_{\boldsymbol{\beta}}(X)]$ locally using $f_{\boldsymbol{\beta}}(X) = \sum_{i=0}^{o-1} \beta^i X^i = (1 - (\beta X)^o)/(1 - \beta X)$
    P computes $\gamma \leftarrow \boldsymbol{\alpha}^\mathsf{T} A B \boldsymbol{\beta}$ and sends $\gamma$ (of degree 0) to V
    P and V run InnerProduct with $\mathbb{i}^{(3)} = n$, $\mathbb{x}^{(3)} = ([f_{\boldsymbol{\alpha}^\mathsf{T} A}(X)], [f_{B\boldsymbol{\beta}}(X)], \gamma)$,
      and $\mathbb{w}^{(3)} = (f_{\boldsymbol{\alpha}^\mathsf{T} A}(X), f_{B\boldsymbol{\beta}}(X))$
    P computes $f_{C\boldsymbol{\beta}}(X)$, whose vector of coefficients is $C\boldsymbol{\beta}$
    P sends $f_{C\boldsymbol{\beta}}(X)$, of degree at most $m-1$, to V
    P and V run MVP with $\mathbb{i}^{(4)} = (m, o)$, $\mathbb{x}^{(4)} = ([f_{\boldsymbol{\beta}}(X)], [f_{C\boldsymbol{\beta}}(X)], [f_C(X)])$,
      and $\mathbb{w}^{(4)} = (f_{\boldsymbol{\beta}}(X), f_{C\boldsymbol{\beta}}(X), f_C(X))$, and where V simulates $[f_{\boldsymbol{\beta}}(X)]$
      locally using $f_{\boldsymbol{\beta}}(X) = (1 - (\beta X)^o)/(1 - \beta X)$
    V queries $[f_{C\boldsymbol{\beta}}(X)]$ in $\alpha$ and receives $y_{\boldsymbol{\alpha}^\mathsf{T} C\boldsymbol{\beta}} = f_{C\boldsymbol{\beta}}(\alpha)$
    V tests $\gamma \overset{?}{=} y_{\boldsymbol{\alpha}^\mathsf{T} C\boldsymbol{\beta}}$
```

Protocol 3: MMP

- (15) $\not\Leftarrow$ (16). This event occurs with probability at most $\frac{m-1}{|\mathbb{F}|-1}$ due to the Schwarz-Zippel lemma.
- (16) $\not\Leftarrow$ (17) This event occurs with probability at most $\frac{o-1}{|\mathbb{F}|-1}$ due to the Schwarz-Zippel lemma.
- (17) $\not\Leftarrow$ (18), because $f_{\boldsymbol{\alpha}^\mathsf{T} A}(X)$ is not the remainder of $f_A(X)$ after division by $X^n - \alpha$. This event occurs with probability at most $\frac{mn-1}{|\mathbb{F}|-1}$, the soundness error of $\mathsf{ModReduce}$.
- (18) $\not\Leftarrow$ (19), because the coefficient vector of $f_{B\boldsymbol{\beta}}(X)$ is not $B\boldsymbol{\beta}$. The probability of this event is at most $\frac{no+n+2o-3}{|\mathbb{F}|-1}$, the soundness error of MVP.
- (19) $\not\Leftarrow$ (20), because γ is not the inner product of the coefficient vectors of $f_{\boldsymbol{\alpha}^\mathsf{T}}(X)$ and $f_{B\boldsymbol{\beta}}(X)$. This event occurs with probability at most $\frac{2n-2}{|\mathbb{F}|-1}$, the soundness error of $\mathsf{InnerProduct}$.
- (20) $\not\Leftarrow$ (21), because the vector of coefficients of $f_{C\boldsymbol{\beta}}(X)$ is not $C\boldsymbol{\beta}$. This event occurs with probability at most $\frac{mo+m+2o-3}{|\mathbb{F}|-1}$, the soundness error of MVP.

It follows that the soundness error of MMP is at most

$$\sigma = \frac{m-1+o-1+mn-1+no+n+2o-3+2n-2+mo+m+2o-3}{|\mathbb{F}|-1} \tag{23}$$

$$= \frac{mn+mo+no+2m+3n+5o-11}{|\mathbb{F}|-1} \quad . \tag{24}$$

$\square$

3.4 Hadamard Product

The next protocol establishes that the Hadamard (or component-wise) product of two vectors is equal to a third. These vectors are represented as the coefficient vectors of polynomials $f_{\boldsymbol{a}}(X)$, $f_{\boldsymbol{b}}(X)$, and $f_{\boldsymbol{c}}(X)$ such that $\boldsymbol{c} = \boldsymbol{a} \circ \boldsymbol{b}$ and $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c} \in \mathbb{F}^{\mathsf{d}+1}$. The protocol relies on the fact that when the elements of the vectors $\boldsymbol{a}$, $\boldsymbol{b}$, and $\boldsymbol{c}$ are arranged on the diagonal of matrices $M_{\boldsymbol{a}}$, $M_{\boldsymbol{b}}$, and $M_{\boldsymbol{c}}$, then $M_{\boldsymbol{a}} M_{\boldsymbol{b}} = M_{\boldsymbol{c}}$ and also $\boldsymbol{\alpha}^\mathsf{T} M_{\boldsymbol{a}} M_{\boldsymbol{b}} \boldsymbol{\beta} = \boldsymbol{\alpha}^\mathsf{T} M_{\boldsymbol{c}} \boldsymbol{\beta}$ for all vectors $\boldsymbol{\beta}, \boldsymbol{\alpha}$. In other words, one can simply sample a random pair of scalars $\alpha, \beta \xleftarrow{\$} \mathbb{F}$, and check the inner product of $\boldsymbol{\alpha}^\mathsf{T} M_{\boldsymbol{a}}$ with $M_{\boldsymbol{b}} \boldsymbol{\beta}$ against the matrix product of $\boldsymbol{\alpha}^\mathsf{T} M_{\boldsymbol{c}} \boldsymbol{\beta}$. Note that the right hand side of this check amounts to $f_{\boldsymbol{c}}(\alpha\beta)$ and and the operands in the left hand side amount to the coefficient vectors of $f_{\boldsymbol{a}}(\alpha X)$ and $f_{\boldsymbol{b}}(\beta X)$, respectively. Formally, the relation is given by

$$\mathcal{R}_{\text{hadamard}} = \left\{ (\mathbb{i}, \mathbb{x}, \mathbb{w}) \;\middle|\; \begin{array}{l} \mathbb{i} = \mathsf{d} \\ \mathbb{x} = ([f_{\boldsymbol{a}}(X)], [f_{\boldsymbol{b}}(X)], [f_{\boldsymbol{c}}(X)]) \\ \mathbb{w} = (f_{\boldsymbol{a}}(X), f_{\boldsymbol{b}}(X), f_{\boldsymbol{c}}(X)) \\ \forall i \in \{0, \ldots, \mathsf{d}\} \,.\, a_i b_i = c_i \end{array} \right\} \quad . \tag{25}$$

```
description: decides $\mathcal{L}(\mathcal{R}_{\text{hadamard}})$
inputs: $\mathbb{i}$: $\mathsf{d}$
        $\mathbb{x}$: $[f_{\boldsymbol{a}}(X)], [f_{\boldsymbol{b}}(X)], [f_{\boldsymbol{c}}(X)]$
        $\mathbb{w}$: $f_{\boldsymbol{a}}(X), f_{\boldsymbol{b}}(X), f_{\boldsymbol{c}}(X)$
begin
    V samples $\alpha, \beta \xleftarrow{\$} \mathbb{F}\backslash\{0\}$ and sends $\alpha, \beta$ to P
    P evaluates $y \leftarrow f_{\boldsymbol{c}}(\alpha\beta)$
    V queries $[f_{\boldsymbol{c}}(X)]$ in $\alpha\beta$ and receives $y = f_{\boldsymbol{c}}(\alpha\beta)$
    P and V run InnerProduct with $\mathbb{i}^{(1)} = \mathsf{d}$, $\mathbb{x}^{(1)} = ([f_{\boldsymbol{a}}(\alpha X)], [f_{\boldsymbol{b}}(\beta X)], y)$,
      $\mathbb{w}^{(1)} = (f_{\boldsymbol{a}}(\alpha X), f_{\boldsymbol{b}}(\beta X))$, where V simulates $[f_{\boldsymbol{a}}(\alpha X)]$ and $[f_{\boldsymbol{b}}(\beta X)]$
      using $[f_{\boldsymbol{a}}(X)]$ and $[f_{\boldsymbol{b}}(X)]$ and the scalars $\alpha$ and $\beta$
```
Protocol 4: Hadamard

Theorem 4 (Security of Hadamard). *Protocol* Hadamard *of Protocol 4 is a Polynomial IOP for $\mathcal{L}(\mathcal{R}_{\text{hadamard}})$ with completeness and soundness with soundness error $\sigma = 4\mathsf{d}/(|\mathbb{F}| - 1)$.*

Proof. Consider the following sequence of equations.

$$\boldsymbol{a} \circ \boldsymbol{b} = \boldsymbol{c} \tag{26}$$

$$M_{\boldsymbol{a}} M_{\boldsymbol{b}} = M_{\boldsymbol{c}} \tag{27}$$

$$\boldsymbol{\alpha}^{\mathsf{T}} M_{\boldsymbol{a}} M_{\boldsymbol{b}} = \boldsymbol{\alpha}^{\mathsf{T}} M_{\boldsymbol{c}} \tag{28}$$

$$\boldsymbol{\alpha}^{\mathsf{T}} M_{\boldsymbol{a}} M_{\boldsymbol{b}} \boldsymbol{\beta} = \boldsymbol{\alpha}^{\mathsf{T}} M_{\boldsymbol{c}} \boldsymbol{\beta} \tag{29}$$

$$\sum_{i=0}^{\mathsf{d}} (\alpha^i \boldsymbol{a}_{[i]})(\boldsymbol{b}_{[i]} \beta^i) = \sum_{i=0}^{\mathsf{d}} \alpha^i \boldsymbol{c}_{[i]} \beta^i \tag{30}$$

$$\mathsf{coeffs}(f_{\boldsymbol{a}}(\alpha X)) \cdot \mathsf{coeffs}(f_{\boldsymbol{b}}(\beta X)) = f_{\boldsymbol{c}}(\alpha\beta) \tag{31}$$

$$\mathsf{coeffs}(f_{\boldsymbol{a}}(\alpha X)) \cdot \mathsf{coeffs}(f_{\boldsymbol{b}}(\beta X)) = y \tag{32}$$

$$(\mathbb{i}, \mathbb{x}) = (\mathsf{d}, ([f_{\boldsymbol{a}}(X)], [f_{\boldsymbol{b}}(X)], y)) \in \mathcal{L}(\mathcal{R}_{\mathsf{InnerProduct}}) \tag{33}$$

Completeness follows from the sequence of implications (26) $\Leftrightarrow$ (27) $\Rightarrow$ (28) $\Rightarrow$ (29) $\Leftrightarrow$ (30) $\Leftrightarrow$ (31) $\Leftrightarrow$ (32) $\Rightarrow$ (33).

For soundness, consider when the reverse implications fail.

- (27) $\not\Leftarrow$ (28). This event happens with probability at most $\mathsf{d}/(|\mathbb{F}| - 1)$ due to the Schwartz-Zippel lemma.
- (28) $\not\Leftarrow$ (29). This event happens with probability at most $\mathsf{d}/(|\mathbb{F}| - 1)$ due to the Schwartz-Zippel lemma.
- (32) $\not\Leftarrow$ (33). This event happens with probability at most $2\mathsf{d}/(|\mathbb{F}| - 1)$, the soundness error of InnerProduct.

Therefore, the probability that V accepts even though $\boldsymbol{a} \circ \boldsymbol{b} \neq \boldsymbol{c}$ is bounded by $\sigma = 4\mathsf{d}/(|\mathbb{F}| - 1)$. □

4 A Polynomial IOP for Arithmetic Circuits

4.1 The Protocol

The next protocol, Protocol 5 puts many of the previously developed tools together into a Polynomial IOP (with preprocessing) for arithmetic circuits as captured by the HPR. To differentiate our protocol from other similar ones, we name it Claymore.

```
description: realizes $\mathcal{R}_{\text{HPR}}$
inputs: $\mathbb{i}$: $(m, n, M)$ with $M \in \mathbb{F}^{m\times n}$
        $\mathbb{x}$: $\mathbf{x} \in \mathbb{F}^n$
        $\mathbb{w}$: $(\mathbf{w_l}, \mathbf{w_r}, \mathbf{w_o}) \in \mathbb{F}^n \times \mathbb{F}^n \times \mathbb{F}^n$
// preprocessing
begin
    I computes $f_M(X)$ whose coefficients correspond to $M$
    I sends $f_M(X)$ of degree at most $m(3n+1)-1$ to V and P
// online
begin
    P computes $f_{wl} \leftarrow \sum_{j=0}^{n-1} \mathbf{w_l}_{[j]} X^j$, $f_{wr} \leftarrow \sum_{j=0}^{n-1} \mathbf{w_r}_{[j]} X^j$, and
      $f_{wo} \leftarrow \sum_{j=0}^{n-1} \mathbf{w_o}_{[j]} X^j$
    P sends $f_{wl}(X)$, $f_{wr}(X)$, and $f_{wo}(X)$, all of degrees at most $n-1$, to V
    P computes $f_{1w}(X) \leftarrow 1 + X f_{wl}(X) + X^{n+1} f_{wr}(X) + X^{2n+1} f_{wo}(X)$
    P computes $f_x(X)$, whose coefficient vectors correspond to
      $\mathbf{x} = M\,(1\|\mathbf{w_l}^\mathsf{T}\|\mathbf{w_r}^\mathsf{T}\|\mathbf{w_o}^\mathsf{T})^\mathsf{T}$
    P and V run MVP with $\mathbb{i}^{(1)} = (m, n)$, $\mathbb{x}^{(1)} = ([f_{1w}(X)], [f_x(X)], [f_M(X)])$,
      $\mathbb{w}^{(1)} = (f_{1w}(X), f_x(X), f_M(X))$ where V simulates $[f_{1w}(X)]$ using
      $f_{1w}(X) = 1 + X f_{wl}(X) + X^{n+1} f_{wr}(X) + X^{2n+1} f_{wo}(X)$, $[f_{wl}(X)]$,
      $[f_{wr}(X)]$, and $[f_{wo}(X)]$; and where V computes $[f_x(X)]$ locally using
      $\mathbf{x} = \mathbb{x}$
    P and V run Hadamard with $\mathbb{i}^{(2)} = n-1$,
      $\mathbb{x}^{(2)} = ([f_{wl}(X)], [f_{wr}(X)], [f_{wo}(X)])$, $\mathbb{w}^{(2)} = (f_{wl}(X), f_{wr}(X), f_{wo}(X))$
```

Protocol 5: Claymore

Theorem 5 (Security of Claymore). *Protocol* Claymore *of Protocol 5 is a Polynomial IOP for $\mathcal{R}_{\text{HPR}}$ with completeness and soundness error $\sigma = \frac{mn+m+6n-7}{|\mathbb{F}|-1}$.*

Proof. Completeness follows from construction. Since the arguments are computed honestly, the subprotocols succeed and guarantee equalities (1) and (2), respectively.

Soundness. If the HPR instance is a false instance, then $\mathbf{x} \neq M(1|\mathbf{w_i}^\mathsf{T}|\mathbf{w_o}^\mathsf{T})^\mathsf{T}$ or $\mathbf{w_l} \circ \mathbf{w_r} \neq \mathbf{w_o}$. As a result either the Hadamard protocol succeeds despite being run on a false instance, or the MVP protocol succeeds despite being run on a false instance. The probabilities of these events are respectively $\sigma_{\mathsf{Hadamard}} = (4n-4)/(|\mathbb{F}|-1)$ and $\sigma_{\mathsf{MVP}} = (mn+m+2n-3)/(|\mathbb{F}|-1)$. The probability that one or both of these events happens is smaller than or equal to the sum of their probabilities: $\sigma = \frac{mn+m+6n-7}{|\mathbb{F}|-1}$. □

4.2 Without Preprocessing

The preprocessing phase can be omitted. In this case, V must compute $f_M(X)$ locally. This task requires $O(mn)$ work, or only $O(k)$ if the matrix M has only k nonzero elements and is represented as such. When this phase is omitted, Claymore should be compared to the Polynomial IOP underlying Aurora [4].

4.3 Optimizations

Let's make a few changes to optimize the protocol. Specifically, the target is to minimize the number of polynomials in the transcript, their degree, the number of polynomial evaluations, and the number of distinct points for evaluation. This list of optimizations is not necessarily exhaustive.

1. Concatenate $\mathbf{w_l}$ and $\mathbf{w_r}$. Specifically, let P send only one polynomial $f_{wi}(X)$ whose vector of coefficients matches with $(\mathbf{w_l}^\mathsf{T}|\mathbf{w_r}^\mathsf{T})$. To make the Hadamard test work, adapt the subprotocol to test the Hadamard product of the (coefficient vectors of the) degree $3n-1$ polynomials $f_{wi}(X)$ and $X^n f_{wi}(X)$ is equal to (the coefficient vector of) $X^n f_{wo}(X)$. This optimization saves 1 polynomial.
2. Unroll the protocol. Specifically, replace every subprotocol invocation with its proper code. While this already happens implicitly, making this step explicit highlights the next steps for optimization.
3. Eliminate the first Flip subprotocol. Specifically, omit $f^{(rv)}(X)$ simply by querying $f(X)$ in X^{-1} instead, and multiplying the result by X^{d}. In this context, $f(X) = r(X)$ and $\mathsf{d} = n-1$, but in fact it pays to flip $f(X) = f_{1w}(X)$ instead.
4. In the MVP protocol, postpone the reduction mudulo $X^{3n+1} - \alpha$ until after $f_M(X)$ is multiplied by $f_{1w}(X)$. This reversal of operations does not change the value of the coefficient c of X^{3n}, which is what needs to be checked. To see this, observe that

$$r(X) \cdot f_{1w}(X^{-1}) \cdot X^{3n} = f_L(X) + c \cdot X^{3n} + X^{3n+1} \cdot f_R(X) \tag{34}$$

$$\equiv f_M(X) \cdot f_{1w}(X^{-1}) \cdot X^{3n} \pmod{X^{3n+1} - \alpha} \tag{35}$$

$$\equiv f_L(X) + y_\alpha \cdot X^{3n} + \alpha \cdot f_R(X) \pmod{X^{3n+1} - \alpha} \tag{36}$$

   and that the degrees of $f_L(X)$ and $f_R(X)$ are at most $3n-1$.
5. In the first invocation of CheckCoefficient, eliminate $f_R(X)$ as it is identically zero (only the top coefficient is being checked now). Moreover, merge the CheckCoefficient identity with the earlier identity of MVP; this merger eliminates the need to transmit one polynomial, namely $r(X)$.

6. Drop the Flip subprotocol from the the second invocation of InnerProduct. Merge the relations $f^{(rv)}(X) = X^{\mathsf{d}} f(X^{-1})$ and $f^{(rv)}(X) \cdot g(X) = h(X)$ and $h(X) = f_L(X) + X^{\mathsf{d}} \cdot c + X^{\mathsf{d}+1} \cdot f_R(X)$ by substituting the first expression into the second, the second into the third, and drop the transmission of $f^{(rv)}(X)$ and $h(X)$ altogether.
7. At this point there are two polynomial identity tests left. They can be merged as well — perhaps in more than one way, but we describe only one. To avoid naming conflicts, rename the α variable of the MVP subprotocol to γ. Move the MVP test to after the Hadamard test, but without changing the order of messages in the transcript. Observe that both tests require the evaluation of $[f_{wo}(X)]$ in a random point, namely $f_{wo}(\alpha\beta)$ for Hadamard and $f_{wo}(z)$ for MVP. By Setting $z = \alpha\beta$, this point is recycled. Moreover, both tests can be rewritten into one where $f_{wo}(\alpha\beta)$ is the left hand side. Equating the right hand sides allows to eliminate $f_{wo}(X)$ altogether.

The reason why merging polynomial identities as described does not affect soundness is because inequalities propagate across substitution. For example, if $f^{(rv)}(X) \neq X^{\mathsf{d}} f(X^{-1})$ or $f^{(rv)}(X) \cdot g(X) \neq h(X)$, then either $X^{\mathsf{d}} f(X^{-1}) \cdot g(X) \neq h(X)$ or else $X^{\mathsf{d}} f(X^{-1}) \cdot g(X) = h(X)$ even though $f^{(rv)}(X) \neq X^{\mathsf{d}} f(X^{-1})$ and $f^{(rv)}(X) \cdot g(X) \neq h(X)$. The point is that we do not care about the second possibility because $f^{(rv)}(X)$ is eliminated and might as well have been defined correctly instead.

For convenience, a fully unrolled protocol with these optimizations is presented in Protocol 7. In the offline phase of this optimized protocol, P sends 1 polynomial of degree at most $m(3n+1)-1$ and no polynomials are evaluated. In the online phase, P sends 5 polynomials and V queries 8 evaluations in 4 distinct points.

4.4 Graph Theoretical Perspective

The previous section provides the high-level steps to go from the rolled-up pseudocode of Protocol 5 to the unrolled and optimized pseudocode of Protocol 7. The result is a difficult to comprehend dump of instructions and a monster formula. Rather than proving that individual every step was performed correctly, we prefer to provide the reader with tools to reinvent the entire sequence of optimizations — and to discover any errors we may have made in the process. This graph theory perspective may have applications to other Polynomial IOPs, so it is described in generic terms.

The idea is to identify a Polynomial IOP with an undirected graph, in which edges represent polynomials and nodes represent identities of polynomials. The identity of a node involves only those polynomials of the edges that enter it. Some polynomials are already known by the verifier (typically, those defined by the index or instance). The other polynomials must either be provided by the prover (or indexer), or the verifier must be capable of simulating their evaluation using polynomial oracles that he already has access to. The verifier accepts if and only if all nodes represent valid identities. For each node, he approximates

```
description: realizes $\mathcal{R}_{\mathrm{HPR}}$
inputs: $\mathbb{i}$: $(m, n, M)$ with $M \in \mathbb{F}^{m \times n}$
        $\mathbb{x}$: $\mathbf{x} \in \mathbb{F}^n$
        $\mathbb{w}$: $(\mathbf{w_l}, \mathbf{w_r}, \mathbf{w_o}) \in \mathbb{F}^n \times \mathbb{F}^n \times \mathbb{F}^n$
        additional parameters: $q$
offline preprocessing:
    I computes $f_M(X)$ whose coefficients correspond to $M$
    I sends $f_M(X)$ of degree at most $m(3n+1)-1$ to V and P
online phase:
    // compute witness polynomial
    P computes $f_{wi}(X)$ whose vector of coefficients corresponds to $(\mathbf{w_l}^{\mathsf{T}} \| \mathbf{w_r})$
    P computes $f_{wo}(X)$ whose vector of coefficients corresponds to $\mathbf{w_o}$
    P sends $f_{wi}(X)$, of degree at most $2n-1$ to V
    // prepare polynomials for matrix-vector product
    P computes $f_{1w}(X) \leftarrow 1 + X f_{wi}(X) + X^{2n+1} f_{wo}(X)$
    P and V compute $f_x(X)$, whose coefficient vector corresponds to $\mathbf{x}$
    // Matrix-Vector Product Relation:
        V samples $\gamma \xleftarrow{\$} \mathbb{F}\backslash\{0\}$ and sends $\gamma$ to P
        P computes $f_{M\boldsymbol{a}} \leftarrow f_M(X) \cdot f_{1w}(X)$ and $r$ and $Q$ such that
          $f_{M\boldsymbol{a}}(X) = Q(X) \cdot (X^{3n+1} - \gamma) + r(X)$ and $\mathsf{deg}(r) \leq 3n$
        P computes $f_L^* \leftarrow r(X) \bmod X^{3n+3q}$
        P sends $f_L^*(X)$ of degree at most $3n-1$, and $Q(X)$ of degree at most
          $m(3n+1)$ to V
        V computes $y_\gamma = f_x(\gamma) = \sum_{i=0}^{m-1} x_i \gamma^i$
    // Hadamard relation:
        V samples $\alpha, \beta \xleftarrow{\$} \mathbb{F}\backslash\{0\}$ and sends $\alpha, \beta$ to P
        P evaluates $y_{\alpha\beta} \leftarrow f_{wo}(\alpha\beta)$
        P computes $f_L$ and $f_R$ such that $\mathsf{deg}(f_L) \leq 3n-1$ and such that
          $f_L(X) + X^{3n} \cdot y_{\alpha\beta} + X^{3n+3q+1} \cdot f_R(X) =$
          $f_{wi}(\alpha X) \cdot (\beta X)^{3n+3q-1} f_{wi}(\beta^{-1} X^{-1})$
        P sends $f_L(X)$ of degree at most $3n-1$, to V
        P sends $f_R(X)$ of degree at most $2n-2$, to V
        V samples $z_1 \xleftarrow{\$} \mathbb{F}\backslash\{0\}$
        V queries $[f_L(X)]$, $[f_R(X)]$, $[f_{wi}(X)]$, $[f_{wi}(X)]$, in $z_1, z_1, \alpha z_1, \beta^{-1} z_1^{-1}$,
          respectively
        V receives $y_L = f_L(z_1)$, $y_R = f_R(z_1)$, $y_\alpha = f_{wi}(\alpha z_1)$, $y_\beta = f_{wi}(\beta^{-1} z_1^{-1})$
    // Merged Test:
        V queries $[Q(X)], [f_L^*(X)], [f_M(X)], [f_{wi}(X)]$ in $\alpha\beta$
        V receives $y_Q = Q(\alpha\beta), y_L^* = f_L^*(\alpha\beta), y_M = f_M(\alpha\beta)$, and
          $y_{wi} = f_{wi}(\alpha\beta)$
27      V checks $\frac{y_Q \cdot ((\alpha\beta)^{2n} - \gamma) + y_L^* + y_\gamma \cdot (\alpha\beta)^{2n-1} - y_M - \alpha\beta \cdot y_{wi} \cdot y_M}{(\alpha\beta)^{2n+1} \cdot y_M} \stackrel{?}{=}$
          $\frac{-y_L - z_1^{3n+1} \cdot y_R + ((\alpha z_1)^n y_\alpha) \cdot \left((\beta z_1)^{3n-1} y_\beta\right)}{z_1^{3n} \cdot (\alpha\beta)^n}$
```

Protocol 6: OptClaymore

this truth value by sampling left and right hand sides in the same random point, and testing the equality of the returned scalars.

While the structure of the graph is determined in advance, the concrete polynomials that given edges represent may depend on how the protocol unfolds. For example, there may be an edge representing the polynomial $f(\alpha X)$, that is only determined after the verifier supplies $\alpha \in \mathbb{F}$. Conversely, some challenges must only be provided after the prover produces a commitment, which in this setting is also a polynomial; in this case, it is the prover and not the verifier who determines the polynomial of a previously unlabeled edge. Only when all the edges are determined, can the verifier compute the truth value of all nodes. Since the values of edges cannot change after being determined, the verifier can always postpone computing these truth values until the last step of its execution.

It is possible to eliminate polynomials that connect two nodes, provided that the nodes' identities can be rewritten to put the same polynomial in the left hand side. At this point, the two nodes can be replaced by a single one whose defining identity equates the right hand sides of the two original nodes. As a result, the polynomial on the original left hand sides can be eliminated from the equations, from the graph, and from the tests.

As an instructive example, consider the verification graph arising from testing that a polynomial $f(X)$ is simultaneously equal to the product of $g(X)$ and $h(X)$, and also splits as $f_L(X) + X^{\mathsf{d}} f_R(X)$. The two identities are therefore $f(X) = g(X) \cdot h(X)$ and $f(X) = f_L(X) + X^{\mathsf{d}} f_R(X)$. Their merger is $g(X) \cdot h(X) = f_L(X) + X^{\mathsf{T}} f_R(X)$. Note that $f(X)$ has been eliminated from the last equation.

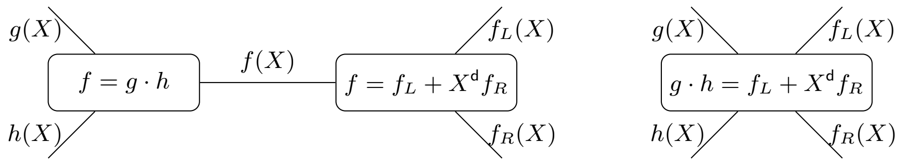

Fig. 1. A Polynomial IOP Graph before (left) and after (right) merging two nodes.

The end result of applying this merge procedure until there are no edges left to eliminate, is a graph in the shape of the a star: it has one node and many outgoing edges with dangling opposite ends.

5 Zero-Knowledge

It turns out that adapting the protocol for zero-knowledge is rather tricky. The naïve approach consists of appending q randomizers to the initial wire vectors $\mathbf{w_l}$, $\mathbf{w_r}$, and $\mathbf{w_o}$. The randomizers will make the witness polynomials q-wise independent, meaning that no distinguisher restricted to at most q queries will obtain any infomation about the witness. Furthermore, the randomizers will propagate to the other polynomials, masking witness information there as well.

Unfortunately, this argument fails in combination with optimization (1) of Section 4.3 and in particular with respect to the polynomial $f_L(X)$. This polynomial does inherit randomizers, but these randomizers stand in nonlinear relation to the ones we started from. As a result, showing that these derived randomizers are uniform or close enough, is a challenging task to say the least.

A solution to this problem is to sacrifice optimization (1) and splitting the witness polynomials $f_{wi}(X)$ back into two polynomials, $f_{wl}(X)$ and $f_{wr}(X)$. Protocol 7 shows the result of dropping this optimization and adding q randomizers to all the witness vectors, starting from the optimized protocol (6). This protocol requires 1 offline polynomial and 6 online ones with maximum degree $m(3n + 3q + 1) - 1$; and 9 evaluations in 4 distinct points.

Theorem 6. *Assume M has rank at least 2. The Polynomial IOP* ZKClaymore *of protocol 7 has statistical zero-knowledge with respect to all distinguishers limited to $q-1$ queries. The success probability of such distinguishers is bounded by $\frac{2m+2q-2}{|\mathbb{F}-1}$.*

Proof. We start by showing how S produces a transcript for $(\mathbb{i}, \mathbb{x})$ without knowledge of $\mathbb{w}$. S chooses a random witness $\mathbb{w}' = (\mathbf{w_l}', \mathbf{w_r}', \mathbf{w_o}') \xleftarrow{\$} \mathbb{F}^n \times \mathbb{F}^n \times \mathbb{F}^n$ subject to $\mathbf{x} = M(1 \,|\, \mathbf{w_l}'^{\mathsf{T}} \,|\mathbf{w_r}'^{\mathsf{T}} \,|\, \mathbf{w_o}'^{\mathsf{T}})^{\mathsf{T}}$ and simulates a protocol execution between $\mathsf{P}(\mathbb{i}, \mathbb{x}, \mathbb{w}')$ and $\mathsf{V}(\mathbb{i}, \mathbb{x})$ with one important difference: S does not choose α, β uniformly at random but such that the equation of line 28 holds. Specifically, S chooses α at random, solves for β, and retries with a new witness if necessary. The polynomial has uniformly random coefficients, so the probability that it has a linear factor over $\mathbb{F}$ approaches a constant greater than $\frac{1}{2}$ as $|\mathbb{F}| \rightarrow \infty$. As a result, S runs in expected polynomial time.

This transcript is accepting by construction. What remains to be shown is that no Turing machine or circuit D can distinguish transcripts produced in this manner from authentic transcripts when restricted $q-1$ queries. We proceed to argue that up to a negligible statistical distance, every polynomial in the transcript that depends on the witness has $q-1$ uniformly random coefficients. As a result, the witness remains perfectly hidden from D.

Decompose the polynomials appearing in the transcript into the sum of data that depend on the instance or index ($\bar{f}$), that depend on the witness ($\hat{f}$), and that depend on the randomizers ($\tilde{f}$). This decomposition gives:

- $f_M = \bar{f}_M$ is independent of the witness.
- $f_{wl} = \hat{f}_{wl} + \tilde{f}_{wl}$, where $\tilde{f}_{wl}$ contains the randomizer sequence $\mathbf{r}^{(l)}$.
- $f_{wr} = \hat{f}_{wr} + \tilde{f}_{wr}$, where $\tilde{f}_{wr}$ contains the randomizer sequence $\mathbf{r}^{(r)}$.
- $f_L^* = \hat{f}_L^* + \tilde{f}_L^*$, where $\tilde{f}_L^*$ contains the convolution of $f_M(X)$ with $X\tilde{f}_{wl}(X) + X^{n+q+1}\tilde{f}_{wr}(X) + X^{2n+2q+1}\tilde{f}_{wo}(X)$, first reduced by $X^{3n+3q+1} - \gamma$ and then reduced by X^{3n+3q}. To show that $\tilde{f}_L^*$ depends linearly on all of $\mathbf{r}^{(l)}$, we consider the following intermediary steps.

```
description: realizes $\mathcal{R}_{\text{HPR}}$
inputs: $\mathbb{i}$: $(m, n, M)$ with $M \in \mathbb{F}^{m \times n}$
        $\mathbb{x}$: $\mathbf{x} \in \mathbb{F}^n$
        $\mathbb{w}$: $(\mathbf{w_l}, \mathbf{w_r}, \mathbf{w_o}) \in \mathbb{F}^n \times \mathbb{F}^n \times \mathbb{F}^n$
        additional parameters: $q$
offline preprocessing:
    I computes $f_M(X)$ whose coefficients correspond to
      $M' = \left( M_{[:,0:(n+1)]} \middle| 0_{m \times q} \middle| M_{[:,(n+1):(2n+1)]} \middle| 0_{m \times q} \middle| M_{[:,(2n+1):(3n+1)]} \middle| 0_{m \times q} \right)$
    I sends $f_M(X)$ of degree at most $m(3n + 3q + 1) - 1$ to V and P
online phase:
    // compute witness polynomial with randomizers
    P samples $\mathbf{r}^{(l)} \xleftarrow{\$} \mathbb{F}^q$ and $\mathbf{r}^{(r)} \xleftarrow{\$} \mathbb{F}^q$
    P computes $f_{wl}(X)$ whose vector of coefficients corresponds to
      $({\mathbf{w_i}}_{[0:n]}^{\mathsf{T}} | \mathbf{r}^{(l)\,\mathsf{T}})$ and $f_{wr}(X)$ whose vector of coefficients corresponds to
      $({\mathbf{w_i}}_{[n:2n]}^{\mathsf{T}} | \mathbf{r}^{(r)\,\mathsf{T}})$
    P computes $f_{wo}(X)$ whose vector of coefficients corresponds to
      $({\mathbf{w_o}}_{[0:n]}^{\mathsf{T}} | \mathbf{r}^{(l)\,\mathsf{T}} \circ \mathbf{r}^{(r)\,\mathsf{T}})$
    P sends $f_{wl}(X)$ and $f_{wr}(X)$, both of degree at most $n + q - 1$, to V
    // prepare polynomials for matrix-vector product
    P computes $f_{1w}(X) \leftarrow 1 + X f_{wl}(X) + X^{n+q+1} f_{rw}(X) + X^{2n+2q+1} f_{wo}(X)$
    P and V compute $f_x(X)$, whose coefficient vector corresponds to $\mathbf{x}$
    // Matrix-Vector Product Relation:
        V samples $\gamma \xleftarrow{\$} \mathbb{F} \backslash \{0\}$ and sends $\gamma$ to P
        P computes $f_{M\boldsymbol{a}} \leftarrow f_M(X) \cdot f_{1w}(X)$ and $r$ and $Q$ such that
          $f_{M\boldsymbol{a}}(X) = Q(X) \cdot (X^{3n+3q+1} - \gamma) + r(X)$ and $\mathsf{deg}(r) \leq 3n + 3q$
        P computes $f_L^* \leftarrow r(X) \bmod X^{3n+3q}$
        P sends $f_L^*(X)$ of degree at most $3n + 3q - 1$, and $Q(X)$ of degree at
          most $m(3n + 3q + 1) - 1$ to V
        V computes $y_\gamma = f_x(\gamma) = \sum_{i=0}^{m-1} x_i \gamma^i$
    // Hadamard relation:
        V samples $\alpha, \beta \xleftarrow{\$} \mathbb{F} \backslash \{0\}$ and sends $\alpha, \beta$ to P
        P evaluates $y_{\alpha\beta} \leftarrow f_{wo}(\alpha\beta)$
        P computes $f_L$ and $f_R$ such that $\mathsf{deg}(f_L) \leq n + q - 1$ and such that
          $f_L(X) + X^{n+q} \cdot y_{\alpha\beta} + X^{n+q+1} \cdot f_R(X) =$
          $f_{wl}(\alpha X) \cdot (\beta X)^{n+q-1} f_{wr}(\beta^{-1} X^{-1})$
        P sends $f_L(X)$ of degree at most $n + q - 1$, to V
        P sends $f_R(X)$ of degree at most $n + q - 1$, to V
        V samples $z_1 \xleftarrow{\$} \mathbb{F} \backslash \{0\}$
        V queries $[f_L(X)]$, $[f_R(X)]$, $[f_{wl}(X)]$, $[f_{wr}(X)]$, in $z_1, z_1, \alpha z_1, \beta^{-1} z_1^{-1}$,
          respectively
        V receives $y_L = f_L(z_1)$, $y_R = f_R(z_1)$, $y_\alpha = f_{wl}(\alpha z_1)$, $y_\beta = f_{wr}(\beta^{-1} z_1^{-1})$
    // Merged Test:
        V queries $[Q(X)], [f_L^*(X)], [f_M(X)], [f_{wl}(X)], [f_{wr}(X)]$ in $\alpha\beta$
        V receives $y_Q = Q(\alpha\beta), y_L^* = f_L^*(\alpha\beta), y_M = f_M(\alpha\beta)$, $y_{wl} = f_{wl}(\alpha\beta)$,
          and $y_{wr} = f_{wr}(\alpha\beta)$
28      V checks
          $\frac{y_Q \cdot ((\alpha\beta)^{2n+2q} - \gamma) + y_L^* + y_\gamma \cdot (\alpha\beta)^{2n+2q-1} - y_M - y_{wl} y_{rw} (\beta z_1)^{n+q-1}) \cdot y_M}{(\alpha\beta)^{2n+2q+1} \cdot y_M} \stackrel{?}{=}$
          $\frac{-y_L - z_1^{n+q+1} \cdot y_R + (\beta z_1)^{n+q-1} y_\alpha y_\beta}{z_1^{n+q}}$
```

Protocol 7: ZKClaymore

- The polynomial

$$X\tilde{f}_{wl}(X) + X^{n+q+1}\tilde{f}_{wr}(X) + X^{2n+2q+1}\tilde{f}_{wo}(X) =$$
$$\sum_{i=0}^{q-1}\mathbf{r}^{(l)}_{[i]}X^{1+n+i} + \sum_{i=0}^{q-1}\mathbf{r}^{(r)}_{[i]}X^{1+2n+q+i} + \sum_{i=0}^{q-1}\mathbf{r}^{(l)}_{[i]}\mathbf{r}^{(r)}_{[i]}X^{1+3n+2q+i}$$

  depends linearly on all of $\mathbf{r}^{(l)}$ by construction.
- The polynomial

$$f_M(X)\cdot(X\tilde{f}_{wl}(X) + X^{n+q+1}\tilde{f}_{wr}(X) + X^{2n+2q+1}\tilde{f}_{wo}(X))$$

  might have fewer nonzero coefficients than either factor. However, the point is that when the randomizers $\mathbf{r}^{(l)}$ are interpreted as *variables*, all of them appear linearly in this product. So this polynomial depends linearly on all of $\mathbf{r}^{(l)}$ as well.
- The polynomial

$$\tilde{r}(X) = f_M(X)\cdot(X\tilde{f}_{wl}(X) + X^{n+q+1}\tilde{f}_{wr}(X) + X^{2n+2q+1}\tilde{f}_{wo}(X))$$
$$\mathsf{mod}\, X^{3n+3q+1} - \gamma$$

  depends linearly on all of $\mathbf{r}^{(l)}$ unless the reduction modulo $X^{3n+3q+1}-\gamma$ induces a cancellation. The probability of this event is at most $m/(|\mathbb{F}|-1)$ due to the Schwartz-Zippel lemma.
- The polynomial $\tilde{f}^*_L(X)$ is the same as the polynomial from the previous bullet point, except without the term in X^{3n+3q}. If there is a randomizer $\mathbf{r}^{(l)}_{[i]}$ that appears only in this term, then either (a) $f_M(X)$ consists of one single term, or (b) the reduction modulo $X^{3n+3q+1}-\gamma$ induced a cancellation. Case (a) is incompatible with M having rank at least 2. Case (b) occures with probability at most $m/(|\mathbb{F}|-1)$ due to the Schwartz-Zippel lemma.

In summary, except with probability at most $\frac{2m}{|\mathbb{F}|-1}$, the polynomial $\tilde{f}^*_L(X)$ depends linearly on all randomizers of $\mathbf{r}^{(l)}$.

– $Q = \hat{Q} + \tilde{Q}$ contains the quotient of dividing the convolution of $f_M(X)$ with $X^{3n+3q+1}\cdot(X^{-1}\tilde{f}_{wi}(X^{-1}) + X^{-2n-2q-1}\tilde{f}_{wo}(X^{-1}))$ by $X^{3n+3q+1}-\gamma$. Specifically, $\tilde{Q}(X)$ satisfies the relation

$$\tilde{Q}(X)\cdot(X^{3n+3q+1}-\gamma) + \tilde{r}(X) = f_M(X)\cdot$$
$$X^{3n+3q+1}\cdot\Big(X^{-1}\tilde{f}_{wl}(X^{-1}) + X^{-n-q-1}\tilde{f}_{wr}(X^{-1}) + X^{-2n-2q-1}\tilde{f}_{wo}(X^{-1})\Big)$$

where $\mathsf{deg}(\tilde{r}(X)) < 3n+3q+1$. Let cX^e be the top term of $f_M(X)$. Then $\tilde{Q}(X)$ has $c\,\mathbf{r}^{(l)}_{[0]}X^{e+2n+3q-1}$ as top term. So $Q(X)$ has at least 1 randomizer. When considering this randomizer fixed, it moves from $\tilde{Q}(X)$ to $\hat{Q}(X)$ and from $\tilde{f}_{wl}(X)$ to $\hat{f}_{wl}(X)$. At this point, the same argument can be repeated to show that the top coefficient of $\tilde{Q}(X)$ depends on $\mathbf{r}^{(l)}_{[1]}$. Repeating the same argument q times establishes that $Q(X)$ has q randomizers.

– $f_L = \hat{f}_L + \tilde{f}_L$ and $f_R = \hat{f}_R + \tilde{f}_R$ are the left and right halves respectively of the polynomial

$$\begin{aligned} f(X) &= f_{wl}(X) \times X^{n+q-1} f_{wr}(X^{-1}) && (37) \\ &= X^{n+q-1}(\hat{f}_{wl}(X) + \tilde{f}_{wl}(X))(\hat{f}_{wr}(X^{-1}) + \tilde{f}_{wr}(X^{-1})) && (38) \\ &= X^{n+q-1}\big(\hat{f}_{wl}(X)\hat{f}_{wr}(X^{-1}) + (\hat{f}_{wl}(X)\tilde{f}_{wr}(X^{-1}) && (39) \\ &\quad + \tilde{f}_{wl}(X)\hat{f}_{wr}(X^{-1}) + \tilde{f}_{wl}(X)\tilde{f}_{wr}(X^{-1})\big) \ . \end{aligned}$$

Specifically, this polynomial is split in order to check the coefficient of X^{n+q}, meaning that f_L has the coefficients to the left of that monomial and f_R the coefficients to the right of it.
The top coefficient of $\tilde{f}_R(X)$ is $\mathbf{r}^{(l)}_{[q-1]}\mathbf{r}^{(r)}_{[q-1]}$, whose distance from uniform is $1/\mathbb{F}$. Consider $\mathbf{r}^{(l)}_{[q-1]}$ and $\mathbf{r}^{(r)}_{[q-1]}$ fixed, so they moves from $\tilde{f}_R(X)$ to $\hat{f}_R(X)$ and from $\tilde{f}_{wl}(X)$ to $\hat{f}_{wl}(X)$ and from $\tilde{f}_{wr}(X)$ to $\hat{f}_{wr}(X)$. Then the top coefficient of $\tilde{f}_R(X)$ becomes $\mathbf{r}^{(l)}_{[q-2]}\mathbf{r}^{(r)}_{[q-2]}$. Repeating the same argument $q-1$ times establishes that $f_R(X)$ has $q-1$ randomizers whose distance from uniform is $1/|\mathbb{F}|$. For the whole sequence, the distance from uniform is $(q-1)/|\mathbb{F}|$. The same argument holds for $f_L(X)$ due to symmetry.

In summary, except with probability at most $\frac{2m+2q-2}{|\mathbb{F}|-1}$, every polynomial in the transcript has $q-1$ uniformly random coefficients. In this case, the witness is perfectly hidden from any collection of $q-1$ queries. Therefore, D's probability of successfully telling authentic and simulated transcripts apart with only $q-1$ queries is bounded by $\frac{2m+2q-2}{|\mathbb{F}-1}$. □

Corollary 1. *Protocol* ZKClaymore *of protocol 7 and parameter $q = 9$ has statistical honest-verifier zero-knowledge with distinguisher probability bounded by* $\frac{2m+2q-2}{|\mathbb{F}-1}$.

6 Dealing with Sparsity

The matrix-vector product protocol MVP requires a representation of an $m \times n$ matrix as a polynomial of degree $mn-1$. The matrices arising from typical constraint systems are not witness data, but rather represent fixed linear transforms on witness data. In this context, an explicit encoding as a list of mn coefficients becomes prohibitively expensive as the size of the witness grows. Furthermore, the matrices arising from constraint systems encoding natural computations are typically sparse, with a number of nonzero elements roughly on the same order as the size of the witness, n. The question therefore arises, is there an analogue of the matrix-vector product protocol that works for sparse matrices?

For arbitrary sparse matrices, such a solution seems to necessarily induce a complexity penalty. However, for sparse matrices exhibiting a particular structure, such a protocol can enjoy a simple description and relatively low performance overhead. The question is whether this restriction limits the expressivity of the underlying constraint system. We argue that rewriting the constraint system to admit a structured linear transform does not pose too big of a problem.

What is needed of a protocol for sparse matrix-vector products? The main fact that is established in the dense protocol, MVP, is that the polynomial $r(X)$ is the remainder after division of the matrix polynomial $f_M(X)$ by $X^n - \alpha$, where $\alpha \in \mathbb{F}\backslash\{0\}$ was supplied by V. The next step is to take the inner product between $r(X)$ and the vector polynomial $f_{\boldsymbol{a}}(X)$. A more general version of this protocol represents the matrix M as the bivariate polynomial $f_M(X,Y) = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} M_{[i,j]} X^i Y^j$. In this variant, V supplies a random α; P sends $r(Y) = f_M(\alpha, Y)$; and then P and V engage in a protocol to establish that $r(Y)$ is consistent with M and α, for instance by showing that $r(z) = f_M(\alpha, z)$ in a random point $z \xleftarrow{\$} \mathbb{F}\backslash\{0\}$. The substitution $Y = X^n$ shows that the univariate polynomial $f_M(X)$ is in fact the natural (dense) univariate simulation of the bivariate polynomial $f_M(X,Y)$. *Any protocol that can establish the correct evaluation of a sparse bivariate polynomial can be used to establish the correct computation of sparse matrix vector products.*

This observation is implicit in Sonic [11]. The bivariate polynomial there decomposes as the sum whose jth term is $\Psi_j(X,Y) = \sum_{i=1}^{n-1} \psi_{j,\sigma_{j,i}} X^i Y^{\sigma_{j,i}}$, where $\psi_{i,j} \in \mathbb{F}$ are coefficients and σ_j is a permutation on $[n]$. The correct evaluation is proved using a permutation argument, which in the end can be realized using univariate polynomials only. The authors furthermore show how the constraint system can be coerced into one such that the linear transformation matches with this format.

We propose a few more solutions for the proof of correct evaluation of a sparse bivariate polynomial below. Each solution presents different requirements for the constraint system. The solutions can be combined, thereby increasing the allowed flexibility.

Diagonal Band The polynomial $f_{\boldsymbol{a}}(X)$, whose vector of coefficients is $\boldsymbol{a} \in \mathbb{F}^n$, can be used to simulate $f_A(X) = f_{\boldsymbol{a}}(X^{n+1})$, which is the polynomial that represents the $n \times n$ diagonal matrix with the elements of $\boldsymbol{a}$ on its diagonal. By taking the sum of b such polynomials, each shifted by one position, one simulates the polynomial $f_M(X) = \sum_{i=0}^{\mathsf{b}-1} X^i f_{\boldsymbol{a}_i}(X^{n+1})$, whose matrix has a diagonal band of width b. The evaluation of the bivariate polynomial $f_M(X,Y)$ is given by

$$f_M(\alpha, z) = \sum_{i=0}^{\mathsf{b}-1} z^i f_{\boldsymbol{a}_i}(\alpha z) \ , \tag{40}$$

and can be computed by querying each of the various $f_{\boldsymbol{a}_i}(X)$ once.

Copy Constraints and Repetitive Constraints Let $f_{\boldsymbol{c}}(X)$ be the polynomial representative of a wire copy constraint, or indeed, any repetitive linear constraint. Let o be the offset between to consecutive repetitions of the constraint. Then the polynomial $f_M(X) = \sum_{i=0}^{k-1} X^{in+i\mathsf{o}} f_{\boldsymbol{c}}(X)$ represents k repetitions of that constraint spaced o elements apart. The bivariate polynomial can be evaluated with

$$f_M(\alpha, z) = \sum_{i=0}^{k-1} \alpha^i z^{i\mathsf{o}} f_{\boldsymbol{c}}(z) \ , \tag{41}$$

which only requires the value of $f_{\boldsymbol{c}}(z)$ in one point. This technique can be extended to matrices in addition to rows.

Explicit Columns Let $f_{\boldsymbol{c}}(X)$ be the polynomial representative of (part of) a column vector appearing in a matrix M, for instance because it has been put into reduced row echelon form. When evaluating, the contribution of this term to $f_M(X, Y)$ can be computed as $f_{\boldsymbol{c}}(\alpha)$.

Isolated Values If the number of isolated nonzero elements without structure is not too great, then it is possible to compute them explicitly. Suppose there is a nonzero value c in column i and row j, then the contribution of this term is simply cX^iY^j. In fact, the offset by (i, j) works for any of the previous techniques as well.

7 Comparison

We compare both variants of Claymore to some other Polynomial IOPs from the literature, namely Sonic, PLONK, Marlin, and Aurora. Of these Polynomial IOPs, the first three give rise to SNARKs after cryptographic compilation. In contrast, Aurora gives rise to a proof system generating short proofs but whose verifier complexity is linear in the size of the witness. Importantly, Claymore is comparable to both types of proof system: with preprocessing, it gives rise to a SNARK; whereas when preprocessing is omitted the proofs remain short but verifier complexity explodes.

Table 1 contains an overview of the comparison. It considers the following key performance indicators for Polynomial IOPs.

- The number of polynomials sent by I during the offline preprocessing phase. This number determines the size of the universal or structured reference strings. While this number contributes to the complexity of I, this complexity is generally speaking not a make or break factor.
- The number of polynomials sent by P during the online proving phase. This number contributes to the size of the proof and to the complexity of both P and V.
- The number of evaluations. This number contributes to the size of the proof, as well as indirectly to the complexity of P and V.
- The number of distinct points for evaluation. Some cryptographic compilers (*e.g.*, [6]) enable the merger of two polynomial evaluations provided that they are being evaluated in the same point. This number limits the number of times this optimization can be applied.

– The maximum degree of all polynomials. This number contributes to indexer and prover complexity in two ways. First, before cryptographic compilation, I and P operate on polynomials of this degree and their complexity is affected accordingly. The exception is if the polynomials are sparse, or otherwise exhibit a structure that enable fast computation. Second, some cryptographic compilers induce overheads that are superlinear in this degree.

Table 1. Comparison between Claymore and other Polynomial IOPs from the literature, with respect to key performance indicators.

	# polynomials offline / online	# evaluations	# distinct points	max. degree
Sonic [11]	$12M/3M+7$	$11M+3$	$9M+2$	$O(n)$
PLONK [8]	8 / 6	7	2	$12(n+a)$
Marlin [7]	9 / 12	18	3	$6k+6$
Aurora [4]	- / 7	8	2	$\max(m,n)$
OptClaymore	1 / 5	8	4	$m(3n+1)-1$
ZkClaymore	1 / 6	9	4	$m(3n+3q+1)-1$

For Sonic, n refers to the number of multiplication gates. However, due to their technique for simulating bivariate polynomials, the addition gates have fan-in bounded by a parameter M. As a result of converting the original circuit into one with this fan-in bound, a number of multiplication gates may have to be added, thus explaining the Landau notation.

For PLONK, n refers to the number of multiplication gates and a refers to the number of addition gates, all of which have fan-in 2. We note that there is a variant of PLONK with larger proofs and smaller prover time, which is not shown in the table.

Aurora does not have a preprocessing phase and as a result the verifier's complexity is linear in the number of nonzero elements in the matrices A, B, C from the R1CS tuple. Marlin uses the same mechanics but uses preprocessing to shrink the verifier's workload for the matrix multiplication; this technique requires 9 polynomials in the uniform or structured reference string (3 per matrix) and a few more in the online protocol. The parameter k denotes the largest number of nonzero elements $\{A, B, C\}$.

Marlin, PLONK, and Aurora work in the Reed-Solomon codeword basis and crucially rely on the structure of the field or of its multiplicative group. In contrast, Sonic and Claymore work for any field.

The table does not reflect Claymore's limited support for sparse linear transforms. In particular, when the matrix of the linear transform has a structure that can be simulated by one or more of the techniques from Section 6, then the maximal degree may be asymptotically on par with the other rows. The price to pay is a greater verifier complexity and the transmission of more than just 1 polynomial in the offline preprocessing phase.

Note that the performance penalty associated with large degree of polynomials that represent sparse matrices is a result of the cryptographic compiler. At the level of the Polynomial IOP, the prover is allowed to represent these polynomials sparsely and operate on them accordingly. One mitigation strategy is to engineer the Polynomial IOP so as to simulate the sparse matrices and their polynomials using low-degree dense ones. Another strategy is to engineer the cryptographic compiler itself so as to reduce this penalty. This latter question is left entirely open as it is out of the scope of this paper.

8 Conclusion

The protocols proposed in this paper challenge the notion that the Reed-Solomon codeword basis is the appropriate basis for representing objects from the arithmetic constraint system in a Polynomial IOP. Instead, the monomial coefficient basis provides a natural and intuitive representation under which the native operations on polynomials are identifiable with matrix operations in the arithmetic constraint system. Moreover, working in this basis does not impose any restrictions on the structure of the field or its multiplicative group.

However, shifting to the monomial coefficient basis does introduce complications in some contexts. For instance, Claymore performs particularly well for arithmetic circuits giving rise to dense linear transforms. However, the arithmetization of long natural computations generally gives rise to sparse linear transforms, and in this respect Claymore's performance is less outstanding. An open question to determine to which degree natural arithmetic constraint systems can be coerced into constraint systems whose linear transforms possess the structural features that allow for efficient simulation. An alternative open question addressing the same problem is to find a method to simulate arbitrary sparse matrices with univariate polynomials whose degree grows linearly in the number of nonzero coefficients of the matrix.

We note that Marlin presents such a solution, based on a clever technique to simulate the matrix's low-degree extension. The question remains whether this technique can be lifted to the monomial coefficient basis. If not, the capacity to deal with sparse matrices constitutes a strong argument in favor of the Reed-Solomon codeword basis and against the monomial coefficient basis.

Two optimizations proposed in this paper are domain-independent. The first eliminates one polynomial by concatenating two witness vectors. The second eliminates another polynomial by implicitizing the third witness vector. While we did not verify that these optimizations carry over to the Reed-Solomon codeword domain, we see no reason why this translation would fail. Furthermore, we noticed that the first optimization seems to break the zero-knowledge property (or at least the proof thereof); in the Reed-Solomon codeword domain, this obstacle may well disappear.

References

1. Ames, S., Hazay, C., Ishai, Y., Venkitasubramaniam, M.: Ligero: Lightweight Sublinear Arguments Without a Trusted Setup. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 2017. pp. 2087–2104. ACM (2017), https://doi.org/10.1145/3133956.3134104
2. Bellare, M., Goldreich, O.: On defining proofs of knowledge. In: Brickell, E.F. (ed.) CRYPTO 1992s. Lecture Notes in Computer Science, vol. 740, pp. 390–420. Springer (1992), https://doi.org/10.1007/3-540-48071-4_28
3. Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Scalable zero knowledge with no trusted setup. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part III. Lecture Notes in Computer Science, vol. 11694, pp. 701–732. Springer (2019). https://doi.org/10.1007/978-3-030-26954-8_23
4. Ben-Sasson, E., Chiesa, A., Riabzev, M., Spooner, N., Virza, M., Ward, N.P.: Aurora: Transparent Succinct Arguments for R1CS. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019 , Part I. Lecture Notes in Computer Science, vol. 11476, pp. 103–128. Springer (2019), https://doi.org/10.1007/978-3-030-17653-2_4
5. Bootle, J., Cerulli, A., Chaidos, P., Groth, J., Petit, C.: Efficient Zero-Knowledge Arguments for Arithmetic Circuits in the Discrete Log Setting. In: Fischlin, M., Coron, J. (eds.) EUROCRYPT 2016, Part II. Lecture Notes in Computer Science, vol. 9666, pp. 327–357. Springer (2016), https://doi.org/10.1007/978-3-662-49896-5_12
6. Bünz, B., Fisch, B., Szepieniec, A.: Transparent SNARKs from DARK Compilers. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020 Part I. Lecture Notes in Computer Science, vol. 12105, pp. 677–706. Springer (2020), https://eprint.iacr.org/2019/1229.pdf
7. Chiesa, A., Hu, Y., Maller, M., Mishra, P., Vesely, N., Ward, N.P.: Marlin: Preprocessing zksnarks with universal and updatable SRS. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020 Part I. Lecture Notes in Computer Science, vol. 12105, pp. 738–768. Springer (2020), https://eprint.iacr.org/2019/1047.pdf
8. Gabizon, A., Williamson, Z.J., Ciobotaru, O.: PLONK: permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. IACR Cryptology ePrint Archive **2019**, 953 (2019), https://eprint.iacr.org/2019/953
9. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems (extended abstract). In: Sedgewick, R. (ed.) ACM STOC. pp. 291–304. ACM (1985). https://doi.org/10.1145/22145.22178
10. Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-size commitments to polynomials and their applications. In: Abe, M. (ed.) ASIACRYPT 2010. Lecture Notes in Computer Science, vol. 6477, pp. 177–194. Springer (2010), https://doi.org/10.1007/978-3-642-17373-8_11
11. Maller, M., Bowe, S., Kohlweiss, M., Meiklejohn, S.: Sonic: Zero-knowledge snarks from linear-size universal and updatable structured reference strings. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) ACM CCS 2019. pp. 2111–2128. ACM (2019), https://eprint.iacr.org/2019/099.pdf

A Other Polynomial IOPs

A.1 Flip

This protocol establishes that the coefficient vector of one polynomial $f^{(rv)}(X)$, is the reverse of another polynomial $f(X)$. Formally, the relation is given by

$$\mathcal{R}_{\text{flip}} = \left\{ (\mathbb{i}, \mathbb{x}, \mathbb{w}) \;\middle|\; \begin{array}{l} \mathbb{i} = \mathsf{d} \\ \mathbb{x} = ([f(X)], [f^{(r)}(X)]) \\ \mathbb{w} = (f(X), f^{(rv)}(X)) \\ f(X) = \sum_{i=0}^{\mathsf{d}} f_i X^i \\ f^{(rv)}(X) = \sum_{i=0}^{\mathsf{d}} f_{\mathsf{d}-i} X^i \end{array} \right\} . \tag{42}$$

```
description: decides $\mathcal{L}(\mathcal{R}_{\text{flip}})$
inputs: $\mathbb{i} : \mathsf{d}$
        $\mathbb{x} : ([f(X)], [f^{(rv)}(X)])$
        $\mathbb{w} : (f(X), f^{(rv)}(X))$
begin
    V samples $z \xleftarrow{\$} \mathbb{F}\backslash\{0\}$
    V queries $[f(X)]$ and $[f^{(rv)}(X)]$ in $z$ and $z^{-1}$ respectively
    V receives $y = f(z)$ and $y^{(rv)} = f^{(rv)}(z^{-1})$
    V tests $y \overset{?}{=} z^{\mathsf{d}} \cdot y^{(rv)}$
```

Protocol 8: Flip

Theorem 7 (Security of Flip). *Protocol* Flip *of Protocol 8 is a Polynomial IOP for $\mathcal{L}(\mathcal{R}_{\text{flip}})$ with completeness and soundness with soundness error $\sigma = \mathsf{d}/(|\mathbb{F}| - 1)$.*

Proof. Completeness follows from construction, starting from the right-hand side of the test of the last line: $z^{\mathsf{d}} \cdot y^{(rv)} = z^{\mathsf{d}} \cdot \sum_{i=0}^{\mathsf{d}} f_{\mathsf{d}-i}(z^{-1})^i = \sum_{i=0}^{\mathsf{d}} f_{\mathsf{d}-i} z^{\mathsf{d}-i} = y$. For soundness, let the coefficients f_i be defined in terms of $f(X) = \sum_{i=0}^{\mathsf{d}} f_i X^i$. Observe that when $f^{(rv)}(X) \neq \sum_{i=0}^{\mathsf{d}} f_{\mathsf{d}-i} X^i$, then the functions $f(X)$ and $X^{\mathsf{d}} \cdot f^{(rv)}(X^{-1})$ are distinct. Furthermore, $f(X)$ is a polynomial of degree at most d and on $\mathbb{F}\backslash\{0\}$ so is $X^{\mathsf{d}} \cdot f^{(rv)}(X^{-1})$, so by Schwartz-Zippel the two functions can agree on at most d points. V accepts only when one of these d points is sampled for z; the probability of this event is $\sigma = \mathsf{d}/(|\mathbb{F}| - 1)$. □

A.2 Check Coefficient

The following protocol demonstrates that the kth monomial coefficient of a polynomial is equal to a given scalar. Formally, the relation is given by

$$\mathcal{R}_{\text{coeff}} = \left\{ (\mathbb{i}, \mathbb{x}, \mathbb{w}) \;\middle|\; \begin{array}{l} \mathbb{i} = (\mathsf{d}, k) \\ \mathbb{x} = ([f(X)], a) \\ \mathbb{w} = f(X) \\ f(X) = \sum_{i=0}^{\mathsf{d}} f_i X^i \text{ with } f_k = a \end{array} \right\} . \tag{43}$$

```
description: decides $\mathcal{L}(\mathcal{R}_{\text{coeff}})$
inputs: $\mathbb{i} : (\mathsf{d}, k)$
        $\mathbb{x} : [f(X)]$
        $\mathbb{w} : f(X)$
begin
    $\mathsf{P}$ computes $f_L \leftarrow \sum_{i=0}^{k-1} f_i X^i$ and $f_R \leftarrow \sum_{i=k+1}^{\mathsf{d}} f_i X^{i-k-1}$
    $\mathsf{P}$ sends to $\mathsf{V}$: $f_L$ and $f_R$ of degrees at most $k-1$ and $\mathsf{d}-k-1$, respectively
    $\mathsf{V}$ samples $z \xleftarrow{\$} \mathbb{F}\backslash\{0\}$ and queries $[f(X)]$, $[f_L(X)]$, and $[f_R(X)]$ in $z$
    $\mathsf{V}$ receives $y = f(z)$, $y_L = f_L(z)$, and $y_R = f_R(z)$
    $\mathsf{V}$ tests $y \overset{?}{=} y_L + a \cdot z^k + z^{k+1} \cdot y_R$
```

Protocol 9: CheckCoefficient

Theorem 8 (Security of CheckCoefficient). *Protocol* CheckCoefficient *of Protocol 9 is a Polynomial IOP for $\mathcal{L}(\mathcal{R}_{\text{coeff}})$ with completeness and soundness with soundness error $\sigma = \mathsf{d}/(|\mathbb{F}| - 1)$.*

Proof. Completeness follows from construction: $f(X) = \sum_{i=0}^{k-1} f_i X^i + aX^k + \sum_{i=k+1}^{\mathsf{d}} f_i X^i$ is an identity of polynomials and the check of the last line samples it in a point. For soundness, observe that when $f_k \neq a$ then no polynomials f_L and f_R of degrees at most $k-1$ and $\mathsf{d}-k-1$ can satisfy the polynomial identity. Sampling it in a random point will result in equality with probability at most $\sigma = \mathsf{d}/(|\mathbb{F}| - 1)$ due to Schwartz-Zippel. $\square$

A.3 Inner Product

The next protocol establishes that a given scalar is the inner product of the vectors of monomial coefficients of two polynomials. Formally, the relation is given by

$$\mathcal{R}_{\text{ip}} = \left\{ (\mathbb{i}, \mathbb{x}, \mathbb{w}) \;\middle|\; \begin{array}{l} \mathbb{i} = \mathsf{d} \\ \mathbb{x} = ([f(X)], [g(X)], a) \\ \mathbb{w} = (f(X), g(X)) \\ f(X) = \sum_{i=0}^{\mathsf{d}} f_i X^i \\ g(X) = \sum_{i=0}^{\mathsf{d}} g_i X^i \\ a = \sum_{i=0}^{\mathsf{d}} f_i g_i \end{array} \right\} . \tag{44}$$

```
description: decides $\mathcal{L}(\mathcal{R}_{\mathrm{ip}})$
inputs: $\mathbb{i} : \mathsf{d}$
            $\mathbb{x} : ([f(X)], [g(X)], a)$
            $\mathbb{w} : (f(X), g(X))$
begin
    P computes $f^{(rv)} \leftarrow \sum_{i=0}^{\mathsf{d}} f_{\mathsf{d}-i} X^i$
    P sends to V: $f^{(rv)}$ of degree at most $\mathsf{d}$
    P and V run Flip with $\mathbb{i} = \mathsf{d}$, $\mathbb{x} = ([f(X)], [f^{(rv)}(X)])$, and
      $\mathbb{w} = (f(X), f^{(rv)}(X))$
    P computes $h \leftarrow f(X)^{(rv)}(X) \cdot g(X)$
    P sends to V: $h(X)$ of degree at most $2\mathsf{d}$
    V samples $z \xleftarrow{\$} \mathbb{F}\backslash\{0\}$ and queries $[f^{(rv)}(X)]$, $[g(X)]$, and $[h(X)]$ in $z$
    V recieves $y_f = f^{(rv)}(z)$, $y_g = g(z)$ and $y_h = h(z)$
    V tests $y_h \overset{?}{=} y_f \cdot y_g$
    P and V run CheckCoefficient with $\mathbb{i} = (2\mathsf{d}, \mathsf{d})$, $\mathbb{x} = ([h(X)], a)$, and
      $\mathbb{w} = h(X)$
```

Protocol 10: InnerProduct

Theorem 9 (Security of InnerProduct). *Protocol* InnerProduct *of Protocol 10 is a Polynomial IOP for $\mathcal{L}(\mathcal{R}_{\mathrm{ip}})$ with completeness and soundness with soundness error $\sigma = 5\mathsf{d}/(|\mathbb{F}| - 1)$.*

Proof. Completeness follows from construction. Since the vector of coefficients of $f^{(rv)}(X)$ is that of $f(X)$ but reversed, the Flip subprotocol succeeds. The equation $h(X) = f^{(rv)}(X) \cdot g(X)$ is a polynomial identity so sampling it in a point results in an equality. Lastly, since the dth coefficient of $h(X)$ is $\sum_{i=0}^{\mathsf{d}} g_i f_i = a$, the CheckCoefficient subprotocol succeeds.

For soundness, suppose $a \neq \sum_{i=0}^{\mathsf{d}} g_i f_i$. Then either $f^{(rv)}(X)$ does not have the reverse coefficient vector from $f(X)$, $h(X) \neq f(X) \cdot g(X)$, or the dth coefficient of $h(X)$ is not a. The probabilities of passing the corresponding checks are at most $\sigma_{\mathsf{Flip}} = \mathsf{d}/(|\mathbb{F}| - 1)$, $\sigma_z = 2\mathsf{d}/(|\mathbb{F}| - 1)$, and $\sigma_{\mathsf{CheckCoefficient}} = 2\mathsf{d}/(|\mathbb{F}| - 1)$, respectively. Therefore, the probability that any one of these checks pass is at most $\sigma = 5\mathsf{d}/(|\mathbb{F}| - 1)$. □