Titan: Efficient Polynomial Commitments from IOPs over Groups

Chethan Kamath* Ravi Prakash† Samipa Samanta‡ Sruthi Sekar§ Nitin Singh¶

Abstract

In this paper, we propose Titan, an efficient polynomial commitment scheme (PCS) with transparent setup. It achieves commitment time of $O(n)$, evaluation time of $O(\sqrt{n})$ while the proof size and verification scales as $O(\sqrt[4]{n})$. Titan features an order of magnitude smaller proof sizes than hash based PCS, while featuring a significantly more efficient prover and verifier compared to state of the art group based schemes like Dory and Hyrax. To achieve this balance, Titan borrows two-tiered commitments from Dory, and realizes outer commitment using interactive protocols of proximity (IOPP) over groups, such as Basefold and WHIR, instead of expensive bilinear pairings. This allows Titan to be instantiated over general curves with discrete-log hardness such as Pasta Curves, instead of requiring pairing friendly curves.

We compile a variant of Spartan protocol for R1CS with Titan PCS to realize a SNARK. Our SNARK construction preserves the prover efficiency of the existing Spartan protocol, while improving proof size and verification quadratically from $O(\sqrt{n})$ to $O(\sqrt[4]{n})$. Concretely, for circuits of size $\geq 2^{22}$ this results in around $3\times$ more efficient proof size and verification.

Our blueprint of combining IOPPs over groups with Pedersen style inner commitments is of independent interest, as are several optimizations towards efficiently realizing WHIR IOPP over prime-order groups.

*IIT Bombay. Email: ckamath@cse.iitb.ac.in
†IIT Bombay. Email: 24m2131@iitb.ac.in
‡IIT Bombay. Email: 24d0380@iitb.ac.in
§IIT Bombay. Email: sruthi@cse.iitb.ac.in
¶IBM Research, India. Email: nitisin1@in.ibm.com

Contents

1 Introduction **3**
1.1 Our Contributions . . . 4

2 Technical Overview **5**
2.1 Titan PCS . . . 5
2.1.1 $O(\sqrt{n})$ Construction . . . 6
Committing, in two stages. . . . 7
Overview of multilinear IOPs of proximity. . . . 8
Group-based multilinear IOPPs. . . . 8
How to evaluate. . . . 9
Obtaining $O(\sqrt[4]{n})$ proof-size and verification time. . . . 10

3 Preliminaries **11**
3.1 Notation . . . 11
3.2 Interactive Oracle Proofs . . . 11
3.3 Interactive Oracle Proofs of Proximity . . . 11
3.4 Polynomial Commitments from IOPPs . . . 12
3.5 IOPP for Multilinear Polynomials . . . 13
3.5.1 Encoding Multilinear Polynomial Oracles . . . 14
3.6 Reed-Solomon Codes over Groups . . . 15
3.7 WHIR Iteration . . . 15
3.8 Sum-Check over Group Polynomials . . . 17

4 Titan: Overview **17**
4.1 Setup . . . 17
4.2 Commitment . . . 18
4.3 Evaluation Protocol for Titan PCS . . . 19

5 Titan PCS: Complete Protocol **20**
5.1 Outer Group Polynomial Evaluation . . . 20
5.2 Evaluation of partial polynomial . . . 21
5.3 Complete Protocol . . . 23
5.4 Extending to List Decoding Regime . . . 27
5.5 Summarizing Costs of Titan PCS . . . 28

6 Optimizations for IOPP over Groups **28**
6.1 Encoding Group Oracle . . . 28
6.2 Efficient Group Sumcheck . . . 29
6.2.1 Computing round messages in group sumcheck. . . . 30
6.3 Speeding up CSP prover . . . 30
6.3.1 Computing Round Messages in CSP. . . . 30

7 Implementation **31**
7.1 Evaluation Summary . . . 32
7.2 SNARK using Spartan and Titan . . . 32
7.3 SNARK using Spartan and Titan:Details . . . 35

A Titan PCS in the Random Oracle Model **37**
A.1 Complete Protocol . . . 37

1 Introduction

Polynomial commitment schemes (PCS) are pre-eminent primitives in the design of *Succinct Non-Interactive Arguments of Knowledge* (SNARKs). Most recent SNARKs are realized as ideal information theoretic protocol in the form of an *Polynomial Interactive Oracle Proof* (PIOP) which are compiled into an argument of knowledge using a polynomial commitment scheme. Several popular SNARKs such as Plonk [GWC19], Sonic [MBKM19], Marlin [CHM+20], Spartan [Set20], Hyperplonk [CBBZ23], Brakedown [GLS+23], Basefold [ZCF24] among others follow the aforementioned modular blueprint. To a great extent, the SNARKs inherit their security and efficiency from the PCS used to instantiate them. For example, whether or not a SNARK requires a trusted or a transparent setup depends on the underlying PCS. Similarly, whether or not a SNARK is plausibly post-quantum safe is also determined by the underlying PCS. Thus, significant body of work exists towards constructing polynomial commitment schemes meeting varied security and efficiency desiderata.

In terms of setup assumptions, arguably, the polynomial commitment schemes with overall practical efficiency require a structured trusted setup. These include the celebrated KZG [KZG10] scheme of Kate, Zaverucha and Goldberg for univariate polynomials and its analogue PST [PST13] for multilinear polynomials. These schemes are attractive as they feature $O(1)$ and $O(\log n)$ proof sizes and verification respectively. Several variants of KZG scheme such as Gemini [BCHO22], Zeromorph [KT23], Samaritan [GPS25] Mercury [EG25a] have also been subsequently developed for multilinear polynomials. Concrete proof sizes for large polynomials of size 1 million and beyond range from about 0.5 KB-10 KB. A major drawback of the above schemes is their dependence on trusted setup and at times slow cryptography resulting from pairing friendly elliptic curves over which they are instantiated. To mitigate these drawbacks, several approaches have been explored to design PCS from transparent setup, which we discuss below.

A promising line of works including Behemoth [SB25], Dew [AGL+23] and recently DewTwo [BMSS25] use groups of unknown order (e.g., class groups) to construct polynomial commitment schemes. While they achieve extremely small proof sizes, currently these schemes are largely of theoretical interest as prover overhead remains practically prohibitive and no experimental evaluation currently exists.

Among the more practical elliptic curve based polynomial commitment schemes are Hyrax [WTs+18] and Dory [Lee21]. Both express polynomial evaluation as a vector-matrix-vector product. Using homomorphic Pedersen commitments to rows of the matrix, the evaluation reduces to two inner product relations. Hyrax commits to the polynomial as row commitments of the associated matrix, which gives it a large $O(\sqrt{n})$ commitment size and verification complexity. Dory improves upon the approach in Hyrax by using bilinear pairings to construct a two-tiered commitment as a commitment over row commitments. It then uses a novel inner product argument with logarithmic verification complexity to realize PCS with $O(\log n)$ verifier, while featuring $O(n)$ commitment cost and $O(\sqrt{n})$ evaluation. Concretely, Dory achieves proof sizes in the range of 20KB-30KB for large polynomials. While featuring essentially optimal asymptotics, Dory employs bilinear pairings and target group operations which result in somewhat slower prover and verifier. It also fails to leverage non pairing-friendly elliptic curves with faster cryptographic operations.

Another popular approach to designing polynomial commitments involves compiling information theoretic protocols into an argument using hash functions. At a high level, these schemes encode a polynomial as a codeword in a linear code and prove evaluation by showing proximity of an associated vector to a linear code via the *interactive oracle proofs of proximity* (IOPP). The proximity claim is recursively reduced in several rounds, till the claim involves vectors small enough to be verified directly. The protocols use Merkle trees based on

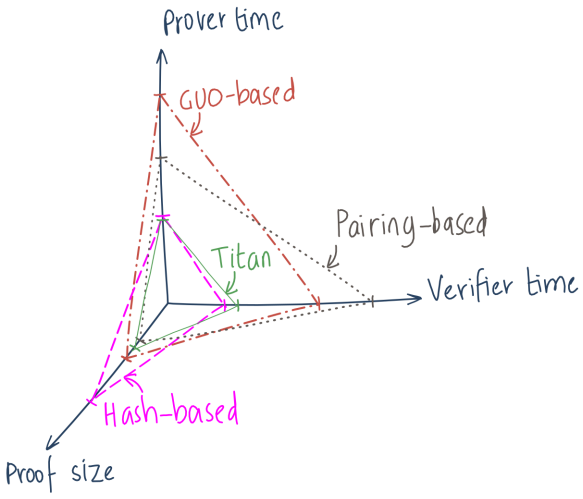


Figure 1: A schematic diagram comparing concrete efficiency of Titan (solid, green) with existing broad classes of PCS, taking into consideration prover-time (y axis), verifier-time (x axis) and proof-size (z axis). The classes compared to are GUO-based [SB25, AGL+23, BMSS25] (dash-dotted, red), pairing-based [WTs+18, Lee21] (dotted, grey) and hash-based [BBHR18a, BGKS20, GLS+23, ZCF24, ACFY24, ACFY25, GLH+24] (dashed, pink).

collision resistant hash functions (CRHF) to realize polynomial oracles. These constructions feature entirely symmetric key cryptography which makes them prover and verifier efficient in addition to being plausibly post-quantum safe. The prominent schemes using this hash based paradigm include FRI [BBHR18a], DEEP-FRI [BGKS20], Brakedown [GLS+23], Basefold [ZCF24], STIR [ACFY24] and recent works WHIR [ACFY25] and DeepFold [GLH+24]. On the downside, the proofs are often quite large ranging from few hundred kilobytes to few megabytes. The proof sizes are even larger when they are based on more conservative (provable) bounds, instead of aggressive (conjectured) bounds which have recently been disproved in [CS25].

A schematic diagram comparing concrete efficiency of the above-discussed classes of PCS can be found in Fig.1.

1.1 Our Contributions

Against this impressive body of work, we contribute a new transparent polynomial commitment scheme Titan, and a variant of Spartan [Set20] compiled with Titan PCS to obtain a SNARK. The key objective of Titan is to achieve order of magnitude smaller proof sizes than hash based schemes, while achieving significantly more efficient prover and verifier than group based schemes. Titan achieves this balance by avoiding expensive elliptic curve operations such as bilinear pairings and target group operations, while incorporating symmetric key based techniques from hash based constructions. These benefits also translate to the SNARK, which improves the transparent variant of Spartan both asymptotically and concretely. We summarise our contributions below:

- We construct transparent polynomial commitment scheme Titan with constant commitment size, $O(n)$ commitment time, $O(\sqrt{n})$ evaluation time, $O(\sqrt[4]{n})$ proof size and $O(\sqrt[4]{n})$ verification time. The scheme can be instantiated over both pairing friendly and non-pairing friendly curves with smooth subgroups of size $O(\sqrt{n})$. Our scheme can be used both with univariate and multilinear polynomials.

- When instantiated over Pasta Curves (which have efficient group operations), Titan achieves 2.5× faster commitment time, 10× faster evaluation time and 4× faster verification than Dory, while $1.5\times - 3.5\times$ in proof size for a practical range of polynomial sizes upto $2^{20} - 2^{30}$. Titan also compares favourably against Hyrax in most metrics. At the cost of giving up post-quantum security, Titan achieves an order of magnitude smaller proofs than hash based schemes. Titan also features extremely fast evaluation proofs, being $O(\sqrt{n})$ asymptotically without expensive pairings. Concretely, for polynomials of size 2^{20}, Titan achieves proof sizes of 25 KB, which can be verified in under 50 millisecond with a single threaded implementation. Additional evaluation appears in Section 7.

- We use Titan PCS to design a new variant of Spartan, which retains the prover efficiency of the original scheme in [Set20], while reducing proof size and verification complexity from $O(\sqrt{n})$ to $O(\sqrt[4]{n})$. Concretely, this leads to about 3× smaller proof size for polynomials of size $\approx 2^{23}$. Additional benchmarks are provided in Table 4.

Towards the goal of designing Titan with above efficiency profile, we make several technical contributions of independent interest.

- We adapt the two-tiered commitments by swapping expensive bilinear pairings for the outer commitment with IOPP over group polynomials. We then adapt techniques from multilinear IOPPs such as Basefold [ZCF24], WHIR [ACFY25] to work over prime-order groups instead of fields. The broad framework for sum-check based PIOPs over groups has been explored in prior work [BCS21], while more recent applications to optimize verifier in inner product arguments (IPA) appear in [EG25b] (and subsequently in [Zha25]). Nevertheless, our application towards realizing two-tiered commitments is novel to the best of our knowledge and contributes additional techniques.

- We propose several optimizations to efficiently enable multilinear IOPPs over groups. A key optimisation we introduce is adapting sumcheck over group polynomials arising in Basefold/WHIR IOPPs to exclusively use multi-scalar multiplications (MSMs) instead of the expensive group-exponentiations. This crucially makes our evaluation proofs extremely fast.

- We implement our scheme in Rust and provide a thorough experimental evaluation and comparison against relevant state of the art schemes.

2 Technical Overview

We present an overview of our techniques here. First, in Sec.2.1, we describe Titan, our multilinear PCS. In particular, we explain why we need to tailor and generalize multilinear IOPPs to groups. Multilinear IOPPs such as Basefold [ZCF24], WHIR [ACFY25] rely on sumcheck protocol and, therefore, key to "group-based IOPPs" is making sumcheck protocol work *efficiently* over groups – this we discuss in Section 6.2. Detailed constructions with security proofs appear in Sections 4 and 5. Finally, we describe how to construct a SNARK for R1CS by combining Titan with MicroSpartan [ZSC24] PIOP (a recent variant of Spartan [Set20]) for R1CS in Section 7.2.

2.1 Titan PCS

We describe Titan PCS in the setting of IOPs. In this setting, we view the function oracle $[\![f]\!]$ as *commitment* to function over a domain. In particular, the prover commits to a polynomial via the function defined by its evaluation over the domain. In the *evaluation* phase, verifier

receives the function oracle $[\![f]\!]$ as purported commitment to a polynomial, the evaluation point $\mathbf{z}$, and the claimed evaluation σ. The evaluation IOP accepts if and only if σ is the correct evaluation of polynomial defined by f at $\mathbf{z}$. Applying BCS [BCS16] transformation we obtain PCS in the *Random Oracle Model* (ROM), where function oracles are realized using vector commitments with succinct opening proofs, and verifier challenges are obtained using random oracle queries. We defer precise definitions to Section 4.

2.1.1 $O(\sqrt{n})$ Construction

We first describe a simplified version of Titan PCS that only achieves $O(\sqrt{n})$ proof-size and verifier time, but conveys the core ideas. At the end of the section we outline how the simplified scheme can be tweaked to reduce proof-size and verifier time to $O(\sqrt[4]{n})$, by reorganizing the oracle access. In the following discussion, $\mathbb{F}$ refers to the scalar field of prime order with a smooth domain $\mathsf{L} \subseteq \mathbb{F}$, and $\mathbb{G}$ refers to a group of same prime order in which the discrete-log problem is computationally hard.

The polynomial commitment scheme for multilinear polynomials in m-variables over $\mathbb{F}$ consists of:

- a *setup* algorithm, which samples a vector $\mathbf{g} = (g_1, \ldots, g_q)$ with $q = 2^{m/2} = \sqrt{n}$ and each g_i uniformly sampled from $\mathbb{G}$.
- a *commit* algorithm, which is used by a prover $\mathcal{P}$ to commit to a polynomial $\widetilde{f}(\mathbf{Z}) \in \mathbb{F}^{\leq 1}[Z_1, \ldots, Z_m]$ in at most m variables as an oracle $[\![f]\!]$; and
- an interactive *evaluation* protocol, which is used by $\mathcal{P}$ to convince a verifier $\mathcal{V}$ with oracle access $[\![f]\!]$ that $\widetilde{f}$ committed by it evaluates to σ at $\mathbf{z}$.

At a high level, $\mathcal{P}$ generates a commits to $\widetilde{f}$ as follows (see Fig.2):

1. Decompose $\widetilde{f}$ into q multilinear polynomials $\widetilde{f_1}(\mathbf{Y})$, ..., $\widetilde{f_q}(\mathbf{Y})$, each in $m/2$ variables $\mathbf{Y} := (Y_1, \cdots, Y_{m/2})$, where $\widetilde{f_i}(\mathbf{Y}) := \widetilde{f}(\langle i \rangle, \mathbf{Y})$ is $\widetilde{f}$ partially evaluated at $\langle i \rangle$, the bit decomposition of i.
2. Use commitment key $\mathbf{g}$ to generate Pedersen commitment $G_i \in \mathbb{G}$ to each row polynomial $\widetilde{f_i}(\mathbf{Y})$, i.e, $G_i = \sum_{j=1}^{q} \widetilde{f_i}(\langle j \rangle) \cdot g_j$.
3. Let $\tilde{G}(\mathbf{X}) \in \mathbb{G}[X_1, \cdots, X_{m/2}]$ be the $m/2$-variate group polynomial obtained by interpolating the group elements $G_1, \cdots, G_q$ over $\{0,1\}^{m/2}$. Output oracle $[\![G]\!]$ to commit to the polynomial $\widetilde{G}(\mathbf{X})$. We treat $[\![G]\!]$ as a commitment to the polynomial $\widetilde{f}$.
4. Additionally, let $\widetilde{g}(X_1, \ldots, X_{m/2})$ be the polynomial interpolating the generators. We assume that the verifier has oracle access $[\![g]\!]$ which commits to the polynomial $\widetilde{g}$ [1].

Next, to prove that $\widetilde{f}(\mathbf{z}) = \sigma$, where $\mathbf{z} \in \mathbb{F}^m$ is parsed as $\mathbf{x}, \mathbf{y} \in \mathbb{F}^{m/2}$, $\mathcal{P}$ and $\mathcal{V}$ proceed as follows, with $\mathcal{V}$ accepting in the end only if all the checks pass:

1. $\mathcal{P}$ computes the partially-evaluated polynomial $\widetilde{a}(\mathbf{Y}) := \widetilde{f}(\mathbf{x}, \mathbf{Y})$, computes its (Pedersen) commitment C and sends it to $\mathcal{V}$.
2. $\mathcal{P}$ and $\mathcal{V}$ execute the evaluation protocol of group-based multilinear IOPP to ensure that $\widetilde{G}(\mathbf{x}) = C$, where $\widetilde{G}$ is purported polynomial committed by oracle $[\![G]\!]$.

[1] Since the generators are public, we assume that the verifier computes the correct encoding of $\widetilde{g}$ one-time. Since the encoding is of size $O(\sqrt{n})$, we assume it is locally stored by the verifier

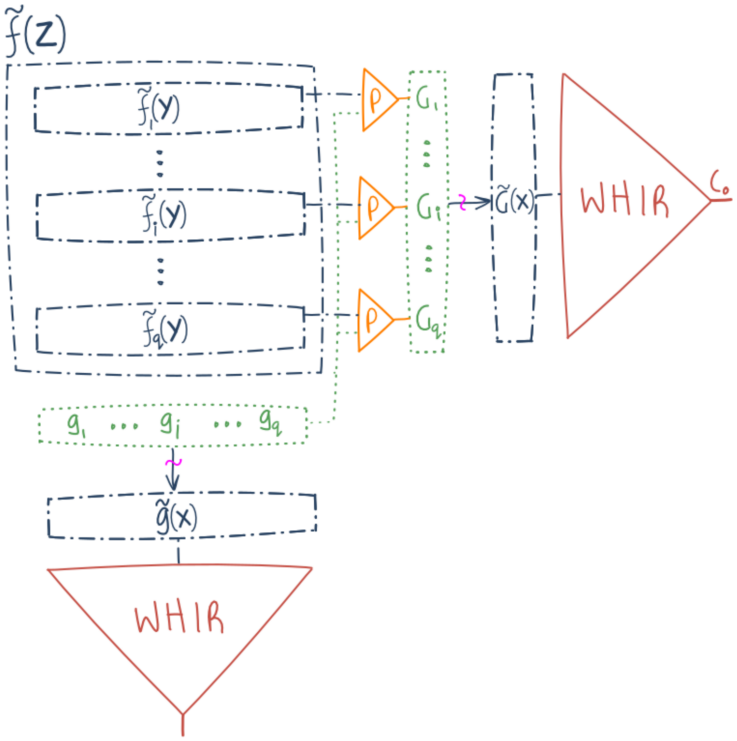


Figure 2: A schematic diagram showing how commitment works in Titan PCS. Red and orange (solid) triangles represent group-based WHIR PCS and Pedersen commitment, respectively. Green (dotted) boxes represent vectors. Blue (dash-dotted) boxes represent polynomials. Arrows with (pink) tilde represent polynomial interpolation.

3. $\mathcal{P}$ uses a proof of knowledge, concretely Compressed Sigma Protocol (CSP) [AC20], to convince $\mathcal{V}$ that it *knows* the partially-evaluated polynomial $\widetilde{a}(\mathbf{Y})$ for the commitment C *and* that $\widetilde{a}(\mathbf{y}) = \sigma$.

On a high level, binding follows as long as $[\![G]\!]$ binds $\mathcal{P}$ to unique $\widetilde{G}$ (later we use IOPs of Proximity to ensure this), while $\widetilde{G}$ in turn binds $\mathcal{P}$ to unique $\widetilde{f}$ due to binding property of the Pedersen commitment. Let us now elaborate a bit more on how the simplified scheme works.

Committing, in two stages. Recall that $q = 2^{m/2} = \sqrt{n}$. For $i \in [q]$, let $\widetilde{eq}_i(\mathbf{X}) := \widetilde{eq}(\mathbf{X}, \langle i \rangle)$ denote ith Lagrange basis polynomial for $\mathbb{F}^{\leq 1}[X_1, \ldots, X_{m/2}]$. As in prior works such as Hyrax [WTs+18] and Dory [Lee21], we view $\widetilde{f}(\mathbf{Z})$ as a $q \times q$ matrix (over $\mathbb{F}$) where the ith row consists of the q (canonical) evaluations of the $m/2$-variate multilinear polynomial $\widetilde{f}_i(\mathbf{Y}) = \widetilde{f}(\langle i \rangle, \mathbf{Y})$. More formally, this decomposition can be written as:

$$\widetilde{f}(\mathbf{X}, \mathbf{Y}) = \sum_{i=1}^{q} \widetilde{eq}(\mathbf{X}, \langle i \rangle) \cdot \widetilde{f}_i(\mathbf{Y}). \tag{1}$$

At this point, instead of using a "two-stage" commitment that relies on computationally-expensive bilinear maps as in Dory, we design and use a more efficient two-stage commitment based on Pedersen commitment and IOPP over groups. It is this switch that allows us to instantiate $\mathbb{G}$ even with plain groups, which *need not* support pairings, leading to efficiency gains. Next we describe our two-tiered commitment.

Recall that $\mathbf{g} = (g_1, g_2, \ldots, g_q) \in \mathbb{G}^q$ is the vector of uniformly-sampled generators, serving as the key to Pedersen commitment. $\mathcal{P}$ uses Pedersen commitment to commit to $\widetilde{f_i}$s, the partial evaluations of $\widetilde{f}$ over the hypercube $\{0,1\}^{m/2}$. That is, the commitment to the polynomial $\widetilde{f_i} \in \mathbb{F}^{\leq 1}[X_1, \cdots, X_{m/2}]$ is:

$$G_i = \mathsf{Com}(\widetilde{f_i}\,; \mathbf{g}) = \sum_{j=1}^{q} \widetilde{f_i}(\langle j \rangle) \cdot g_j. \tag{2}$$

At this point there are q group elements $G_1, \ldots, G_q$, which $\mathcal{P}$ interpolates into a multilinear polynomial $\widetilde{G}(\mathbf{X})$ with coefficients over $\mathbb{G}$, i.e., $\widetilde{G}(\mathbf{X})$ is the unique polynomial such that $\widetilde{G}(\langle i \rangle) = G_i$ for all $i \in [q]$. The oracle $[\![G]\!]$ for $\widetilde{G}$ serves as the eventual commitment to $\widetilde{f}$. To this end, we now review multilinear IOPPs and explain their generalization to multilinear polynomials over groups.

Overview of multilinear IOPs of proximity. We briefly recall the general blueprint for realizing a (univariate) PCS from an IOPP for Reed-Solomon Code $\mathsf{RS}[\mathbb{F}, \mathsf{L}, m]$ of degree $2^m - 1$ and evaluation domain $\mathsf{L} \subseteq \mathbb{F}$:

- To commit to a polynomial $\widehat{f}$ of degree $d := \rho|\mathsf{L}|$, where ρ is the rate of the code, $\mathcal{P}$ commits to its evaluations $\mathbf{f} \in \mathbb{F}^{|\mathsf{L}|}$ over L.
- To verify a claim $\widehat{f}(v) = \sigma$, $\mathcal{P}$ and $\mathcal{V}$ run an IOPP to check that the quotient function $q(x) = (\widehat{f}(x) - \sigma)/(x - v)$ is close to a polynomial of degree at most $(d - 1)$.

We invoke tools for multilinear IOPPs from Basefold [ZCF24] and their subsequent formalization in WHIR [ACFY25]. While we borrow notation, definitions and results from WHIR [ACFY25], we note that these results were first implied by Basefold [ZCF24] specialized to Reed-Solomon codes. Both works exploit the natural correspondence between m-variate multilinear polynomials and degree-2^m univariate polynomials: the univariate polynomial $\widehat{f}(X)$ corresponding to $\widetilde{f}(\mathbf{X})$ is defined as

$$\widehat{f}(X) := \widetilde{f}(X, X^2, \ldots, X^{2^{m-1}}). \tag{3}$$

Given this correspondence, WHIR enables commitments to multilinear polynomials by reducing their evaluation-claims to checking proximity to *constrained* Reed-Solomon codes $\mathsf{CRS}[\mathbb{F}, \mathsf{L}, m, \widetilde{w}, \sigma]$ which is the set of (evaluations of) univariate polynomials $\widehat{f} \in \mathbb{F}^{<2^m}[X]$ whose corresponding multilinear polynomial $\widetilde{f}$ satisfies

$$\sum_{\mathbf{b} \in \{0,1\}^m} \widetilde{w}(\widetilde{f}(\mathbf{b}), \mathbf{b}) = \sigma,$$

where $\widetilde{w}$ is an $(m+1)$-variate polynomial[2].

Group-based multilinear IOPPs. We now describe generalization of multilinear IOPPs to groups. To this end, first we define Reed Solomon encoding over groups $\mathsf{RS}[\mathbb{G}, \mathsf{L}, m]$ to be the set of functions $g : \mathsf{L} \to \mathbb{G}$ such that there exists a polynomial $\widehat{G} \in \mathbb{G}[X]$ of degree $< 2^m$ such that $g(x) = \widehat{g}(x)$ for all $x \in \mathsf{L}$. Then we show that both $\mathsf{RS}[\mathbb{F}, \mathsf{L}, m]$ and $\mathsf{RS}[\mathbb{G}, \mathsf{L}, m]$ have similar distance properties, which is necessary for folding to work. Next, to enable evaluation, we need to make sumcheck and folding work for $\mathsf{RS}[\mathbb{G}, \mathsf{L}, m]$. While the folding ports naturally to groups, making sumcheck protocol work efficiently over groups requires care. In [BCS21], Bootle et al. generalized the classical sumcheck protocol to polynomials

[2] In this paper, we only consider multilinear $\widetilde{w}$

defined over modules. In our setting we view $\mathbb{G}$ as a vector space over the field $\mathbb{F}$. Under this interpretation, the evaluation of a polynomial over groups is well-defined and we can define multiplication with field polynomials by taking scalar multiplication of the coefficients. Consequently, the algebraic structure required for sumcheck naturally extends to polynomials over $\mathbb{G}$. While this naïve version of group sumcheck works, it incurs expensive group exponentiations. In Section 6.2, we describe a more efficient sumcheck over groups that uses *multi-scalar-multiplications* (MSMs) instead of $O(n)$ group exponentiations [3]. With all these group machinery in place, Basefold/WHIR extend to group-based polynomials, and thus $\mathcal{P}$ (in Titan) does the following in Step 3 of the commit phase (Step 4 is similar):

1. Encode the group polynomial $\widetilde{G}$ as a Reed Solomon codeword using the correspondence from Eq.3, i.e., as evaluations of the associated degree-$\sqrt{n}$ univariate group polynomial $\widehat{G}(X) = \widetilde{G}(X, X^2, \ldots, X^{\sqrt{n}-1})$ over L. Since $\widetilde{G}$ is a $\sqrt{n}$-sized polynomial, the size of the (codeword) oracle is also $O(\sqrt{n})$ (group elements).

2. When compiling to a non-interactive argument in the Random Oracle Model, use a Merkle tree based on a collision resistant hash function $\mathcal{H} : \mathbb{G}^* \to \{0,1\}^{2\lambda}$ to succinctly commit to encoding of $\widetilde{G}$ thereby realizing the oracle $[\![G]\!]$.

We refer the reader to Sec.4.2 for further details.

How to evaluate. Finally, to evaluate a claim $\widetilde{f}(\mathbf{x}, \mathbf{y}) = \sigma$, we proceed in two stages. In the first stage, the prover proves $C \in \mathbb{G}$ to be the evaluation $\widetilde{G}(\mathbf{x})$ of the polynomial $\widetilde{G}$ encoded by the oracle $[\![G]\!]$. Note that in the honest case, C is also the Pedersen commitment of the partially evaluated polynomial $\widetilde{a}(\mathbf{Y}) = \widetilde{f}(\mathbf{x}, \mathbf{Y})$. Next, the prover shows knowledge of the field polynomial $\widetilde{a}(Y)$ which opens the commitment C and satisfies $\widetilde{a}(\mathbf{y}) = \sigma$. This latter proof of knowledge uses the *Compressed Sigma Protocol* (CSP) framework [AC20]. Thus, the evaluation protocol works as follows:

1. $\mathcal{P}$ computes the partially evaluated polynomial $\widetilde{a}(\mathbf{Y}) = \widetilde{f}(\mathbf{x}, \mathbf{Y})$, and a Pedersen commitment $C = \mathsf{Com}(\widetilde{a}; \mathbf{g})$ to it using the commitment key $\mathbf{g}$. $\mathcal{P}$ sends C to $\mathcal{V}$.

2. $\mathcal{V}$ checks, using multilinear IOPP over groups that $\widetilde{G}(\mathbf{x}) = C$ for $\widetilde{G}$ encoded by $[\![G]\!]$.

3. $\mathcal{P}$ shows knowledge of field polynomial with coefficients $\mathbf{a} \in \mathbb{F}^q$ via a *proof of knowledge* for the relation:

$$\left\{ (C, \sigma, \mathbf{g}; \mathbf{a}) : \sum_{i=1}^{\sqrt{n}} a_i \cdot g_i = C \ \wedge \ \sum_{i=1}^{\sqrt{n}} a_i \cdot \widetilde{eq}(\mathbf{y}, \langle i \rangle) = \sigma \right\}$$

Note that the CSP incurs a communication and verification complexity $O(\sqrt{n})$. Also note that the above IOPP only ensures that $[\![G]\!]$ encodes a word which is close to a Reed Solomon codeword – for a PCS we need to bind G to a unique polynomial. This can, however, be resolved using standard approaches of ensuring proximity of the function to *unique* polynomial or using *out-of-domain sampling* to leverage larger proximity parameters. We refer the reader to Sec.5 for the formal details.

[3] Specifically, it uses $O(\sqrt{n})$ MSMs each of size $O(\sqrt{n})$

Obtaining $O(\sqrt[4]{n})$ proof-size and verification time. The simplified Titan PCS we just described yields a realization of a transparent multilinear PCS to a polynomial $\widetilde{f}$ of size n with $O(\sqrt{n})$ sized oracle. To bring down the proof-size and verifier time to $O(\sqrt[4]{n})$, we rely on a couple of common optimization tricks:

1. First, we exploit the fact that IOPPs like Basefold/WHIR query the oracle over sets structured as cosets of a subgroup. Thus we commit to evaluated polynomials *coset-wise*. In more detail, instead of Merkle committing to a polynomial $\widetilde{f}$ by setting the leaves to be *individual* evaluations, we first partition the evaluation domain L into appropriate-sized cosets and then Merkle commit with leaves corresponding to evaluations *over these cosets*. For best performance, we set the coset size to be roughly $\sqrt[4]{n}$, which results in a Merkle tree of depth $\log(\sqrt[4]{n})$ with $O(\sqrt[4]{n})$ leaves.

2. Naively, a CSP verifier over instance of size $\sqrt{n}$ incurs $O(\sqrt{n})$ costs. Prior works such as [GMN22, BMM+21, GPPS24] have obtained more efficient CSP verifiers in *trusted setup*, by having structured commitment keys. More recently, in [EG25b], sublinear verifier effort for *inner product arguments* (IPA) in transparent setting has been achieved based on the following observation: generators in each round of an IPA can be viewed as a polynomial in evaluation basis, obtained by folding the polynomial corresponding to original generators by IPA verifier's challenges. This allows delegating the folding to the prover, and have it send the folded generators once the instance size decreases to $o(n)$. The verifier can then check the correctness of reduced instance, by querying the original polynomial. In Titan also, we encode Pedersen generators as multilinear oracle and terminate the CSP when the folded instance size becomes $\sqrt[4]{n}$, giving us a protocol with $O(\sqrt[4]{n})$ communication and verification complexity.

Optimizing IOPPs over Groups. We discuss several optimizations to efficiently realize IOPPs over groups. In Section 6, we describe optimizations targeting efficiency of encoding group oracles, executing sumcheck protocol over groups arising in multilinear IOPPs and efficiently computing round messages in a CSP.

Spartan +Titan SNARK In Section 7.2, we combine Titan PCS with Spartan [Set20] PIOP for R1CS satisfiability to obtain a SNARK for R1CS. We improve upon the transparent version of Spartan (Spartan$_{\text{DL}}$ in [Set20]), reducing proof sizes and verification time (both $\sqrt[4]{n}$ instead of $\sqrt{n}$) while featuring comparably performant prover[4]. The concrete comparison is presented in Section 7. Our current implementation is not ZK, but in the SNARK, we can obtain a hiding commitment to the witness by using the hiding Pedersen commitments in the inner layer. IOPP can run over the hiding matrix commitment. The ZK-variant of CSP incurs small evaluation overhead (additional $\sqrt{n}$ sized blinding), with other parameters virtually the same

Comparison with Recent Group Based IOPPs. We end the overview with a comparison of our techniques with very recent related works [EG25b]. These works also use group based IOPPs to obtain sublinear verification for IPAs. However, in both the aforementioned works, the group-based IOPP works over public polynomials of size $O(n)$ – with significant cost to the prover. In contrast, our application of group-based IOPP is more pervasive; we use it in a two-tiered setting leading to group polynomials of size $O(\sqrt{n})$, where one of the group oracles is adversarially generated. We additionally optimise the group sumcheck to use MSMs instead of group-exponentiations. While [EG25b] asymptotically achieve polylog(n) proof size and verification, we employ only one round of folding to *square-root* size in the IOPP to terminate at the instance size of $O(\sqrt[4]{n})$. Concretely, this results in better proof sizes as our

[4]with the caveat that the implementation of Spartan is zero-knowledge, unlike ours

initial oracles are small ($\propto \sqrt{n}$) and the overhead of multiple rounds of Merkle tree openings to achieve polylog(n) efficiency exceeds sending the $O(\sqrt[4]{n})$ sized instance in full. Smaller Merkle trees also increase the effectiveness of path compression, with nodes near the top shared amongst many paths – a property leveraged by our implementation.

3 Preliminaries

3.1 Notation

For $n \in \mathbb{N}$, we denote the sequence of integers $(1, \ldots, n)$ by $[n]$. We denote the binary representation of an integer $i \in \mathbb{N}$ by $\langle i \rangle$. We denote λ to be a security parameter. We use negl to denote a negligible function: for any integer $c > 0$, there exists a $n \in \mathbb{N}$ such that for all $x > n$, $\mathsf{negl}(x) \leq 1/x^c$. Throughout, $\mathbb{F}$ denotes a field of prime order and $\mathbb{G}$ denotes a group of same prime order. We use $\mathsf{L} \subseteq \mathbb{F}$ to denote a *smooth* subset of $\mathbb{F}$ i.e it is a multiplicative coset of $\mathbb{F}^*$ whose order is a power of 2. For $k \in \mathbb{N}$, we define $\mathsf{L}^{(k)} := \{x^k : x \in \mathsf{L}\}$. We use $\mathbb{F}^{\leq d}[X_1, \ldots, X_m]$ to denote the set of all m-variate multivariate polynomials over $\mathbb{F}$ of individual degree at most d. In particular, $\mathbb{F}^{\leq d}[X]$ denotes the set of univariate polynomials of degree at most d, and $\mathbb{F}^{\leq 1}[X_1, \ldots, X_m]$ denotes the set of m-variate multilinear polynomials (of size ($n := 2^m$). We reserve the "tilde" symbol over a function (e.g $\widetilde{f}$) to indicate that it is a *multilinear* polynomial. Similarly, a "hat" symbol over a function (e.g $\widehat{f}$) denotes the associated univariate polynomial of the multilinear polynomial $\widetilde{f}$. The oracle symbol $[\![\widetilde{f}]\!]$ denotes the oracle access to the Reed Solomon encoding of $\widetilde{f}$ over a smooth domain. We define the equality polynomial $\widetilde{eq}$ to be the multilinear extension of the equality function $eq : \mathbb{F}^{2m} \to \mathbb{F}$

$$\widetilde{eq}(\mathbf{X}, \mathbf{Y}) = \prod_{i=1}^{m} (X_i \cdot Y_i + (1 - X_i) \cdot (1 - Y_i)).$$

We sometimes use 3 arguments to denote partitioning of a vector into sub-vectors. We will use $\widetilde{eq}(x, y; z)$ to denote that the left hand vector is a concatenation of x and y.

For two functions $f, g : \mathsf{L} \to \mathbb{F}$, $\Delta(f, g)$ denotes the relative Hamming distance between f and g. For a set $\mathcal{S} \subseteq \mathbb{F}^{|\mathsf{L}|}, \Delta(f, \mathcal{S}) := \min_{h \in \mathcal{S}} \Delta(f, h)$.

3.2 Interactive Oracle Proofs

Definition 3.1. *Interactive Oracle Proofs(IOPs)[BCS16, RRR16] are information theoretic objects that generalizes the notion of Probabilistically Checkable Proofs(PCPs)[BFLS91, FGL+96, AS92, ALM+92]. An IOP is an interactive proof where the verifier $\mathcal{V}$ is given query access to prover $\mathcal{P}$'s message at each round. An IOP is an interactive proof for a relation $\mathcal{R} = (x, w)$ where $\mathcal{R}$ is an oracle relation such the statement x consists of oracles to polynomials over a field $\mathbb{F}$. We denote $\langle \mathcal{P}, \mathcal{V} \rangle(x)$ the output of $\mathcal{V}$ after interacting with $\mathcal{P}$ on input x; which is accept or reject.*

3.3 Interactive Oracle Proofs of Proximity

An IOP of proximity is for a family of codes $\mathcal{C}$ where both prover $\mathcal{P}$ and the verifier $\mathcal{V}$ receive as common input a specification of a code $C \in \mathcal{C}$ which is a set of functions $f : \mathsf{L} \to \mathbb{F}$ for $\mathsf{L} \subseteq \mathbb{F}$. We also assume that $\mathcal{V}$ has oracle access to a function $f^{(0)} : \mathsf{L} \to \mathbb{F}$ and that $\mathcal{P}$ also receives the same function as the explicit input.

Definition 3.2. *An r-round IOPP $S = (\mathcal{P}, \mathcal{V})$ is a $(r+1)$-round IOP for the error correcting code $C = \{f : \mathsf{L} \to \mathbb{F}\}$ with respect to a distance measure Δ and proximity parameter δ if*

- *The first prover message denoted by* $[\![f^{(0)}]\!]$*, is a purported codeword of* C
- $\Pr[\langle \mathcal{P}, \mathcal{V} \rangle([\![f]\!]^{(0)}) = \text{accept} | \Delta(f^{(0)}, C) = 0] = 1$ *and*
- *For any malicious prover* $\mathcal{P}^*$*,* $\Pr[\langle \mathcal{P}, \mathcal{V} \rangle = \text{accept} | \Delta(f^{(0)}, C) > \delta] \leq \mathsf{negl}$

The sum of lengths of all prover message except for the first message is the IOPP proof length, total number of queries is the query complexity.

Our construction leverages interactive oracle proofs of proximity (IOPPs) as defined above.

3.4 Polynomial Commitments from IOPPs

Several efficient constructions of univariate polynomial commitment schemes [BGKS20, KPV22, ZCF24, ACFY24, ACFY25] and associated SNARKs [AHIV17, BBHR18a, BBHR18b, BCR+19] in the transparent setting have leveraged IOPPs which allow a verifier to succinctly verify that a committed function is close to some linear code. Recent works [ZCF24, ACFY25] have proposed IOPPs for multilinear polynomials. In this work, we adapt techniques from multilinear IOPPs to groups in designing the polynomial commitment scheme Titan. Similar to [ZCF24], we describe our polynomial commitment scheme (PCS) in the setting of interactive oracle proofs , where oracle to the function is treated as the commitment. Additionally, we consider *extractability* under random point evaluation, which allows extracting the polynomial from an adversary who succeeds in proving evaluation on uniformly sampled points. This restricted notion of extractability suffices for several applications of the PCS, and can be easily extended to standard extractability in the case of multilinear polynomials, using an additional sum-check.

Definition 3.3. *A polynomial commitment scheme* PC *for multilinear polynomials over field* $\mathbb{F}$ *consists of a tuple of oracle algorithms* (**Setup**, **Commit**, **Open**, **Eval**) *such that*

- $\mathbf{Setup}(1^\lambda, m) \to \mathsf{pp}$ *takes the security parameter* λ *and the number of variables* $m \in \mathbb{N}$ *and outputs public parameters* pp*.*
- $\mathbf{Commit}(\mathsf{pp}, \widetilde{f}) \to [\![f]\!]$ *takes public parameters* pp *and the multilinear polynomial* $\widetilde{f} \in \mathbb{F}^{\leq 1}[X_1, \ldots, X_m]$ *and outputs a function oracle* $[\![f]\!]$ *(analogue of commitment in standard model).*
- $\mathbf{Open}(\mathsf{pp}, f, \widetilde{f}) \to b$ *is a relation that takes public parameters* pp*, a function (word)* f*, a multilinear polynomial* $\widetilde{f}$*, and outputs a bit* b *indicating accept or reject.*
- $\mathbf{Eval}(\mathsf{pp}, [\![f]\!], m, \mathbf{z}, \sigma; \widetilde{f}) \to b$ *is an interactive oracle protocol* $\langle \mathcal{P}_{\mathbf{Eval}}(\widetilde{f}), \mathcal{V}_{\mathbf{Eval}} \rangle(\mathsf{pp}, [\![f]\!], m, \mathbf{z}, \sigma)$ *between a PPT prover and a PPT verifier with common inputs consisting of* $[\![f]\!]$*, an evaluation point* $\mathbf{z} \in \mathbb{F}^m$ *and a value* $\sigma \in \mathbb{F}$*.* $\mathcal{P}$ *additionally knows a multilinear polynomial* $\widetilde{f}$*.* $\mathcal{P}$ *wants to convince* $\mathcal{V}$ *that* $\widetilde{f}$ *is an opening to* $[\![f]\!]$ *and* $\widetilde{f}(\mathbf{z}) = \sigma$*.* $\mathcal{V}$ *outputs a bit* b *at the end of the protocol indicating accept or reject.*

PC *must satisfy completeness, binding and knowledge soundness.*

- PC *is* ***complete*** *if for any* $\lambda, m \in \mathbb{N}$*,* $\widetilde{f} \in \mathbb{F}^{\leq 1}[X_1, \cdots, X_m]$ *and* $\mathbf{z} \in \mathbb{F}^m$

$$\Pr\left[\mathbf{Eval}(\mathsf{pp}, [\![f]\!], m, \mathbf{z}, \sigma; \widetilde{f}) = 1 \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathbf{Setup}(1^\lambda, m); \\ [\![f]\!] \leftarrow \mathbf{Commit}(\mathsf{pp}; \widetilde{f}); \\ \sigma \leftarrow \widetilde{f}(\mathbf{z}) \end{array}\right] = 1$$

- PC *is* ***binding*** *if for any* $m \in \mathbb{N}$ *and PPT adversary* $\mathcal{A}$*,*

$$\Pr\left[b_0 = b_1 = 1 \wedge \widetilde{f_0} \neq \widetilde{f_1} \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathbf{Setup}(1^\lambda, m); \\ (\llbracket f \rrbracket, \widetilde{f_0}, \widetilde{f_1}) \leftarrow \mathcal{A}(\mathsf{pp}); \\ b_0 \leftarrow \mathbf{Open}(\mathsf{pp}; \llbracket f \rrbracket, \widetilde{f_0}); \\ b_1 \leftarrow \mathbf{Open}(\mathsf{pp}; \llbracket f \rrbracket, \widetilde{f_1}) \end{array} \right] \leq \mathsf{negl}(\lambda)$$

- PC *is* ***knowledge sound with random point evaluation*** *if for any PPT adversary* $\mathcal{A}$*, there exists a PPT algorithm* Ext *such that the following probability is negligible in* λ*:*

$$\Pr\left[\begin{array}{l} b = 1 \wedge \\ \mathcal{R}_{Ev}(\mathsf{pp}, \llbracket f \rrbracket, \mathbf{z}, \sigma; \widetilde{f}) = 0 \end{array} \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathbf{Setup}(1^\lambda, m), \\ \llbracket f \rrbracket \leftarrow \mathcal{A}(\mathsf{pp}), \\ \mathbf{z} \leftarrow \mathbb{F}^m,\ \sigma \leftarrow \mathcal{A}(\mathsf{pp}, \mathbf{z}), \\ \widetilde{f} \leftarrow \mathsf{Ext}^{\mathcal{A}}(\mathsf{pp}, \llbracket f \rrbracket, \mathbf{z}, \sigma) \\ b \leftarrow \langle \mathcal{A}, \mathcal{V}_{Ev} \rangle (\mathsf{pp}, \llbracket f \rrbracket, \mathbf{z}, \sigma) = 1 \end{array} \right]$$

where the oracle relation $\mathcal{R}_{Ev}$ *is defined as*

$$\mathcal{R}_{Ev} = \{((\mathsf{pp}, \llbracket f \rrbracket, \mathbf{z}, \sigma); \widetilde{f}) : \mathbf{Open}(\mathsf{pp}, f; \widetilde{f}) = 1 \wedge \widetilde{f}(\mathbf{z}) = \sigma\}.$$

3.5 IOPP for Multilinear Polynomials

We briefly summarize tools we use from multilinear IOPPs such as [ZCF24, ACFY25]. Our definitions and notation borrow heavily from WHIR [ACFY25]. In particular, we consider IOPPs for *Constrained Reed Solomon Codes* for polynomial evaluation claims. Let $m \in \mathbb{N}$ be a parameter and let $n = 2^m$. We consider multilinear polynomials $\widetilde{f} \in \mathbb{F}^{\leq 1}[X_1, \ldots, X_m]$, and evaluation claims of the type $\widetilde{f}(\mathbf{z}) = \sigma$ for some $\mathbf{z} \in \mathbb{F}^m$ and $\sigma \in \mathbb{F}$. In WHIR, the multilinear polynomial $\widetilde{f}$ is encoded as a Reed-Solomon codeword over a suitably defined domain.

Definition 3.4 (Reed Solomon Code)**.** *For finite field* $\mathbb{F}$*, a domain* $\mathsf{L} \subseteq \mathbb{F}$ *and* $m \in \mathbb{N}$*, we define* Reed Solomon Code $C := \mathsf{RS}[\mathbb{F}, \mathsf{L}, m]$ *as the set of functions* $f : \mathsf{L} \to \mathbb{F}$ *for which there exists a polynomial* $p \in \mathbb{F}[X]$ *of degree* $< 2^m$ *such that* $f(x) = p(x)$ *for all* $x \in \mathsf{L}$*.*

We assume that $\rho := 2^m/|\mathsf{L}|$ where $\rho \in (0, 1)$, as the rate of C. To map multilinear polynomial $\widetilde{f}$ to a Reed-Solomon codeword, we define the associated univariate polynomial

$$\widehat{f}(X) := \widetilde{f}(X, X^2, \cdots, X^{2^{m-1}}) \tag{4}$$

which has degree $< 2^m$. Then, we say that the codeword $\left(\widehat{f}(x)\right)_{x \in \mathsf{L}}$ encodes the multilinear polynomial $\widetilde{f}$ as a codeword in $\mathsf{RS}[\mathbb{F}, \mathsf{L}, m]$. Going ahead, we continue to use the notation $\widehat{f}$ to denote the univariate polynomial of degree $< 2^m$ associated with m-variate multilinear polynomial $\widetilde{f}$ given by Equation (4).

To denote the multilinear polynomials satisfying certain evaluation claims, WHIR introduces *constrained* Reed-Solomon codes. Specifically, given an $(m+1)$-variate multilinear polynomial $\widetilde{w}(Z, \mathbf{X})$ with $\mathbf{X}$ denoting variables $X_1, \ldots, X_m$ and $\sigma \in \mathbb{F}$, we define constrained Reed-Solomon code as the subcode of $\mathsf{RS}[\mathbb{F}, \mathsf{L}, m]$ as:

Definition 3.5 (Constrained RS code [ACFY25])**.** *For* $\mathbb{F}, \mathsf{L}, m, \widetilde{w}, \sigma$ *as defined above. A* constrained Reed-Solomon *code* $\mathsf{CRS}[\mathbb{F}, \mathsf{L}, m, \widetilde{w}, \sigma]$ *is defined as the set of codewords in* $\mathsf{RS}[\mathbb{F}, \mathsf{L}, m]$*, consisting of evaluations of polynomials* $\widehat{f} \in \mathbb{F}^{<2^m}[X]$ *over* L *such that their equivalent multilinear polynomial* $\widetilde{f} \in \mathbb{F}[X_1, \ldots, X_m]$ *satisfies* $\sum_{\mathbf{b} \in \{0,1\}^m} \widetilde{w}(\widetilde{f}($
$\mathbf{b}), \mathbf{b}) = \sigma$*.*

In particular, taking $\widetilde{w}(Z, \mathbf{X}) = Z \cdot \widetilde{eq}(\mathbf{X}, \boldsymbol{\alpha})$ in the above definition, the subcode consists of codewords corresponding to polynomials $\widetilde{f}$ such that $\widetilde{f}(\boldsymbol{\alpha}) = \sigma$. As a key ingredient, WHIR [ACFY25], presents IOPP for constrained Reed-Solomon codes. The IOPPs in [ZCF24, ACFY25] proceed in several iterations; each iteration involves k "sum-check" rounds to reduce the instance of proximity testing with respect to code of dimension 2^m to checking proximity with respect to a code of dimension 2^{m-k}. We use simplified variants of above IOPPs, where we restrict to single iteration with $k \approx m/2$. Using two-tiered commitments ensures that the group oracles are small $\approx \sqrt{n}$ – with one iteration offering the best trade-off for practical values of n. Therefore, a single iteration of WHIR, adapted to groups is a key tool we use in our construction. We give a self-contained description of an iteration in WHIR in Section 3.7.

We next discuss folding of functions, a key notion used in IOPPs [BBHR18a, ZCF24, ACFY25]. Throughout the paper, we consider functions over *smooth domains*, i.e, subgroups of roots of unity of order 2^r for some $r \in \mathbb{N}$.

Definition 3.6 (Folded Functions)**.** *Let $\mathsf{L} \subseteq \mathbb{F}$ be a smooth domain and $f : \mathsf{L} \to \mathbb{F}$ be a function. Let $\mathsf{L}^{(2)}$ denote the set $\{x^2 : x \in \mathsf{L}\}$, i.e, the set of "squares" in L. For $\alpha \in \mathbb{F}$, we define the function $f_\alpha := \mathsf{fold}(f, \alpha)$ as the function $f_\alpha : \mathsf{L}^{(2)} \to \mathbb{F}$, with $f_\alpha(x^2)$ for $x \in \mathsf{L}$ defined by:*

$$f_\alpha(x^2) = \frac{f(x) + f(-x)}{2} + \alpha \cdot \frac{f(x) - f(-x)}{2x} \tag{5}$$

Recall that we use $\mathsf{L}^{(t)}$ to denote the set $\{x^t : x \in \mathsf{L}\}$.

The above definition also implies that given oracle access to f over L, oracle access to f_α over $\mathsf{L}^{(2)}$ can be simulated via two oracle queries to f at the square-roots of the queried point. We can extend Definition 3.6 to describe *multi-folding* of a function using $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_k)$ as:

$$\mathsf{fold}(f, (\alpha_1, \ldots, \alpha_k)) = \mathsf{fold}(\mathsf{fold}(f, \alpha_1, \ldots, \alpha_{k-1}), \alpha_k)$$

We note that in this case, the function $\mathsf{fold}(f, \boldsymbol{\alpha})$ is defined over the domain $\mathsf{L}^{(2^k)}$. It is also easy to verify that oracle access to $\mathsf{fold}(f, \boldsymbol{\alpha})$ can be simulated using 2^k oracle queries to f at the 2^k pre-images of the queried point under the map $x \mapsto x^{2^k}$. We use this property to realize efficient oracle access to f. The folding of "polynomial" functions is nicely captured by the following Lemma.

Lemma 3.1 ([ACFY25])**.** *For a function $f : \mathsf{L} \to \mathbb{F}$ given by $f(x) = \widetilde{f}(\mathbf{x})$ for $\widetilde{f} \in \mathbb{F}[X_1, \ldots, X_m]$ and $\boldsymbol{\alpha} \in \mathbb{F}^k$ for some $k \leq m$,we have $\mathsf{fold}(f, \boldsymbol{\alpha})(y) \equiv \widetilde{f}(\boldsymbol{\alpha}, \mathbf{y})$ for $y \in \mathsf{L}^{(2^k)}$. Here $\mathbf{x}$ and $\mathbf{y}$ denote the vectors $(x, x^2, \ldots, x^{2^{m-1}})$ and $(y, y^2, \ldots, y^{2^{m-k-1}})$ respectively.*

3.5.1 Encoding Multilinear Polynomial Oracles

Multilinear IOPPs (e.g. [ACFY25]) require queries to the folded function $\mathsf{fold}(f, \boldsymbol{\alpha})$. We earlier noted that a query to $\mathsf{fold}(f, \boldsymbol{\alpha})$ can be simulated using 2^k queries to f. We now provide a more explicit description of the same. For the honest prover, we have $f = \widetilde{f}$ for some m-variate multilinear polynomial. We note that for $y = x^{2^k} \in \mathsf{L}^{(2^k)}$, we have:

$$\begin{aligned} \mathsf{fold}(f, \boldsymbol{\alpha})(y) = \widehat{f_{\boldsymbol{\alpha}}}(y) &= \widetilde{f}(\boldsymbol{\alpha}, y, y^2, \ldots, y^{2^{m-k-1}}) \quad \text{(Lemma 3.1)} \\ &= \sum_{\mathbf{b} \in \{0,1\}^k} \widetilde{eq}(\boldsymbol{\alpha}, \mathbf{b}) \cdot \widetilde{f}(\mathbf{b}, y, y^2, \ldots, y^{2^{m-k-1}}) \end{aligned} \tag{6}$$

Thus, we commit to f as follows: For each $y \in \mathsf{L}^{(2^k)}$, the prover commits to the vector

$$\left(f(\mathbf{b}, y, y^2, \ldots, y^{2^{m-k-1}})\right)_{\mathbf{b} \in \{0,1\}^k}$$

as one leaf of the Merkle tree. Thus, each leaf of the Merkle tree corresponds evaluations of multilinear polynomial $\widetilde{p}_y(\mathbf{b}) = \widetilde{f}(\mathbf{b}, y, \ldots, y^{2^{m-k-1}})$ over $\mathbf{b} \in \{0,1\}^k$. It can be shown that $\widetilde{p}_y$ is completely determined by y and evaluations of $\widehat{f}$ on the coset $\{x : x^{2^k} = y\}$, and thus the above commitment can be seen as committing to $f$over L in a coset-wise manner. To open $\mathsf{fold}(f, \boldsymbol{\alpha})(y)$, the prover opens the above vector and sends it to the verifier, who checks consistency with the Merkle root and computes $\mathsf{fold}(f, \boldsymbol{\alpha})(y)$ using Equation (6). Thus, the verifier effort to verify q openings is $O(q(m + 2^k))$, and the prover needs to provide only q authentication paths instead of $q \cdot 2^k$ paths.

3.6 Reed-Solomon Codes over Groups

We now define Reed-Solomon codes over group $\mathbb{G}$ of prime order. Let $\mathsf{L} \subseteq \mathbb{F}$ be a domain and $m \in \mathbb{N}$. We define $\mathsf{RS}[\mathbb{G}, \mathsf{L}, m]$ as the set of functions $g : \mathsf{L} \to \mathbb{G}$ such that there exists a polynomial $\widehat{g}(X) \in \mathbb{G}[X]$ of degree $< 2^m$ such that $g(x) = \widehat{g}(x)$ for all $x \in \mathsf{L}$. We note that the codes $\mathsf{RS}[\mathbb{F}, \mathsf{L}, m]$ and $\mathsf{RS}[\mathbb{G}, \mathsf{L}, m]$ have identical generator matrices, and hence the same distance properties. This follows from the (vector space) isomorphism $x \mapsto x \cdot g$ between $\mathbb{F}$ and $\mathbb{G}$ for any generator g of $\mathbb{G}$. In subsequent sections, use WHIR IOPP for Reed-Solomon codes over $\mathbb{G}$.

We now include a self-contained description of tools and definitions from prior work to help the reader understand our construction.

3.7 WHIR Iteration

We now describe a single iteration of WHIR which reduces testing proximity of a function to $\mathsf{CRS}[\mathbb{F}, \mathsf{L}, m, \widetilde{w}, \sigma]$ to testing proximity of a function to $\mathsf{CRS}[\mathbb{F}, \mathsf{L}^{(2^k)}, m - k, \widetilde{w}', \sigma']$ for some $k \in \mathbb{N}$. We assume $\widetilde{w}(Z, \mathbf{X}) = Z \cdot \widetilde{eq}(\mathbf{X}, \boldsymbol{\alpha})$ for some $\boldsymbol{\alpha} \in \mathbb{F}^m$, which is the case of interest to us. Let ρ denote the rate of the Reed-Solomon code, i.e, $\rho = 2^m/|\mathsf{L}|$. Let $f : \mathsf{L} \to \mathbb{F}$ be a function and let $\delta \in (0, 1 - \sqrt{\rho})$ be a proximity parameter. The claim $f \in \mathsf{CRS}[\mathbb{F}, \mathsf{L}, m, \widetilde{w}, \sigma]$ is essentially a sum-check instance $\sum_{\mathbf{b} \in \{0,1\}^m} \widetilde{w}(\widetilde{f}(\mathbf{b}), \mathbf{b}) = \sigma$. Therefore, initially $\mathcal{P}$ and $\mathcal{V}$ run k rounds of sum-check protocol as below:

SumCheck Phase: The prover $\mathcal{P}$ and verifier $\mathcal{V}$ interact as follows:

- In round i for $i \in [k]$.
  - $\mathcal{P}$ computes:
$$h_i(X) = \sum_{\mathbf{b} \in \{0,1\}^{m-i}} \widetilde{w}(f(\boldsymbol{\alpha}_i, X, \mathbf{b}), \boldsymbol{\alpha}_i, X, \mathbf{b})$$
    where $\boldsymbol{\alpha}_i = (\alpha_1, \ldots, \alpha_{i-1})$ denotes the vector of challenges sent in the first $i - 1$ rounds.
  - $\mathcal{P}$ sends $h_i(X)$.
  - $\mathcal{V}$ checks: If $i > 1$, $\mathcal{V}$ checks $h_i(0) + h_i(1) = h_{i-1}(\alpha_{i-1})$. For $i = 1$, it checks $h_1(0) + h_1(1) = \sigma$.
  - $\mathcal{V}$ sends $\alpha_i \leftarrow \mathbb{F}$.
- The prover sends the function $f' = \mathsf{fold}(f, (\alpha_1, \ldots, \alpha_k))$[5].
- The reduced claim is now $\sum_{\mathbf{b} \in \{0,1\}^{m-k}} \widetilde{w}(f'(\mathbf{b}), \boldsymbol{\alpha}, \mathbf{b}) = h_k(\alpha_k)$ where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_k)$.

[5]The prover only needs to send the message positions of the codeword, as they uniquely specify the function

In our presentation we consider only one iteration and $\mathcal{P}$ sends the function f' in full. In [ACFY25] f' is again sent as an oracle for subsequent iterations.

Query Phase: After $\mathcal{P}$ sends f' in the final step, the verifier makes queries to check consistency of f' and f, and that f' satisfies the evaluation claim at the end of the sum-check phase.

- $\mathcal{V}$ samples t random points $z_1, \ldots, z_t \leftarrow \mathsf{L}^{(2^k)}$ and queries $y_i = [\![\mathsf{fold}(f, \boldsymbol{\alpha})]\!](z_i)$ for $i \in [t]$. Note that the verifier simulates queries to $\mathsf{fold}(f, \boldsymbol{\alpha})$ using 2^k queries to $[\![f]\!]$.
- $\mathcal{V}$ checks $\widetilde{f'}(\mathbf{z}_i) = y_i$ for all $i \in [t]$, where $\mathbf{z}_i = (z_i, z_i^2, \ldots, z_i^{2^{m-k-1}})$.
- $\mathcal{V}$ checks reduced claim: $\sum_{\mathbf{b} \in \{0,1\}^{m-k}} \widetilde{w}(f'(\mathbf{b}), \boldsymbol{\alpha}, \mathbf{b}) = h_k(\alpha_k)$.

By replacing evaluation claims on $\widetilde{f'}$ with equivalent sum-check instances and batching the $t+1$ resulting sum-check instances using uniformly sampled $\gamma \leftarrow \mathbb{F}$, $\mathcal{V}$'s checks reduce to the following inner product:

$$\sum_{\mathbf{b} \in \{0,1\}^{m-k}} \widetilde{f'}(\mathbf{b}) \cdot \left(\widetilde{eq}(\boldsymbol{\alpha}, \mathbf{b}, \mathbf{z}) + \sum_{i=1}^{t} \gamma^i \cdot \widetilde{eq}(\mathbf{b}, \mathbf{z}_i) \right) = \sigma'$$

Thus, $\mathcal{V}$'s checks can be done in $O(t \cdot 2^{m-k})$ $\mathbb{F}$ operations. The soundness of the sum-check steps follows from the following Lemma based on [ACFY25, Lemma 2].

Lemma 3.2. *Let $C := \mathsf{CRS}[\mathbb{F}, \mathsf{L}, m, \widetilde{w}, \sigma]$ be a constrained Reed-Solomon code with rate $\rho = 2^m/|\mathsf{L}|$. Let $f : \mathsf{L} \to \mathbb{F}$ be a function such that $\Delta(f, C) > \delta$ for some $\delta \in (0, 1 - \sqrt{\rho})$. For $k < m$, let $h_1, \ldots, h_k$ be $\mathcal{P}$'s polynomials in $\mathbb{F}^{<3}[X]$ and $\alpha_1, \ldots, \alpha_k$ be $\mathcal{V}$'s challenges in an accepting transcript for the sum-check phase. Then $\Delta(\mathsf{fold}(f, \boldsymbol{\alpha}), C_k) > \delta$ with probability $\geq 1 - \mathrm{poly}(2^m, 1/\rho)/|\mathbb{F}|$, where $C_k := \mathsf{CRS}[\mathbb{F}, \mathsf{L}^{(2^k)}, m-k, \widetilde{w}_k, \sigma_k]$ with $\widetilde{w}_k(Z, \mathbf{X}) = Z \cdot \widetilde{w}(\alpha_1, \ldots, \alpha_k, \mathbf{X})$ and $\sigma_k = h_k(\alpha_k)$.*

Intuitively, Lemma 3.2 implies that if prover's initial function is δ-far from C, its folding remains δ-far from the constrained code determined by sum-check claim in later rounds. After k rounds, since the prover is forced to send a codeword in C_k, the consistency checks succeed with probability at most $(1-\delta)^t$. Thus we have:

Lemma 3.3. *For all $\delta \in (0, 1 - \sqrt{\rho})$, if $\Delta(f, \mathsf{CRS}[\mathbb{F}, \mathcal{L}, m, \widetilde{w}, \sigma]) > \delta$, then the verifier accepts in the above protocol with probability at most $(1-\delta)^t + \frac{\mathrm{poly}(\rho^{-1}, m)}{|\mathbb{F}|}$.*

In [ACFY25], the above result was proved under the assumption that Reed-Solomon codes satisfy *mutual correlated agreement* up to $\delta < 1 - \sqrt{\rho}$. In [ACFY25], the authors only proved the assumption for $\delta < (1-\rho)/2$; it has subsequently been proved for $\delta < 1 - \sqrt{\rho}$ in [Hab25]. For honestly encoded oracles, the soundness of WHIR iteration follows from much simpler arguments.

Evaluation Claims on Honest Oracles: We can strengthen the soundness bound in Lemma 3.3 when the oracle $[\![f]\!]$ is *known* to encode a valid codeword (e.g, it is output as part of one time pre-processing). In this case, the soundness of consistency check between f' and $\mathsf{fold}(f, \boldsymbol{\alpha})$ improves to ρ^t, where t is the number of queries. We state this below:

Lemma 3.4. *For $f \in \mathsf{RS}[\mathbb{F}, \mathsf{L}, m]$ and $f \notin \mathsf{CRS}[\mathbb{F}, \mathsf{L}, m, \widetilde{w}, \sigma]$, the verifier in WHIR iteration accepts with probability at most $\rho^t + O(k/|\mathbb{F}|)$.*

Proof. We observe that if $f \in \mathsf{RS}[\mathbb{F}, \mathsf{L}, m]$, we have $\mathsf{fold}(f, \boldsymbol{\alpha}) \in \mathsf{RS}[\mathbb{F}, \mathsf{L}', m-k]$. Therefore, if $f' \neq \mathsf{fold}(f, \boldsymbol{\alpha})$, we have $\Delta(\mathsf{fold}(f, \boldsymbol{\alpha}), f') \geq 1 - \rho$. Thus if $f' \neq \mathsf{fold}(f, \boldsymbol{\alpha})$, the consistency checks succeed with probability at most ρ^t. If f' is honest, then the soundness follows from the soundness of the sum-check protocol which in this case is $O(k/|\mathbb{F}|)$. □

The above result allows us to optimize proof sizes when querying trusted oracles.

3.8 Sum-Check over Group Polynomials

In this work, we consider polynomials over a group $\mathbb{G}$ of prime order written additively. In [BCS21], Bootle et al. considered generalization of sum-check protocol for polynomials over modules. We consider the special case when the module is a group $\mathbb{G}$ (viewed as a $\mathbb{F}$-module, in fact a $\mathbb{F}$-vector space). A univariate polynomial $\widehat{g}(X) \in \mathbb{G}[X]$ is given by $\widehat{g}(X) := \sum_{i=1}^{d} g_i X^{i-1}$ for some $g_1, \ldots, g_d \in \mathbb{G}$. A multilinear polynomial $\widetilde{g}(X_1, \ldots, X_m) \in \mathbb{G}[X_1, \ldots, X_m]$ is given by $\widetilde{g}(X_1, \ldots, X_m) = \sum_{\mathbf{b} \in \{0,1\}^m} g_{\mathbf{b}} \cdot \prod_{j=1}^{m} X_j^{b_j}$ for some $g_{\mathbf{b}} \in \mathbb{G}$. Note that the evaluations $\widehat{g}(a) \in \mathbb{G}$ and $\widetilde{g}(\mathbf{a}) \in \mathbb{G}$ for $a \in \mathbb{F}$ and $\mathbf{a} \in \mathbb{F}^m$ are well-defined. We can also see multiplication of a field polynomial with a group polynomial extends naturally by just taking MSMs of the coefficients .

By using the above discussion, we can define sum-check for group polynomials.

Lemma 3.5 ([BCS21]). *Let $\mathbb{G}$ be a group of prime order written additively. Let $\widetilde{g}(X_1, \ldots, X_m) \in \mathbb{G}[X_1, \ldots, X_m]$ and $\widetilde{f}(X_1, \ldots, X_m) \in \mathbb{F}[X_1, \ldots, X_m]$ be multilinear polynomials over the group $\mathbb{G}$ and field $\mathbb{F}$ respectively. Then, there exists an interactive protocol between a prover $\mathcal{P}$ and verifier $\mathcal{V}$ to verify the claim $\sum_{\mathbf{b} \in \{0,1\}^m} \widetilde{g}(\mathbf{b}) \cdot \widetilde{f}(\mathbf{b}) = \sigma$ for $\sigma \in \mathbb{G}$ where:*

- *$\mathcal{P}$ performs $O(2^m)$ group and field operations,*
- *$\mathcal{V}$ performs $O(m)$ group operations and field operations, in addition to one query each to polynomials $\widetilde{g}$ and $\widetilde{f}$.*
- *The protocol achieves perfect completeness and soundness error of $O(m/|\mathbb{F}|)$.*

We note that the messages sent by the prover in the above protocol are group elements describing the univariate group polynomials.

4 Titan: Overview

We now expand on the construction of Titan, our transparent polynomial commitment scheme sketched in Section 2. Let $\mathbb{G}$ denote a subgroup of prime order in an elliptic curve E, where the discrete-log problem is hard. We use $\mathbb{F}$ to denote the scalar field of prime order. We also assume $\mathcal{H}$ to be a collision resistant hash function $\mathcal{H} : \mathbb{G}^* \to \{0,1\}^{2\lambda}$. As seen earlier, the core technique in Titan is to decompose the evaluation claim into an evaluation claim over a group oracle, and a proof of knowledge using CSP. We set both instance sizes to be $\approx O(\sqrt{n})$. In this section, we fully specify parameters, which can be optimised for specific settings.

4.1 Setup

Let m denote the number of variables and $n = 2^m$ denote the size of the polynomials. Let $\mathbf{g} = (g_1, \ldots, g_q)$ be a vector of uniformly and independently sampled generators from $\mathbb{G}$. We consider group polynomials of two sizes - polynomials of size p obtained as commitment of the field polynomial and polynomial of size q as the MLE of generators $\mathbf{g}$. Observe that we have $n = p \cdot q$. We refer to former as the *outer* group polynomial, while to the latter as *inner* group polynomial. Next, we choose parameters to encode the group polynomials as Reed-Solomon codes. Let $\rho_1, \rho_2 \in (0,1)$ denote the rate of the Reed-Solomon codes used to encode the group polynomials. Accordingly, let $\mathsf{L}_1, \mathsf{L}_2$ be smooth multiplicative subgroups of $\mathbb{F}$ of sizes $\rho_1^{-1} \cdot p$ and $\rho_2^{-1} \cdot q$ respectively. Let $\ell_p = \log p$ and $\ell_q = \log q$ denote the number of variables outer and inner group polynomials respectively. We also define sub-domains K_1 and K_2 of L_1 and L_2 respectively to encode folded functions. For folding factors k_1 and k_2, we set $\ell_i = 2^{k_i}$, $i \in [2]$ and define $\mathsf{K}_i = \mathsf{L}_i^{(\ell_i)}$.

	Polynomial-size parameters
m	Number of variables; $n = 2^m$ is the polynomial size
p, q	Sizes of the two families of group polynomials; $n = p \cdot q$
$\ell_p = \log p$	Number of variables in the outer ($\widetilde{G}$) group polynomial
$\ell_q = \log q$	Number of variables in the inner/generator ($\widetilde{g}$) group polynomial
$\mathbf{g} = (g_1, \ldots, g_q)$	Uniformly sampled generators from $\mathbb{G}$ (commitment key)
	Reed-Solomon encoding parameters
$\rho_1, \rho_2 \in (0, 1)$	Code rates for the outer and inner RS codes
$\mathsf{L}_1 \subseteq \mathbb{F}$	Smooth multiplicative domain of size $\rho_1^{-1} \cdot p$ (outer)
$\mathsf{L}_2 \subseteq \mathbb{F}$	Smooth multiplicative domain of size $\rho_2^{-1} \cdot q$ (inner)
	Folding / coset parameters
k_1, k_2	Folding factors for outer and inner codes
ℓ_1, l_2	Coset size (folding arity) for code $i \in \{1, 2\}$
$m_1 = p/\ell_1$	Size of folded outer polynomial
$m_2 = q/\ell_2$	Size of folded inner polynomial
$\mathsf{K}_i = \mathsf{L}_i^{(\ell_i)}$	Folded sub-domain (ℓ_i-th powers of L_i)
t_1, t_2	Number of consistency check queries in outer and inner group IOPPs
	Oracles
$[\![g]\!]$	WHIR oracle committing to generator polynomial $\widetilde{g}$; locally held by $\mathcal{V}$

Figure 3: Parameter summary for the Titan PCS

Encoding of Generators. The verifier also computes one-time encoding of the polynomial $\widetilde{g}(\mathbf{X})$ interpolating the generators $\mathbf{g}$. Following the coset-wise encoding in Section 3.5.1, the oracle $[\![g]\!]$ when queried on $y \in \mathsf{K}_2$ returns the evaluations $\big(\widetilde{g}(\mathbf{b}, \mathbf{y})\big)_{\mathbf{b} \in \{0,1\}^{k_2}}$, where $\mathbf{y} = (y, y^2, \ldots, y^{2^{\ell_q - k_2 - 1}}) \in \mathbb{F}^{\ell_q - k_2}$. We set $\ell_2 = O(\sqrt{q})$ and thus $|\mathsf{K}_2| = O(\sqrt{q})$. When we set $q = O(\sqrt{n})$, both ℓ_2 and $|\mathsf{K}_2|$ are $O(\sqrt[4]{n})$. We summarize the parameters in Figure 3 for quick reference.

4.2 Commitment

Let $\widetilde{f} \in \mathbb{F}^{\leq 1}[X_1, \ldots, X_m]$ be a multilinear polynomial. Like prior work such as Hyrax [WTs+18] and Dory [Lee21], we view $\widetilde{f}$ as $p \times q$ matrix, with entries $f_{ij} := \widetilde{f}(\langle i \rangle, \langle j \rangle)$.

$$\begin{matrix} g_1 & g_2 & \cdots & g_q \end{matrix}$$
$$\begin{pmatrix} f_{11} & f_{12} & \cdots & f_{1q} \\ f_{21} & f_{22} & \cdots & f_{2q} \\ \vdots & \vdots & \ddots & \vdots \\ f_{p1} & f_{p2} & \cdots & f_{pq} \end{pmatrix} \begin{matrix} G_1 \\ G_2 \\ \vdots \\ G_p \end{matrix}$$

The above matrix view is captured by the following decomposition:

$$\widetilde{f}(\mathbf{X}, \mathbf{Y}) = \sum_{i=1}^{p} \widetilde{eq}(\mathbf{X}, \langle i \rangle) \cdot \widetilde{f}(\langle i \rangle, \mathbf{Y}) \tag{7}$$

Here $\mathbf{X}$ and $\mathbf{Y}$ denote blocks of variables $(X_1, \ldots, X_{\ell_p})$ and $(Y_1, \ldots, Y_{\ell_q})$ respectively, while $\langle i \rangle$ denotes the point in $\{0, 1\}^{\ell_p}$ canonically determined by i. Using $\mathbf{g}$ as a commitment key, we define the homomorphic commitment to polynomial $\widetilde{h} \in \mathbb{F}^{\leq 1}[X_1, \ldots, X_{\ell_q}]$ as:

$$\mathsf{Com}(\widetilde{h}\,;\, \mathbf{g}) = \sum_{i=1}^{q} \widetilde{h}(\langle i \rangle) \cdot g_i \tag{8}$$

Now, let $\widetilde{f}_i(\mathbf{Y}) := \widetilde{f}(\langle i \rangle, \mathbf{Y})$ denote the ith polynomial in the decomposition of $\widetilde{f}$ in Equation (7). Let $G_i = \mathsf{Com}(\widetilde{f}_i\,; \mathbf{g})$

be the commitment to $\widetilde{f}_i$ for $i \in [p]$. At this point, we compute an *outer commitment* over the vector $(G_1, \ldots, G_p)$, which commits to $\widetilde{f}$. Instead of relying on bilinear pairings, we interpolate a group polynomial $\widetilde{G}(\mathbf{X})$ such that $\widetilde{G}(\langle i \rangle) = G_i$ for $i \in [p]$ and commit to $\widetilde{G}$ as an oracle $[\![G]\!]$. Formally, we have:

$$\widetilde{G}(\mathbf{X}) = \sum_{i=1}^{p} \widetilde{eq}(\mathbf{X}, \langle i \rangle) \cdot G_i \tag{9}$$

Next, we specify the encoding of $\widetilde{G}$ as an oracle.

Encoding of $\widetilde{G}$. As in Section 3.5.1, we commit to $\widetilde{G}$ as evaluations over cosets for $y \in \mathsf{K}_1$. For $y \in \mathsf{K}_1$, let $\mathbf{y}$ denote the vector $(y, y^2, \ldots, y^{2^{\ell_p - k_1 - 1}}) \in \mathbb{F}^{\ell_p - k_1 - 1}$. Then for each $y \in \mathsf{K}_1$, the oracle $[\![G]\!]$ returns the vector $\left(\widetilde{G}(\mathbf{b}, \mathbf{y})\right)_{\mathbf{b} \in \{0,1\}^{k_1}}$. To realize the oracle $[\![G]\!]$, we commit G as a Merkle tree with $m_1 := |\mathsf{K}_1|$ leaves, each being a vector of size 2^{k_1}. For our construction, we set $\ell_1, m_1 \approx O(\sqrt{p})$. When $p = O(\sqrt{n})$, we have $\ell_1, m_1 = O(\sqrt[4]{n})$.

4.3 Evaluation Protocol for Titan PCS

In the evaluation PIOP, the verifier $\mathcal{V}(\mathbf{z}, \sigma, [\![G]\!])$ with inputs $(\mathbf{z}, \sigma)$ and oracle access $[\![G]\!]$ needs to be convinced that $[\![G]\!]$ encodes a multilinear polynomial $\widetilde{G}$, which in turn commits to the field polynomial $\widetilde{f} \in \mathbb{F}[X_1, \ldots, X_m]$ satisfying $\widetilde{f}(\mathbf{z}) = \sigma$. Let us parse $\mathbf{z} = (\mathbf{x}, \mathbf{y})$ with $\mathbf{x} \in \mathbb{F}^{\ell_p}$ and $\mathbf{y} \in \mathbb{F}^{\ell_q}$. Since $\widetilde{G}$ commits to $\widetilde{f}$ row wise, by homomorphism of the Pedersen commitment, $\widetilde{G}(\mathbf{x})$ is the Pedersen commitment of the polynomial $\widetilde{f}(\mathbf{x}, \mathbf{Y})$. Therefore, we have the following meta-protocol for evaluation:

1. $\mathcal{P}$ sends commitment C (purportedly to $\widetilde{f}(\mathbf{x}, \mathbf{Y})$).
2. $\mathcal{P}$ and $\mathcal{V}$ execute WHIR IOPP to check that $\widetilde{G}(\mathbf{x}) = C$, for $\widetilde{G}$ encoded by $[\![G]\!]$.
3. $\mathcal{P}$ shows knowledge of field polynomial with coefficients $\mathbf{a} \in \mathbb{F}^q$ via the *proof of knowledge* protocol:

$$\mathrm{PoK}\left\{(C, \sigma, \mathbf{g}; \mathbf{a}) : \sum_{i=1}^{q} a_i \cdot g_i = C \;\wedge\; \sum_{i=1}^{q} a_i \cdot \widetilde{eq}(\mathbf{y}, \langle i \rangle) = \sigma\right\}$$

Decoding G to a polynomial: More precisely, the second step of the meta-protocol involves choosing a proximity parameter $\delta \in (0, 1)$ followed by $\mathcal{P}$ and $\mathcal{V}$ invoking WHIR IOPP to check that G is δ-close to $\mathsf{CRS}[\mathbb{G}, \mathsf{L}_1, \ell_p, \widetilde{w}, \sigma]$, where $\widetilde{w}(Z, \mathbf{X}) = Z \cdot \widetilde{eq}(\mathbf{X}, \mathbf{x})$.

We choose an IOPP involving *single* WHIR iteration, which concretely leads to better proof sizes (our oracles are $O(\sqrt{n})$ to start with). Further, note that an IOPP only ensures that $[\![G]\!]$ is close to "some" polynomial satisfying the evaluation claim. For a polynomial commitment scheme we need to bind G to a unique polynomial, for which there are two popular approaches:

- *Unique Decoding Regime*: In this regime, the proximity parameter δ is set below the unique decoding radius of the code, i.e, $\delta < (1 - \rho_1)/2$, which ensures there is a unique polynomial δ-close to G and hence all evaluation claims hold with respect to this polynomial. This regime incurs larger proof sizes due to higher query complexity to achieve target soundness.
- *List Decoding Regime*: In this regime, we use proximity parameter beyond the unique decoding radius

all the way up to $1-\sqrt{\rho_1}-\gamma$ (Johnson's radius) for some $\gamma > 0$. Using the standard technique of *out of domain* sample, the verifier can bind prover to a unique polynomial in the δ-ball for evaluation proofs.

We describe and prove the soundness of Titan PCS in the unique decoding regime, choosing to focus more on the interplay of group based IOPPs with compressed sigma protocol to achieve overall soundness (and knowledge-soundness). The construction and security proofs can be adapted to list-decoding regime in a standard manner which we sketch in Section 5.4. Our implementation and benchmarks are with respect to proximity parameters under list-decoding regime.

5 Titan PCS: Complete Protocol

We elaborate on the protocols for two checks required by $\mathcal{V}$ in the final two steps of the meta-protocol in Section 4.3.

5.1 Outer Group Polynomial Evaluation

In Figure 4, we present an adaptation of WHIR iteration to group polynomials to check G is δ-close to $\mathsf{CRS}[\mathbb{G}, \mathsf{L}_1, \ell_p, \widetilde{w}, C]$ given $\delta \in (0,1)$, $\widetilde{w}(Z, \mathbf{X}) = Z.\widetilde{eq}(\mathbf{X}, \mathbf{x})$ for $\mathbf{x} \in \mathbb{F}^{\ell_p}$ and $C \in \mathbb{G}$. We have the following:

- **Public Parameters**: As in Figure 3.
- **Common Input**: $(\mathbf{x}, C, \delta, [\![G]\!])$.
- **Prover's Input**: $\widetilde{G} \in \mathbb{G}^{\leq 1}[X_1, \ldots, X_{\ell_p}]$.
- **Protocol (Π_{outer})**:
  1. In round i for $i \in [k_1]$:
     - $\mathcal{P}$ computes $h_i(X) = \sum_{\mathbf{b} \in \{0,1\}^{m_1 - i}} \widetilde{w}(\widetilde{G}(\boldsymbol{\alpha}_i, X, \mathbf{b}), \boldsymbol{\alpha}_i, X, \mathbf{b})$, where $\boldsymbol{\alpha}_i = (\alpha_1, \ldots, \alpha_{i-1})$ denotes the vector of challenges sent in the first $i-1$ rounds.
     - $\mathcal{P}$ sends $h_i(X)$.
     - $\mathcal{V}$ checks: If $i > 1$, $\mathcal{V}$ checks $h_i(0) + h_i(1) \stackrel{?}{=} h_{i-1}(\alpha_{i-1})$. For $i = 1$, it checks $h_1(0) + h_1(1) \stackrel{?}{=} C$.
     - $\mathcal{V}$ sends $\alpha_i \leftarrow \mathbb{F}$.
  2. $\mathcal{P}$ sends evaluations $(c_1, \ldots, c_{m_1})$ of the polynomial $\widetilde{G}'(\cdot) = \widetilde{G}(\boldsymbol{\alpha}, \cdot)$, where $\boldsymbol{\alpha} = \boldsymbol{\alpha}_{k_1+1}$.
  3. $\mathcal{V}$ samples t random points $z_1, \ldots, z_t \leftarrow \mathsf{K}_1$.
  4. $\mathcal{V}$ obtains $\mathbf{v}_i \in \mathbb{G}^{\ell_1}$ by querying $[\![G]\!]$ at z_i for all $i \in [t]$.
  5. $\mathcal{V}$ checks consistency of $\widetilde{G}'$ with $[\![G]\!]$: $\forall i \in [t]$
     - Check $\sum_{j=1}^{m_1} c_j \cdot \widetilde{eq}(\langle j \rangle, \mathbf{z}_i) \stackrel{?}{=} \sum_{j=1}^{\ell} \mathbf{v}_i[j] \cdot \widetilde{eq}(\langle j \rangle, \alpha)$ where $\mathbf{z}_i = (z_i, z_i^2, \ldots, z_i^{2^{\ell_p - k_1 - 1}})$.
  6. $\mathcal{V}$ checks reduced sum-check claim:
     - Check: $\sum_{j=1}^{m_1} c_j \cdot \widetilde{eq}(\boldsymbol{\alpha}, \langle j \rangle; \mathbf{x}) \stackrel{?}{=} h_{k_1}(\alpha_{k_1})$.
  7. $\mathcal{V}$ accepts if all the checks pass, else it rejects.

Figure 4: IOPP for Outer Group Polynomial Evaluation

Lemma 5.1. *For $\delta \in (0, 1 - \sqrt{\rho_1})$, query bound t and other parameters as in Figure 3, the protocol in Figure 4 accepts with probability 1 whenever $G \in \mathsf{CRS}[\mathbb{G}, \mathsf{L}_1, \ell_p, \widetilde{w}, C]$. For G which is δ-far from $\mathsf{CRS}[\mathbb{G}, \mathsf{L}_1, \ell_p, \widetilde{w}, C]$ it rejects e.w.p. $\approx (1-\delta)^t$. The prover incurs MSM of size $O(p)$ in $\mathbb{G}$, while $\mathcal{V}$ incurs MSM of size $m_1 + t \cdot \ell_1$ and $t \cdot (m_1 + \ell_1)$ $\mathbb{F}$ operations. The communication is $3k_1 + m_1 + t\ell_1$ $\mathbb{G}$ elements.*

Proof. The completeness of the protocol can be routinely verified. We show soundness. Assume that an adversarial prover $\mathcal{P}^*$ produces a transcript

$$(G, h_1, \alpha_1, \ldots, h_k, \alpha_k, \widetilde{G}').$$

We show that $\mathcal{V}$ rejects the transcript if $\Delta(G, \widetilde{G}) > \delta$ for all $\widetilde{G}$ satisfying $\widetilde{G}(\mathbf{x}) = C$. Let $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_k)$ denote the vector of $\mathcal{V}$'s challenges. If one of the sum-check conditions fail, $\mathcal{V}$ rejects. Assume that polynomials $h_1, \ldots, h_k$ satisfy sum-check verification constraints. Then, by Lemma 3.2, for $\delta < 1 - \sqrt{\rho}$, if G is δ-far from every polynomial $\widetilde{H}$ satisfying $\widetilde{H}(\mathbf{x}) = C$, with overwhelming probability $\mathsf{fold}(G, \boldsymbol{\alpha})$ is δ-far from every polynomial $\widetilde{H}'$ satisfying $\sum_{i=1}^{m} \widetilde{H}'(\langle i \rangle) \cdot \widetilde{eq}(\boldsymbol{\alpha}, \langle i \rangle; \mathbf{x}) = h_k(\alpha_k)$. Clearly, in this case either $\widetilde{G}'$ fails the reduced sum-check claim in Step (6), or the consistency check succeeds with probability at most $(1-\delta)^t$. This shows that the protocol has soundness error $\approx (1-\delta)^t$. The verification in Figure 4 checks $t+1$ inner products. The verifier uses a random $\gamma \leftarrow \mathbb{F}$ to batch the checks involving group vectors resulting in MSM of size $t \cdot \ell_1 + m_1$ and $t \cdot (m_1 + \ell_1)$ $\mathbb{F}$ operations. The communication complexity can also be easily verified. □

5.2 Evaluation of partial polynomial

To complete the evaluation proof, we need to show that $\widetilde{a}(\mathbf{y}) = \sigma$ where the polynomial $\widetilde{a}$ opens the commitment C sent by the prover in the meta-protocol in Section 4.3. We can formulate this as a Compressed Sigma Protocol [AC20], where the prover proves knowledge of coefficient vector $(a_1, \ldots, a_q)$ such that:

$$\sum_{i=1}^{q} a_i \cdot g_i = C \quad \wedge \quad \sum_{i=1}^{q} a_i \cdot \widetilde{eq}(\langle i \rangle, \mathbf{y}) = \sigma \tag{10}$$

We note that the linear form evaluating to σ is dictated by our interpretation of $(a_1, \ldots, a_q)$ as evaluations of $\widetilde{a}$ over the boolean hypercube $\{0,1\}^{\ell_q}$. For the folding parameter $k := k_2$ (see Figure 3) the compressed sigma protocol for Equation 10 proceeds in k rounds; in each round the instance is reduced to a smaller instance using verifier's challenges. After k rounds, the prover sends the witness for the reduced instance in *full*. We present the protocol in Figure 5. The prover sends the folded vectors $\mathbf{g}_k$ and $\mathbf{a}_k$ after k rounds. In standard CSP the vectors are folded by splitting them in the middle and combining the two parts using the verifier's challenge. In a slight departure, we split them as consisting of *even* and *odd* positions, which is conceptually equivalent. Our split and fold leads to simpler algebraic interpretation: If $\widetilde{g}_i(\cdot)$ is a multilinear polynomial which interpolates the round i vector $\mathbf{g}_i$, the multilinear polynomial $\widetilde{g}(\alpha, \cdot)$ interpolates the round $i+1$ vector $\mathbf{g}_{i+1}$ with α being the folding challenge. After k rounds, the verifier checks correctness of $\mathbf{g}_k$ by querying the oracle $[\![g]\!]$ folded by challenges $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_k)$. The folding of the vector describing the linear form is easier to describe. In a CSP, when $(\alpha, 1-\alpha)$ is used to fold the generators, the reverse $(1-\alpha, \alpha)$ is applied to fold the linear form. Thus the linear form $\mathbf{u}_k$ is given as evaluations $\widetilde{eq}(\overline{\boldsymbol{\alpha}}, \langle j \rangle; \mathbf{y})$ for $j \in [m_2]$ where $\overline{\boldsymbol{\alpha}} = (1-\alpha_1, \ldots, 1-\alpha_k)$. The complete protocol appears in Figure 5.

Lemma 5.2. *Let* pp *denote the public parameters in Figure 3. Let* $k := k_2$, $\rho := \rho_2$ *and* $\mathcal{R}_{\text{inner}}(\mathsf{pp})$ *denote the relation consisting of tuples* $((C, \mathbf{y}, \sigma); \mathbf{a} \in \mathbb{F}^q)$ *such that* $\sum_{i=1}^{q} a_i \cdot g_i = C$ *and* $\sum_{i=1}^{q} a_i \cdot \widetilde{eq}(\langle i \rangle, \mathbf{y}) = \sigma$. *Then, assuming hardness of discrete log in* $\mathbb{G}$, *the protocol* Π_{inner} *is* complete *and* $(\mu_1, \ldots, \mu_k)$*-special sound for* $\mu_i = 3$ *for relation* $\mathcal{R}_{\text{inner}}$ *with soundness error* $\varepsilon + 3^k \cdot \rho^t$. *Here* ε *denotes the soundness error of standard CSP for* $\mathcal{R}_{\text{inner}}$, *where* $\mathcal{V}$ *also folds the generators.*

- **Public Parameters**: pp as in Figure 3.
- **Common Input**: pp, $C \in \mathbb{G}$, $\sigma \in \mathbb{F}$.
- **$\mathcal{P}$'s Input**: $\mathbf{a} \in \mathbb{F}^q$.
- **Protocol** (Π_{inner}):

1. $\mathcal{P}$ and $\mathcal{V}$ set $\mathbf{g}_0 = \mathbf{g}$, $\mathbf{a}_0 = \mathbf{a}$, $\mathbf{u}_0 = (\widetilde{eq}(\langle j \rangle, \mathbf{y}))_{j \in [q]}$.
2. Set $k := k_2$. For $i = 0$ to $k-1$:
   - $\mathcal{P}$ computes $\mathbf{g}_i^e$, $\mathbf{g}_i^o$ of length $q/2^{i+1}$ given by $\mathbf{g}_i^e[\,j\,] = \mathbf{g}[\,2j\,]$ and $\mathbf{g}_i^o[\,j\,] = \mathbf{g}[\,2j+1\,]$ for $j \in [q/2^{i+1}]$. Similarly, it computes vectors $\mathbf{a}_i^e$, $\mathbf{a}_i^o$, $\mathbf{u}_i^e$, and $\mathbf{u}_i^o$.
   - $\mathcal{P}$ computes:
   $$L_i = \langle \mathbf{a}_i^o, \mathbf{g}_i^e \rangle,\ R_i = \langle \mathbf{a}_i^e, \mathbf{g}_i^o \rangle,\ \ell_i = \langle \mathbf{a}_i^o, \mathbf{u}_i^e \rangle,\ r_i = \langle \mathbf{a}_i^e, \mathbf{u}_i^o \rangle$$
   - $\mathcal{P}$ sends L_i, R_i, ℓ_i, r_i.
   - $\mathcal{V}$ sends $\alpha_i \leftarrow \mathbb{F}$.
   - $\mathcal{P}$ and $\mathcal{V}$ update commitment and linear form:
   $$C_{i+1} = \alpha_i(1-\alpha_i) \cdot C_i + (1-\alpha_i)^2 \cdot L_i + \alpha_i^2 \cdot R_i,$$
   $$\sigma_{i+1} = \alpha_i(1-\alpha_i) \cdot \sigma_i + (1-\alpha_i)^2 \cdot \ell_i + \alpha_i^2 \cdot r_i$$
   - $\mathcal{P}$ folds vectors: $\mathbf{a}_{i+1} = \alpha_i \mathbf{a}_i^e + (1-\alpha_i) \cdot \mathbf{a}_i^o$, $\mathbf{g}_{i+1} = (1-\alpha_i)\mathbf{g}_i^e + \alpha_i \cdot \mathbf{g}_i^o$, $\mathbf{u}_{i+1} = (1-\alpha_i)\mathbf{u}_i^e + \alpha_i \cdot \mathbf{u}_i^o$.
3. After k rounds, $\mathcal{P}$ sends $\mathbf{g}_k \in \mathbb{G}^{m_2}$, $\mathbf{a}_k \in \mathbb{F}^{m_2}$.
4. $\mathcal{V}$ samples $z_1, \ldots, z_t \in \mathsf{K}_2$. It queries $\mathbf{v}_i := [\![\mathsf{fold}(\widetilde{g}, \boldsymbol{\alpha})]\!](z_i)$.
5. $\mathcal{V}$ checks consistency of $\mathbf{g}_k$ with $\widetilde{g}$: Let $\mathbf{z}_i = (z_i, z_i^2, \ldots, z_i^{2^{\ell_q - k - 1}})$, $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_k)$, $\overline{\boldsymbol{\alpha}} = (1-\alpha_1, \ldots, 1-\alpha_k)$.
   - $\forall i \in [t]$: $\sum_{j=1}^{m_2} \mathbf{g}_k[\,j\,] \cdot \widetilde{eq}(\langle j \rangle, \mathbf{z}_i) \stackrel{?}{=} \sum_{j=1}^{m_2} \mathbf{v}_i[\,j\,] \cdot \widetilde{eq}(\langle j \rangle, \boldsymbol{\alpha})$.
   - $\mathcal{V}$ checks reduced instance:
   $$\sum_{j=1}^{m_2} \mathbf{g}_k[\,j\,] \cdot \mathbf{a}_k[\,j\,] \stackrel{?}{=} C_k,\ \sum_{j=1}^{m_2} \mathbf{a}_k[\,j\,] \cdot \widetilde{eq}(\langle j \rangle, \overline{\boldsymbol{\alpha}}; \mathbf{y}) \stackrel{?}{=} \sigma_k$$
6. $\mathcal{V}$ accepts if all the checks succeed.

Figure 5: CSP for partial evaluation

Proof. The completeness may be verified straightforwardly. We prove special soundness using the standard tree of accepting transcripts argument. When $\mathcal{V}$ also folds the generators, the protocol in Figure 5 is an instance of standard compressed sigma protocol [AC20]. Consider a tree of 3^k accepting transcripts by invoking the CSP tree finding algorithm with rewinding access to adversarial prover $\mathcal{P}^*$. Let $\boldsymbol{\alpha}^{(i)}$ denote the vector of challenges for i^{th} transcript and $\mathbf{g}_k^{(i)}$ denote the vector $\mathbf{g}_k$ in the ith transcript. Let E_i, $i \in [3^k]$ denote the event that $\mathbf{g}_k^{(i)} \not\equiv \widetilde{g}(\boldsymbol{\alpha}^{(i)}, \cdot)$, i.e, the vector $\mathbf{g}_k^{(i)}$ does not correspond to correct folding of the generators. If no event E_i occurs, the tree of transcripts is identical to the one in standard CSP, and thus witness can be extracted e.w.p. ε. Since we assume $\mathbf{g}$ (which is part of pp) is honestly encoded as a codeword in $\mathsf{RS}[\mathbb{G}, \mathsf{L}_2, \log q]$, if $\mathbf{g}_k^{(i)}$ is not identical to coefficients of $\widetilde{g}(\boldsymbol{\alpha}, \cdot)$, they determine different codewords in $\mathsf{RS}[\mathbb{G}, \mathsf{K}_2, \log m_2]$. It can be seen that the rate of $\mathsf{RS}[\mathbb{G}, \mathsf{K}_2, \log m_2]$ is also ρ. Hence, the consistency check in Step (5) failing to catch the discrepancy is at most $(1-\delta)^t = \rho^t$. Thus, for a fixed i, $\Pr[E_i]$ is upper bound by ρ^t. Thus probability that one of the E_i's occurs is $\leq 3^k \cdot \rho^t$. This implies a soundness bound of $\varepsilon + 3^k \cdot \rho^t$ for the protocol. □

5.3 Complete Protocol

The evaluation protocol for Titan PCS can be now described as a sequential composition of protocols Π_{outer} and Π_{inner} as we describe in Figure 6. We summarise the construction of Titan PCS in Figure 7. The properties are formally captured in Theorem 5.1.

- **Public Parameters**: pp as in Figure 3 and $\delta := (1-\rho_1)/2$.
- **Common Input**: $\mathbf{z} \in \mathbb{F}^m$, $\sigma \in \mathbb{F}$, $[\![G]\!]$.
- **Prover's Input**: $\widetilde{f} \in \mathbb{F}^{\leq 1}[X_1, \ldots, X_m]$, $\widetilde{G} = \mathsf{Com}(\widetilde{f}\,;\, \mathbf{g})$.
- **Evaluation Protocol ($\Pi_{\mathbf{Eval}}$)**: Parse $\mathbf{z}$ as $(\mathbf{x}, \mathbf{y})$ with $\mathbf{x} \in \mathbb{F}^{\ell_p}$ and $\mathbf{y} \in \mathbb{F}^{\ell_q}$.
  1. $\mathcal{P}$ sends commitment $C = \mathsf{Com}(\widetilde{f}(\mathbf{x}, \mathbf{Y})\,;\, \mathbf{g})$ to the partial evaluation $\widetilde{f}(\mathbf{x}, \mathbf{Y})$.
  2. $b_{\text{outer}} \leftarrow \langle \mathcal{P}(\mathsf{pp}, \mathbf{x}, C, \delta, [\![G]\!]; \widetilde{G}, \widetilde{f}), \mathcal{V}_{\text{outer}}(\mathsf{pp}, \mathbf{x}, C, \delta, [\![G]\!]) \rangle$
  3. $b_{\text{inner}} \leftarrow \langle \mathcal{P}(\mathsf{pp}, C, \mathbf{y}, \sigma; \mathbf{a}), \mathcal{V}_{\text{inner}}(\mathsf{pp}, C, \mathbf{y}, \sigma) \rangle$.
  4. $\mathcal{V}$ returns $b_{\text{outer}} \wedge b_{\text{inner}}$

Figure 6: Titan Evaluation Protocol

- **Setup**$(1^\lambda, m)$: Outputs pp in Figure 3 as described in Section 4.1. Set $\delta := (1-\rho_1)/2$.
- **Commit**$(\mathsf{pp}, \widetilde{f}) \to [\![G]\!]$: Outputs $[\![G]\!]$ where G is constructed as in Section 4.2.
- **Open**$(\mathsf{pp}, [\![G]\!]; \widetilde{f}) \to b$: Computes $\widetilde{G} = \mathsf{Com}(\widetilde{f}\,;\, \mathbf{g})$ as in Section 4.2 and outputs $b = 1$ if $\Delta(\widetilde{G}, G) < \delta$. Outputs $b = 0$ otherwise.
- **Eval**: The evaluation protocol $\langle \mathcal{P}_{\mathbf{Eval}}(\widetilde{f}), \mathcal{V}_{\mathbf{Eval}} \rangle(\mathsf{pp}, [\![G]\!], \mathbf{z}, \sigma)\rangle$ is given by Figure 6.

Figure 7: Titan PCS under Unique Decoding Regime

Before stating the theorem we record the guarantees of the two constituent protocols as games. Throughout, $\langle \mathcal{P}^*, \mathcal{V} \rangle(x)$ denotes the output bit of $\mathcal{V}$ after interacting with $\mathcal{P}^*$ on common input x, as in Section 3.2. For the outer protocol, define the relation and language

$$\mathcal{R}_{\text{outer}} := \left\{ \left((\mathsf{pp}, [\![G]\!], \mathbf{x}, C, \delta);\ \widetilde{G} \right) \ :\ \widetilde{G} \in \mathsf{RS}[\mathbb{G}, \mathsf{L}_1, \ell_p] \ \wedge\ \Delta(G, \widetilde{G}) < \delta \ \wedge\ \widetilde{G}(\mathbf{x}) = C \right\}, \tag{11}$$

where G denotes the word held by the oracle $[\![G]\!]$, and let $\mathcal{L}_{\text{outer}}$ be the set of statements $(\mathsf{pp}, [\![G]\!], \mathbf{x}, C, \delta)$ for which such a $\widetilde{G}$ exists. Note that membership asserts two things at once: that G is δ-close to the code, and that the unique nearby codeword takes the value C at $\mathbf{x}$.

Lemma 5.3 (Π_{outer}, restated from Lemma 5.1)**.** *For any adversarial prover $\mathcal{P}^*$ and any statement $(\mathsf{pp}, [\![G]\!], \mathbf{x}, C, \delta)$ fixed before $\mathcal{V}_{\text{outer}}$ samples its randomness,*

$$\Pr\left[\begin{array}{c} \left\langle \mathcal{P}^*,\, \mathcal{V}_{\text{outer}}(\mathsf{pp}, \mathbf{x}, C, \delta, [\![G]\!]) \right\rangle = 1 \\ \bigwedge \\ (\mathsf{pp}, [\![G]\!], \mathbf{x}, C, \delta) \notin \mathcal{L}_{\text{outer}} \end{array} \right] \leq\ \varepsilon_{\text{outer}} := (1-\delta)^{t_1},$$

the probability being over the randomness of $\mathcal{V}_{\text{outer}}$.

Proof. If $(\mathsf{pp}, [\![G]\!], \mathbf{x}, C, \delta) \notin \mathcal{L}_{\text{outer}}$ then G is δ-far from $\mathsf{CRS}[\mathbb{G}, \mathsf{L}_1, \ell_p, \widetilde{w}, C]$ with $\widetilde{w}(Z, \mathbf{X}) = Z \cdot \widetilde{eq}(\mathbf{X}, \mathbf{x})$: either G is δ-far from the code outright, or the unique nearby codeword fails the evaluation constraint, and in both cases no codeword in the δ-ball satisfies the constraint. Lemma 5.1 then bounds the probability that $\mathcal{V}_{\text{outer}}$ accepts in the interaction by $(1-\delta)^{t_1}$. □

Lemma 5.4 (Π_{inner}, restated from Lemma 5.2)**.** *There is an expected polynomial time extractor* $\mathsf{Ext}_{\text{inner}}$*, with rewinding access to* $\mathcal{P}^*$*, such that for any adversarial prover* $\mathcal{P}^*$ *and any statement* $(\mathsf{pp}, C, \mathbf{y}, \sigma)$*,*

$$\Pr\left[\begin{array}{c} \langle \mathcal{P}^*, \mathcal{V}_{\text{inner}}(\mathsf{pp}, C, \mathbf{y}, \sigma)\rangle = 1 \\ \wedge \\ \big((C, \mathbf{y}, \sigma);\ \mathbf{a}\big) \notin \mathcal{R}_{\text{inner}} \end{array} \middle| \ \mathbf{a} \leftarrow \mathsf{Ext}^{\mathcal{P}^*}_{\text{inner}}(\mathsf{pp}, C, \mathbf{y}, \sigma) \right] \ \leq\ \varepsilon_{\text{inner}} := \varepsilon_{\text{CSP}} + 3^{k_2}\rho_2^{t_2},$$

the probability being over the randomness of $\mathcal{V}_{\text{inner}}$ *in the interaction and that of* $\mathsf{Ext}_{\text{inner}}$*.*

Proof. Immediate from Lemma 5.2: Π_{inner} is $(3, \ldots, 3)$-special sound for $\mathcal{R}_{\text{inner}}$ with soundness error $\varepsilon_{\text{CSP}} + 3^{k_2}\rho_2^{t_2}$, and the tree-finding algorithm of the compressed sigma protocol, run with rewinding access to $\mathcal{P}^*$, is the required extractor: from a 3^{k_2}-tree of accepting transcripts it returns $\mathbf{a}$ with $\big((C, \mathbf{y}, \sigma); \mathbf{a}\big) \in \mathcal{R}_{\text{inner}}$, so the event that the interaction accepts while extraction fails is bounded by the stated soundness error. □

Theorem 5.1. *The scheme* Titan *PCS consisting of algorithms* (**Setup**, **Commit**, **Open**, **Eval**) *as in Figure 7 for polynomials in* $\mathbb{F}^{\leq 1}[X_1, \ldots, X_m]$ *satisfies* completeness, binding, *and* knowledge soundness with random point evaluation *with soundness error*

$$\varepsilon_{\text{RPE}} \ \leq\ (p+1)\big(\varepsilon_{\text{outer}} + \varepsilon_{\text{inner}}\big) + \varepsilon_{\text{DLOG}} + \frac{p \log p}{|\mathbb{F}|}$$

for $\varepsilon_{\text{outer}}$ *and* $\varepsilon_{\text{inner}}$ *as in Lemmas 5.3 and 5.4. Here* t_1 *and* t_2 *denote number of consistency queries in* Π_{outer} *and* Π_{inner} *respectively, while* ε_{CSP} *and* $\varepsilon_{\text{DLOG}}$ *denote soundness of standard CSP and adversary's success probability in breaking discrete-log in* $\mathbb{G}$ *respectively.*

Proof. We skip completeness as it follows trivially by construction. To prove binding, suppose PPT adversary $\mathcal{A}$ outputs $([\![G]\!], \widetilde{f_0}, \widetilde{f_1})$ in the binding game. If $\widetilde{f_0}$ and $\widetilde{f_1}$ are two distinct polynomials in $\mathbb{F}^{\leq 1}[X_1, \ldots, X_m]$ satisfying $\mathbf{Open}(\mathsf{pp}, G; \widetilde{f_i}) = 1$ for $i = 0, 1$; by definition of **Open** we must have $\Delta(G, \widetilde{G_i}) < \delta$ for $\widetilde{G_i} = \mathsf{Com}(\widetilde{f_i}\,; \mathbf{g})$. Since $\delta < (1 - \rho_1)/2$, by triangle inequality we must have $\Delta(\widetilde{G_0}, \widetilde{G_1}) < 1 - \rho_1$. As $\widetilde{G_0}$ and $\widetilde{G_1}$ are codewords (we identify polynomials with their respective encodings) in $\mathsf{RS}[\mathbb{G}, \mathsf{L}_1, \ell_p]$ which has minimum distance $1 - \rho_1$, we must have $\widetilde{G_0} = \widetilde{G_1}$. Thus $(\widetilde{f_0}, \widetilde{f_1})$ break the "binding" of Pedersen commitment scheme, and hence any PPT adversary $\mathcal{A}$ outputs them with negligible probability.

We now prove knowledge-soundness with random point evaluation. Informally, the outer IOPP Π_{outer} binds the prover to a unique multilinear polynomial $\widetilde{G}$ with coefficients $G_1, \ldots, G_p \in \mathbb{G}$. Subsequently, for each evaluation claim for $\mathbf{z} = (\mathbf{x}, \mathbf{y})$, the prover shows knowledge of opening for a linear combination of G_i's determined by $\mathbf{x}$, specifically for $C = \sum_{i=1}^{p} \widetilde{eq}(\mathbf{x}, \langle i \rangle) \cdot G_i$. These openings for several random $\mathbf{x}$ can be combined to obtain openings for each G_i, $i \in [t]$, which determine the "rows" of the polynomial $\widetilde{f}$. The formal proof follows.

Let $\mathcal{A}$ denote the adversary in the knowledge soundness game in Definition 3.3. Let

$$\big(G, \mathbf{z}, \sigma, \mathsf{tr} = (C, \ldots)\big)$$

be a successful transcript produced by $\mathcal{A}$[6]. In the above, we use tr to denote the transcript of the evaluation protocol $\Pi_{\mathbf{Eval}}([\![G]\!], \mathbf{z}, \sigma)$.

Call a transcript of $\Pi_{\mathbf{Eval}}$ *accepting* if both constituent interactions accept, i.e. $b_{\text{outer}} = \langle \mathcal{P}^*, \mathcal{V}_{\text{outer}}(\mathsf{pp}, \mathbf{x}, C, \delta, [\![G]\!])\rangle = 1$ and $b_{\text{inner}} = \langle \mathcal{P}^*, \mathcal{V}_{\text{inner}}(\mathsf{pp}, C, \mathbf{y}, \sigma)\rangle = 1$, which is exactly the acceptance condition $b_{\text{outer}} \wedge b_{\text{inner}}$ of $\mathcal{V}_{\mathbf{Eval}}$ in Figure 6.

[6]Note that entire oracle is considered part of transcript in an IOP. During BCS compilation, Merkle tree extractor can be used to extract consistent oracle

The extractor Ext repeatedly rewinds $\mathcal{A}$, on uniformly sampled points $\mathbf{z}_i = (\mathbf{x}_i, \mathbf{y}_i)$ to collect p accepting transcripts

$$\big(G, \mathbf{z}_i, \sigma_i, \mathsf{tr}_i = (C_i, \ldots)\big), i \in [p]$$

under the following *early abort* rule: Ext runs $\mathcal{A}$ once first and aborts immediately if that transcript is not accepting; otherwise Ext continues until it collects $p-1$ further accepting transcripts. Writing ϵ for $\mathcal{A}$'s success probability and T for the cost of one run of $\mathcal{A}$ together with the ensuing call to $\mathsf{Ext}_{\text{inner}}$, the expected running time is

$$(1-\epsilon)\cdot T + \epsilon \cdot \left(1 + \frac{p-1}{\epsilon}\right) \cdot T = p \cdot T,$$

Hence Ext runs in expected polynomial time for *every* ϵ (c.f. *Forking Lemma* [BCC+16, Lemma 1] for interactive protocols).

Call an accepting transcript a *bad accepting transcript* if the conditions of one of the two relations fail: either $(\mathsf{pp}, [\![G]\!], \mathbf{x}, C, \delta) \notin \mathcal{L}_{\text{outer}}$, or $\mathsf{Ext}_{\text{inner}}$ fails to return $\mathbf{a}$ with $\big((C, \mathbf{y}, \sigma); \mathbf{a}\big) \in \mathcal{R}_{\text{inner}}$.

Claim 5.1. *For a single run of $\mathcal{A}$ in the game of Definition 3.3,*

$$\Pr[\textit{the transcript is a bad accepting transcript}] \le \varepsilon_{\text{outer}} + \varepsilon_{\text{inner}}.$$

Proof of Claim 5.1. A bad accepting transcript lies in the union of

$$E_{\text{outer}} := \left[\left\langle \mathcal{P}^*, \mathcal{V}_{\text{outer}}(\mathsf{pp}, \mathbf{x}, C, \delta, [\![G]\!])\right\rangle = 1 \ \wedge \ (\mathsf{pp}, [\![G]\!], \mathbf{x}, C, \delta) \notin \mathcal{L}_{\text{outer}}\right]$$

and the analogous E_{inner} for the inner interaction, since badness requires one of the two relations to fail while the corresponding interaction accepts. These are exactly the events of Lemmas 5.3 and 5.4, with $\mathcal{A}$ in the role of $\mathcal{P}^*$.

For E_{outer}: the oracle $[\![G]\!]$ is fixed by $\mathcal{A}$ before the point is sampled, and C is sent by $\mathcal{A}$ in Step (1) of Figure 6, the statement $(\mathsf{pp}, [\![G]\!], \mathbf{x}, C, \delta)$ is therefore fixed before $\mathcal{V}_{\text{outer}}$ samples its randomness. Thus, by Lemma 5.3, $\Pr[E_{\text{outer}}] \le \varepsilon_{\text{outer}}$. Similarly, for E_{inner}: $(\mathsf{pp}, C, \mathbf{y}, \sigma)$ is fixed before the interaction of Step (3) begins, so Lemma 5.4 gives $\Pr[E_{\text{inner}}] \le \varepsilon_{\text{inner}}$. A union bound proves the claim. □

Claim 5.1 applies to any single run of $\mathcal{A}$, and we invoke it for $p+1$ runs: the p transcripts Ext collects, and the *challenge* transcript of the game, on the point $\mathbf{z}$ sampled by the game itself. A union bound over these $p+1$ runs gives

$$\Pr[\text{ some of the } p+1 \text{ transcripts is a bad accepting transcript }] \ \le \ (p+1)\cdot(\varepsilon_{\text{outer}} + \varepsilon_{\text{inner}}), \tag{12}$$

We condition the remainder of the analysis on the event that none of these $p+1$ transcripts are bad accepting transcripts. Later, we will add the residual probability in Equation 12 to obtain the final soundness error. Let $\mathbf{z}_i = (\mathbf{x}_i, \mathbf{y}_i)$ where $\mathbf{x}_i \in \mathbb{F}^{\ell_p}$, $\mathbf{y}_i \in \mathbb{F}^{\ell_q}$. Since tr_i contains an accepting transcript for $\Pi_{\text{inner}}(C_i, \mathbf{y}_i, \sigma_i)$, invoking extractor $\mathsf{Ext}^{\mathcal{A}}_{\text{inner}}$ for the protocol Π_{inner}, we obtain $(a_{i1}, \ldots, a_{iq})$ such that: $\sum_{j=1}^{q} a_{ij} \cdot g_j = C_i$ for $i \in [p]$. By the conditioning hypothesis, for all $i \in [p]$, we have $(\mathsf{pp}, [\![G]\!], \mathbf{x}_i, C_i, \delta) \in \mathcal{L}_{\text{outer}}$, where $\delta < (1-\rho_1)/2$. Since $\delta < (1-\rho_1)/2$, we see that the polynomials $\widetilde{G}_i$ witnessing the statements $(\mathsf{pp}, [\![G]\!], \mathbf{x}_i, C_i, \delta) \in \mathcal{L}_{\text{outer}}$ are identical, i.e, $\widetilde{G}_i = \widetilde{G}$ for the unique $\widetilde{G}$ satisfying $\Delta(G, \widetilde{G}) < \delta$. Thus, we have $C_i = \widetilde{G}(\mathbf{x}_i)$ for every i. The

extractor Ext sets up the following system of equations with unknowns G_i, $i \in [p]$:

$$\begin{bmatrix} \widetilde{eq}(\mathbf{x}_1, \langle 1 \rangle) & \widetilde{eq}(\mathbf{x}_1, \langle 2 \rangle) & \cdots & \widetilde{eq}(\mathbf{x}_1, \langle p \rangle) \\ \widetilde{eq}(\mathbf{x}_2, \langle 1 \rangle) & \widetilde{eq}(\mathbf{x}_2, \langle 2 \rangle) & \cdots & \widetilde{eq}(\mathbf{x}_2, \langle p \rangle) \\ \vdots & \vdots & \ddots & \vdots \\ \widetilde{eq}(\mathbf{x}_p, \langle 1 \rangle) & \widetilde{eq}(\mathbf{x}_p, \langle 2 \rangle) & \cdots & \widetilde{eq}(\mathbf{x}_p, \langle p \rangle) \end{bmatrix} \begin{bmatrix} G_1 \\ G_2 \\ \vdots \\ G_p \end{bmatrix} = \begin{bmatrix} C_1 \\ C_2 \\ \vdots \\ C_p \end{bmatrix}$$

$$= \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1q} \\ a_{21} & a_{22} & \cdots & a_{2q} \\ \vdots & \vdots & \ddots & \vdots \\ a_{p1} & a_{p2} & \cdots & a_{pq} \end{bmatrix} \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_q \end{bmatrix}$$

As shown in Lemma 5.5 (reproduced from [ZCF24]) the matrix consisting of evaluations $\widetilde{eq}(\mathbf{x}_i, \langle j \rangle)$ for uniformly sampled $\mathbf{x}_i$ is invertible with overwhelming probability. Thus, Ext can compute $p \times q$ matrix $\mathbf{F}$ such that $\mathbf{F} \cdot \mathbf{g} = \mathbf{G}$. The extractor outputs the unique multilinear polynomial $\widetilde{f}$ satisfying $\widetilde{f}(\langle i \rangle, \langle j \rangle) = F_{ij}$ for $i \in [p]$, $j \in [q]$. Note that by construction G_i commits to the ith row polynomial of $\widetilde{f}$. We now argue that $\widetilde{f}$ satisfies the relation $\mathcal{R}_{ev}$ whenever $\mathcal{A}$ succeeds in the evaluation protocol. Let Fail denote the event $b = 1 \wedge \mathcal{R}_{\mathbf{Eval}}(\mathsf{pp}, [\![G]\!], \mathbf{z}, \sigma; \widetilde{f}) = 0$ for $\widetilde{f}$ output by Ext, where $b = 1$ is the event that the challenge transcript is accepting. We bound $\Pr[\mathsf{Fail}]$ conditioned on the hypothesis that none of the $p + 1$ accepting transcripts is bad. The failure of $\widetilde{f}$ from being a witness is bound by the union of the following cases:

- G is δ-far from $\mathsf{RS}[\mathbb{G}, \mathsf{L}_1, \ell_p]$. This contradicts the conditioning hypothesis which asserts that $(\mathsf{pp}, [\![G]\!], \mathbf{x}_i, C_i, \delta) \in \mathcal{L}_{\text{outer}}$, and thus $\Delta(G, \widetilde{G}) < \delta$ for some $\widetilde{G}$.
- The polynomial $\widetilde{G}'$ obtained by solving the system differs from the unique codeword $\widetilde{G}$ δ-close to G. The conditioning gives $C_i = \widetilde{G}(\mathbf{x}_i)$ for that unique codeword at every $i \in [p]$, and the system has a unique solution whenever the $\widetilde{eq}$-matrix is invertible. Thus $\widetilde{G}' \neq \widetilde{G}$ is bound by the probability that the matrix is not invertible, which using Lemma 5.5 is at most $p \log p / |\mathbb{F}|$.
- The evaluation claim $\widetilde{f}(\mathbf{z}) = \sigma$ does not hold. Again, from the hypothesis that the main transcript is accepting and not bad, we have $C = \widetilde{G}(\mathbf{x})$ for the same polynomial $\widetilde{G}$ as above, and hence C is a Pedersen commitment to the polynomial $\widetilde{f}(\mathbf{x}, \mathbf{Y})$ by the construction of $\widetilde{f}$. The interaction being accepting and not bad, $\mathsf{Ext}_{\text{inner}}$ returns $\widetilde{a}$ satisfying $\mathsf{Com}(\widetilde{a}; \mathbf{g}) = C$ and $\widetilde{a}(\mathbf{y}) = \sigma$. If $\widetilde{f}(\mathbf{z}) \neq \sigma$ then $\widetilde{a}(\mathbf{Y}) \neq \widetilde{f}(\mathbf{x}, \mathbf{Y})$, as the two differ at $\mathbf{y}$. Since both have commitment C, this breaks the binding of Pedersen commitment and yields a nontrivial discrete-log relation among $\mathbf{g}$. This, this event occurs with probability at most ϵ_{DLOG}.

Adding the above probabilities,

$$\Pr[\ \mathsf{Fail} \ \wedge \ \text{none of the } p+1 \text{ transcripts is bad }] \ \leq \ \varepsilon_{\text{DLOG}} + \frac{p \log p}{|\mathbb{F}|}$$

Adding back the conditioning of (12), we have shown

$$\Pr[\mathsf{Fail}] \ \leq \ (p+1)\, \varepsilon_{\text{outer}} + (p+1)\, \varepsilon_{\text{inner}} + \varepsilon_{\text{DLOG}} + \frac{p \log p}{|\mathbb{F}|}$$

which is the bound in the statement. □

Remark 5.1 (On the non-interactive argument)**.** *Theorem 5.1 establishes that the evaluation IOP is knowledge sound, with an extractor that* rewinds *the adversary. The security of the*

non-interactive argument obtained by BCS compilation is therefore heuristic. Nevertheless, the compiled argument in the ROM can be shown to preserve the knowledge soundness of the IOP within a polynomial factor. This is done by compiling the two components separately *into non-interactive arguments and showing their sequential composition to be knowledge sound, rather than compiling the composed protocol in one step. We present this analysis in Appendix A.*

Lemma 5.5. *[ZCF24] For randomly chosen* $\mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_p} \leftarrow \mathbb{F}^p$ *the vectors* $(\widetilde{eq}(\langle j \rangle, \mathbf{x_1})_{j\in[p]}$, $(\widetilde{eq}(\langle j \rangle, \mathbf{x_2})_{j\in[p]}$, $\cdots$, $(\widetilde{eq}(\langle j \rangle, \mathbf{x_p})_{j\in[p]}$ *are linearly independent with overwhelming probability.*

Proof. Let us define the event E_i, where $E_i = 1$ if ith vector $(\widetilde{eq}(\langle j \rangle, \mathbf{x_i}))_{j\in[p]}$ is NOT in the span of the first $i-1$ vectors assuming all of the $i-1$ vectors are linearly independent. i.e $E_i = 1$ denotes that all the i vectors over the random choice of $x_1, \cdots, x_i$ are linearly independent . Then,

$$\begin{aligned}&\Pr[\text{all p vectors are linearly independent}]\\ &= \Pr[E_1 = 1] \cdot \Pr[E_2 = 1 | E_1 = 1] \cdots \Pr[E_p = 1 | E_{p-1} = 1]\end{aligned}$$

For any $i \in [1, p]$ and any $\mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_p} \in \mathbb{F}^p$ we will show that $\Pr[E_i = 1 | E_{i-1} = 1] \geq 1 - \frac{\log p}{|\mathbb{F}|}$ over the random choice of $x_i \leftarrow \mathbb{F}$. Let $M \in \mathbb{F}^{i-1 \times p}$ be the matrix consisting of the vectors $(\widetilde{eq}(\langle j \rangle, \mathbf{x_1})_{j\in[p]}, \cdots, (\widetilde{eq}(\langle j \rangle, \mathbf{x_{i-1}})_{j\in[p]}$ as rows. We get $M' \in \mathbb{F}^{i \times p}$ by adding $(\widetilde{eq}(\langle j \rangle, x_i)_{j\in[p]}$ as a row. Now $E_{i-1} = 1$ implies that all the $i-1$ vectors are linearly independent and rank of M is $i-1$. *If* $E_i = 1$ then rank of M' will be $i = \text{rank}(M) + 1$ which implies kernel$(M') \subsetneq$ kernel(M). So, we want to analyze the probability that there is one nonzero vector in kernel(M) which does not belong to kernel(M'). From the Rank Nullity theorem we know that $|\text{kernel}(M)| = |\mathbb{F}^{p-i}| > 1$. So we can pick a nonzero vector $G = (G_1, \cdots, G_p) \in \mathbb{G}^p \in$ kernel(M) and check the probability that G is also in kernel(M') i.e $M'.G = 0^i$ which is the same as the probability that $\langle (G_1, \cdots, G_p), (\widetilde{eq}(\langle j \rangle, x_i)_{j\in[p]} \rangle = G(x_i) = 0$. G being a $\log p$-variate multilinear polynomial, the probability that G is also in kernel(M') is $\frac{\log p}{|\mathbb{F}|}$ by Schwartz-Zippel lemma. So, the probability that rank of $M' = i + 1$ is at least $1 - \frac{\log p}{|\mathbb{F}|}$, which implies

$$\Pr[E_i = 1 | E_{i-1} = 1] = 1 - \frac{\log p}{|\mathbb{F}|}$$

So, the probability that all the p vectors over the random choice of the p points are linearly independent is

$$\left(1 - \frac{\log p}{|\mathbb{F}|}\right)^p \geq 1 - p \log p / |\mathbb{F}|$$

□

5.4 Extending to List Decoding Regime

We crucially relied on proximity parameter $\delta < (1 - \rho_1)/2$ to bind the prover to a unique polynomial $\widetilde{G}$ in the δ-neighborhood of G. For $(1 - \rho_1)/2 \leq \delta < 1 - \sqrt{\rho_1} - \gamma$ for a constant $\gamma > 0$, the δ-ball contains up to poly(p, γ^{-1}) polynomials in the δ-ball around G. The standard approach to use δ up to $1 - \sqrt{\rho_1}$ (Johnson's radius) is to request evaluation C_0 of the polynomial at a random point $z_0 \leftarrow \mathbb{F}$, called the *out of domain* sample. With overwhelming probability, the claimed evaluation at z_0 uniquely identifies the polynomial in the δ-neighborhood of G. To force the identified polynomial during evaluation, the evaluation claim $\widetilde{G}(\mathbf{x}) = C$ is appended with the evaluation claim $\widetilde{G}(\mathbf{z}_0) = C_0$. When the IOP is compiled to a non-interactive argument in ROM via BCS [BCS16] transformation, the point z_0 can be deterministically obtained by querying the *random oracle* RO with the Merkle tree commitment of G, i.e, $z_0 = \mathsf{RO}(\mathsf{rt})$, where $\mathsf{rt} = \mathsf{MerkleTree}^{\mathsf{RO}}(G)$. The evaluation $C_0 = \widetilde{G}(\mathbf{z}_0)$ is then included part of the commitment for $\widetilde{f}$.

5.5 Summarizing Costs of Titan PCS

We summarize the costs of the Titan PCS described above. For parameters $p, q, \rho_1, \rho_2, \ell_1, \ell_2, m_1, m_2, t_1$ and t_2 (see Figure 3), such that $n = p \cdot q$, $\ell_1 \cdot m_1 = p$, $\ell_2 \cdot m_2 = q$, the costs are as follows:

- **Commit**: The cost of committing to $\widetilde{f}$ is dominated by p MSMs of size q each to compute $\widetilde{G}$. Other costs include encoding $\widetilde{G}$ as a codeword and hashing it as a Merkle tree, which are $\widetilde{O}(p) = o(n)$.
- **Eval** Prover: Discarding the polynomial evaluation, the prover incurs MSM of size p to compute partial evaluation C, group sumcheck to prove outer evaluation $\widetilde{G}(\mathbf{x}) = C$ costs $\sqrt{p}$ MSMs of size $\sqrt{p}$. Finally the CSP costs k_2 MSMs of size q. Thus the total prover cost is $\approx \mathrm{MSM}(p) + \sqrt{p}\mathrm{MSM}(\sqrt{p}) + k_2 \cdot \mathrm{MSM}(q) + O(p+q)\,\mathbb{F}$.
- **Eval** Verifier: The verifier incurs cost $\mathrm{MSM}(t_1\ell_1 + m_1) + t_1(m_1 + \ell_1)\,\mathbb{F}$ during consistency checks in Π_{outer}, and $\mathrm{MSM}(t_2\ell_2 + m_2) + t_2(m_2 + \ell_2)\,\mathbb{F}$ in Π_{inner}.
- **Eval** Proof Size: We calculate proof size as: From Lemma 5.1, communication in Π_{outer} is $t_1\ell_1 + m_1 + 3k_1\ \mathbb{G}$. The communication during Πinner is (i) $k_2(2\,\mathbb{G} + 2\,\mathbb{F})$ during CSP rounds and (ii) $m_2\,\mathbb{G} + m_2\,\mathbb{F}$ to communicate reduced instance. We do not count the openings of oracle $[\![g]\!]$ towards proof-size, as we assume that verifier stores it locally. Summarising,

$$|\pi| = (t_1\ell_1 + m_1 + m_2 + 3k_1 + 2k_2)\,\mathbb{G} + (m_2 + 2k_2)\,F$$

We describe parameter settings to obtain soundness of $2^{-\lambda}$. In the list-decoding regime, choosing $\delta \approx 1 - \sqrt{\rho_1}$, gives the soundness of $\varepsilon_{\mathrm{RPE}} \approx p \cdot \rho_1^{t_1/2} + 3^{k_2} \cdot \rho_2^{t_2}$. Thus, we set $t_1 \approx 2(\lambda + \log p)/\log(1/\rho_1)$ and $t_2 \approx (\lambda + 1.6k_2)/\log(1/\rho_2)$. Thus, $t_1, t_2 = O(\lambda)$. We also set $p, q = O(\sqrt{n})$, $\ell_1 = \sqrt{p/t_1}$, $m_1 = \sqrt{p \cdot t_1}$, $\ell_2 = \sqrt{q/t_2}$ and $m_2 = \sqrt{q \cdot t_2}$. With these choices, we obtain verifier complexity of $O(\lambda^{3/2} \cdot n^{1/4})$ field operations and $O(\lambda n^{1/4})$ MSMs. The proof size is $O(\lambda n^{1/4})$ group elements. In Section 7, we explicitly specify parameters for each choice of m.

6 Optimizations for IOPP over Groups

In this section we describe optimizations for efficiently realizing IOPP over groups, as highlighted in Section 2. Throughout, let$n$ denote the size of multilinear group polynomial $\widetilde{g} \in \mathbb{G}^{\leq 1}[X_1, \ldots, X_{\log n}]$. When used within Titan, n will be $O(\sqrt{N})$, where N is the size of the field polynomial.

6.1 Encoding Group Oracle

Given a polynomial $\widetilde{g} \in \mathbb{G}^{\leq 1}[X_1, \ldots, X_m]$ in evaluation basis, with evaluations $g_1, \ldots, g_n$ over the hypercube, encoding g as an oracle requires evaluations $\widehat{g}(x)$ for $x \in \mathsf{L}$. We avoid realizing $\widehat{g}$ explicitly, and instead rely on the following recursion:

$$\widetilde{g}(x, x^2, \ldots, x^{2^{m-1}}) = (1 - x) \cdot \widetilde{g}(0, x^2, \ldots, x^{2^{m-1}}) + x \cdot \widetilde{g}(1, x^2, \ldots, x^{2^{m-1}})$$

Thus, evaluations of $\widetilde{g}$ over L can be constructed in a "butterfly" like fashion from evaluations of $\widetilde{g}(0, \cdot)$ and $\widetilde{g}(1, \cdot)$ over domain $\mathsf{L}^{(2)}$. The above recursion allows us to directly use evaluations $g_1, \ldots, g_n$ at the final layer of the procedure. While we incur $\frac{n}{2} \log n$ scalar multiplications, we save on $\approx n \log n$ scalar multiplication required to compute $\widehat{g}(X)$ explicitly.

6.2 Efficient Group Sumcheck

Naive sumcheck over groups incurs $O(n)$ group-exponentiations for the prover. Here, we describe a variant using costs roughly equivalent of MSM of size $O(n)$, which in practice is substantially faster (optimized Pippenger's based MSM can be 20× to 50× faster). Let $m = \log n$ and $\widetilde{f}$ be an m-variate polynomial which is MLE of the function $f : \{0,1\}^m \to \mathbb{G}$, we consider sumcheck instance of the form:

$$\sum_{\mathbf{x} \in \{0,1\}^m} \widetilde{eq}(\boldsymbol{\alpha}, \mathbf{x}) \cdot \widetilde{f}(\mathbf{x}) = \sigma \tag{13}$$

The sumcheck protocol requires the prover to send polynomials $g_i(X)$ in round i for $i \in [m]$, where the polynomial $g_i(X)$ is defined by:

$$g_i(X) := \sum_{\mathbf{x} \in \{0,1\}^{m-i}} \widetilde{eq}(\boldsymbol{\alpha}; \boldsymbol{\rho}_i, X, \mathbf{x}) \cdot \widetilde{f}(\boldsymbol{\rho}_i, X, \mathbf{x}) \tag{14}$$

where $\boldsymbol{\rho}_i$ denotes the vector of challenges $(r_1, \ldots, r_{i-1})$ sent by the verifier in previous $i-1$ rounds. Note that $\boldsymbol{\rho}_1$ is an empty vector. To interpolate polynomial g_i, it suffices to compute $g_i(u)$ for three distinct values of u. We discuss computation of $g_i(u)$ for $u \in \mathbb{F}$ next. We have:

$$g_i(u) := \sum_{\mathbf{x} \in \{0,1\}^{m-i}} \widetilde{eq}(\boldsymbol{\alpha}, \boldsymbol{\rho}_i, u, \mathbf{x}) \cdot \widetilde{f}(\boldsymbol{\rho}_i, u, \mathbf{x}) \tag{15}$$

Folklore methods incur $O(2^m) = O(n)$ cost to compute $g_i(u)$ for all $i \in [m]$. We start by defining $\widetilde{h}(\mathbf{x}) = \widetilde{eq}(\boldsymbol{\alpha}, \mathbf{x}) \cdot \widetilde{f}(\mathbf{x})$. Let $\ell = m/2$. As a first step, the prover computes the partial sums:

$$S_i(\mathbf{b}) = \sum_{\mathbf{x} \in \{0,1\}^{m-i}} \widetilde{h}(\mathbf{b}, \mathbf{x}) \tag{16}$$

for all $1 \le i \le \ell$ where S_i is computed for all $\mathbf{b} \in \{0,1\}^i$. Note that computing S_i for given $\mathbf{b} \in \{0,1\}^i$ incurs MSM of size $O(2^{m-i})$. We compute the above sums as follows:

1. First compute $S_\ell(\mathbf{b})$ for all $\mathbf{b} \in \{0,1\}^\ell$. This incurs total 2^ℓ MSMs of size 2^ℓ each.

2. Next, we notice that $S_{i-1}(\mathbf{b}) = S_i(\mathbf{b}, 0) + S_i(\mathbf{b}, 1)$ and thus the table S_{i-1} can be computed from S_i in 2^i $\mathbb{G}$ additions.

3. Proceeding, the tables S_i for all $1 \le i \le \ell$ can be computed in 2^ℓ MSMs of size 2^ℓ and $O(2^\ell) = O(\sqrt{n})$ additions in $\mathbb{G}$.

4. As we show in section 6.2.1, round messages $g_i(u)$, $u \in \{0,1,2\}$ for the ith round for $i < \ell$ can be computed using MSM of size $O(2^i)$ over table S_i as:

$$g_i(u) := \sum_{\mathbf{b} \in \{0,1\}^i} \frac{\widetilde{eq}(\mathbf{z}, \mathbf{b}) \cdot \widetilde{eq}(\mathbf{z}, \boldsymbol{\alpha}_i)}{\widetilde{eq}(\boldsymbol{\alpha}_i, \mathbf{b})} \cdot S_i(\mathbf{b})$$

   In the above $\boldsymbol{\alpha}_i$ denotes the prefix $(\alpha_1, \ldots, \alpha_i)$ of $\boldsymbol{\alpha}$.

5. For rounds $i \ge \ell$, the size of the polynomial reduces to $O(\sqrt{n})$, and hence the prover computes round messages in usual manner.

As it can be seen, the above scheme reduces group-exponentiation to only $O(\sqrt{n})$, while accomplishing remaining work over groups using MSMs.

6.2.1 Computing round messages in group sumcheck.

We now consider cost of generating sumcheck messages for each round for the sumcheck protocol in Section 6.2. For round $i < \ell$, we have: (Let $\mathbf{z} = (r_1, \ldots, r_{i-i}, u)$ denote the vector containing u appended to challenges sent).

$$\begin{aligned}
g_i(u) &= \sum_{\mathbf{x}\in\{0,1\}^{m-i}} \widetilde{eq}(\alpha, \mathbf{z}, \mathbf{x}) \cdot \widetilde{f}(\mathbf{z}, \mathbf{x}) \\
&= \sum_{\mathbf{x}\in\{0,1\}^{m-i}} \widetilde{eq}(\alpha, \mathbf{z}, \mathbf{x}) \sum_{\mathbf{b}\in\{0,1\}^{i}} \widetilde{eq}(\mathbf{z}, \mathbf{b}) \cdot \widetilde{f}(\mathbf{b}, \mathbf{x}) \\
&= \sum_{\mathbf{b}\in\{0,1\}^{i}} \widetilde{eq}(\mathbf{z}, \mathbf{b}) \sum_{\mathbf{x}\in\{0,1\}^{m-i}} \widetilde{eq}(\alpha, \mathbf{z}, \mathbf{x}) \widetilde{f}(\mathbf{b}, \mathbf{x}) \\
&= \sum_{\mathbf{b}\in\{0,1\}^{i}} \widetilde{eq}(\mathbf{z}, \mathbf{b}) \sum_{\mathbf{x}\in\{0,1\}^{m-i}} \widetilde{eq}(\alpha, \mathbf{b}, \mathbf{x}) \left(\frac{\widetilde{eq}(\alpha, \mathbf{z}, \mathbf{x})}{\widetilde{eq}(\alpha, \mathbf{b}, \mathbf{x})}\right) \widetilde{f}(\mathbf{b}, \mathbf{x}) \\
&= \sum_{\mathbf{b}\in\{0,1\}^{i}} \frac{\widetilde{eq}(\mathbf{z}, \mathbf{b}) \cdot \widetilde{eq}(\mathbf{z}, \alpha_i)}{\widetilde{eq}(\alpha_i, \mathbf{b})} \cdot S_i(\mathbf{b}) \qquad (17)
\end{aligned}$$

In the above, we have $\alpha_i = \alpha[1 : i]$ and we use the fact that:

$$\frac{\widetilde{eq}(\alpha, \mathbf{z}, \mathbf{x})}{\widetilde{eq}(\alpha, \mathbf{b}, \mathbf{x})} = \frac{\widetilde{eq}(\alpha_i, \mathbf{z})}{\widetilde{eq}(\alpha_i, \mathbf{b})} \qquad (18)$$

Given the precomputed values of $S_i(\mathbf{b})$, the above can be computed using MSM of size $O(2^i)$. Thus, the total cost to compute first ℓ round messages is $O(2^\ell) = O(\sqrt{n})$ MSM. The final ℓ messages can be computed by usual techniques, as the size of the polynomial after fixing the first ℓ challenges is $O(\sqrt{n})$ and so it incurs $O(\sqrt{n})$ group exponentiations. The total complexity of a sum-check for group polynomial is therefore $\sqrt{n}\,\mathrm{MSM}(\sqrt{n}) + O(\sqrt{n})$ group-exp.

6.3 Speeding up CSP prover

We reduce costs of CSP prover by using MSMs to speed up costly initial iterations. In more detail, CSP prover starts with initial commitment key $\mathbf{g}_0 = \mathbf{g}$ and witness $\mathbf{a}_0 = \mathbf{a}$ in $\mathbb{G}^n$ and $\mathbb{F}^n$ respectively. In our variant of the CSP, the commitment key $\mathbf{g}_i$ and the witness $\mathbf{a}_i$ in the ith round is given by evaluations of polynomials $\widetilde{g}(c_1, \ldots, c_{i-1}, \mathbf{x})$ and $\widetilde{a}(1 - c_1, \ldots, 1 - c_{i-1}, \mathbf{x})$ where $\widetilde{g}$ and $\widetilde{a}$ are polynomials interpolating vectors $\mathbf{g}$ and $\mathbf{a}$ and $c_1, \ldots, c_{i-1}$ are challenges so far. Let $\mathbf{c}_i = (c_1, \ldots, c_{i-1})$ and $\mathbf{c}'_i = (1 - c_1, \ldots, 1 - c_{i-1})$. In terms of these polynomials, round messages L_i and R_i, cross term L_i can be computed via 2^{i-1} MSMs, each of size 2^{m-i} (See the below paragraph on Computing Round Messages in CSP). For smaller values of i, the above computation is faster than folding the key $\mathbf{g}_i$ using 2^i group-exponentiations. In our implementations, we use MSMs till we terminate the CSP at approximately $m/2$ rounds.

6.3.1 Computing Round Messages in CSP.

We show how round message L_i (a cross-commitment) in a CSP can be expressed in terms of group polynomials. This interpretation allows it to be computed via 2^{i-1} MSMs, each of size

2^{m-i}:

$$\begin{aligned} L_i &= \sum_{\mathbf{x}\in\{0,1\}^{m-i}} \widetilde{a}(\mathbf{c}'_i, 1, \mathbf{x}) \cdot \widetilde{g}(\mathbf{c}_i, 0, \mathbf{x}) \\ &= \sum_{\mathbf{x}\in\{0,1\}^{m-i}} \widetilde{a}(\mathbf{c}'_i, 1, \mathbf{x}) \sum_{\mathbf{b}\in\{0,1\}^{i-1}} \widetilde{eq}(\mathbf{c}_i, \mathbf{b}) \cdot \widetilde{g}(\mathbf{b}, 0, \mathbf{x}) \\ &= \sum_{\mathbf{b}\in\{0,1\}^{i-1}} \widetilde{eq}(\mathbf{c}_i, \mathbf{b}) \sum_{\mathbf{x}\in\{0,1\}^{m-i}} \widetilde{a}(\mathbf{c}'_i, 1, \mathbf{x}) \cdot \widetilde{g}(\mathbf{b}, 0, \mathbf{x}) \end{aligned}$$

7 Implementation

We implement Titan PCS and the SNARK using Spartan and Titan in Rust[7] using Pallas curve from Pasta pair of curves. The curve features a 256-bit scalar field.

Experimental Setup: All experiments were conducted on a machine running Ubuntu 24.04.3 LTS with Linux kernel 6.8.0. The system is equipped with a single AMD EPYC 9754 processor with a maximum clock frequency of 2.25 GHz, and 251 GB of RAM. To ensure fair comparison, all benchmarks were executed in a single-threaded manner by setting `RAYON_NUM_THREADS=1`, disabling parallel test execution, and pinning the process to a fixed CPU core (core 0) using `taskset`.

Comparison and Metrics. We compare the Titan commitment scheme against several transparent polynomial commitment schemes: Brakedown [GLS+23], Dory [Lee21], Hyrax [WTs+18], Kopis [SL20], and WHIR [ACFY25]. We report the following metrics: (i) commitment time; (ii) evaluation time; (iii) verification time; and (iv) proof size. We note that Brakedown and WHIR are plausibly post-quantum safe, while remaining schemes, including ours rely on classical assumptions. We include them to illustrate our savings with respect to hash based schemes – at the cost of sacrificing post-quantum security.

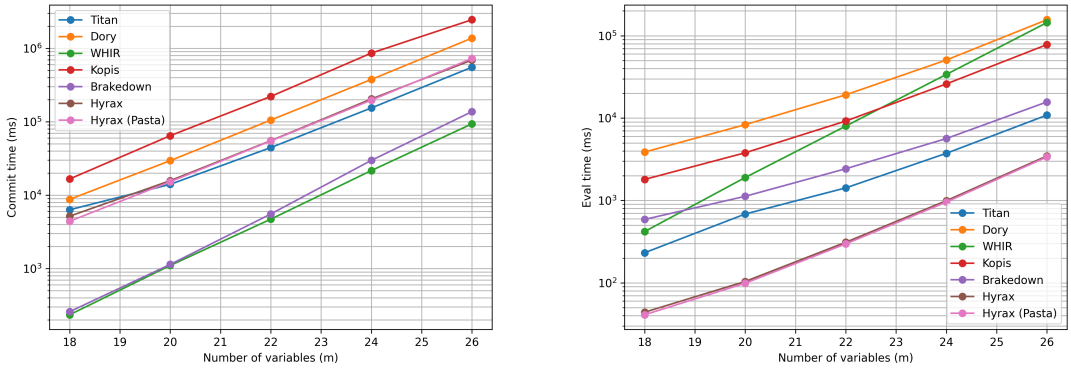
Dory and Kopis are instantiated using the `quarks-zk` rust crate [Qua]. For Brakedown and Hyrax we use the implementations available via Arkworks `poly-commit` library [Ark], with custom benchmark modules added to measure commitment, evaluation, and verification times. WHIR is evaluated using the available implementation [Wiz], which we instrument to separately report commitment and evaluation time. Titan is instantiated over the Pasta curve using the Pallas scalar field (256-bit) with group operations implemented via the Pallas backend. Brakedown, Dory and Kopis are all instantiated over the scalar field of BLS12-381 curve, while Hyrax is instantiated over the Pasta curve and WHIR is evaluated over a 256-bit prime field supported by its implementation.

Titan Parameters: For Titan PCS, we use $\rho_1 = \rho_2 = 1/16$ as the rates for Reed-Solomon codes for encoding the outer group polynomial and the Pedersen generators respectively. The number of outer and inner queries are set to $t_1 = 70$ and $t_2 = 35$ to achieve ≈ 128 bits of soundness. We implement Merkle path compression techniques to reduce communication overhead. The parameters for different polynomial sizes are reported in the table 1. We skew the matrix towards q (the size of generators), as querying oracle $[\![g]\!]$ has better soundness (we know it encodes a polynomial). For encoding the outer group polynomial, we roughly skew the size of folded polynomial and the size of cosets as $\sqrt{t_1 p}$ and $\sqrt{p/t_1}$ which leads to optimal proof size and verifier computation.

[7] https://github.com/ravi-prakash02/titan-implementation

Table 1: Parameter settings for TITAN for different polynomial sizes. p and q denote sizes of outer and inner group polynomials. m_1 and m_2 denote size of folded outer and inner polynomials. All sizes are reported in log scale.

m	p	q	ℓ_1	m_1	ℓ_2	m_2
18	8	10	1	7	5	5
20	8	12	1	7	6	6
22	9	13	2	7	6	7
24	10	14	2	8	7	7
26	11	15	3	8	8	7



(a) Commitment time

(b) Evaluation time

Figure 8: Performance comparison: Commitment and Evaluation Times

7.1 Evaluation Summary

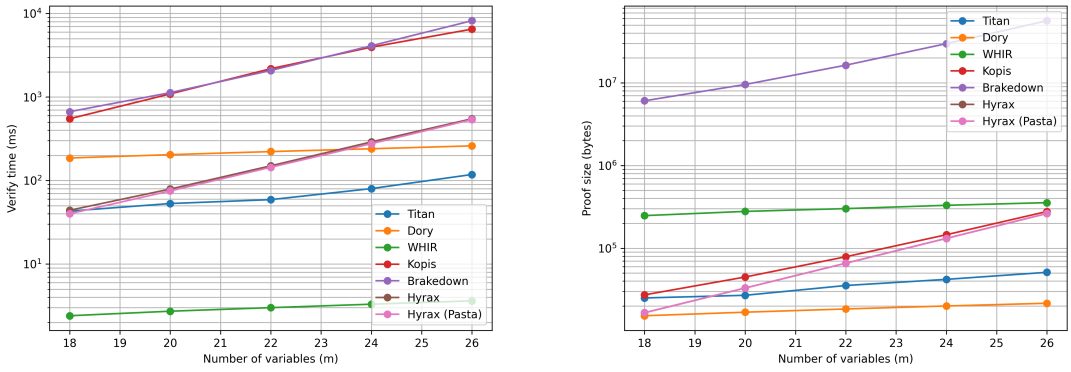
In Table 2 we present the commit time, evaluation time, verification time and proof size of Brakedown, Dory, Hyrax, Kopis, Titan, and WHIR grouped by the number of variables $m \in \{18, 20, 22, 24, 26\}$. The lowest times for commit, eval, verify and the lowest proof size for each m are represented in teal. Additionally, we provide a visual representation of these metrics in Figures 8a, 8b, 9a, and 9b. These figures compare the commitment time, evaluation time, verification time, and proof size, respectively across all schemes as the number of variables m increases from 18 to 26. From the evaluation, we can see that Titan PCS meets the desiderata we started with: much smaller proof sizes than hash based schemes, while superior prover and verification performance compared to group based schemes, without sacrificing proof size significantly. While commitment and evaluation in Titan are considerably faster than Dory, the superior asymptotics of Dory verifier start catching up for larger polynomials. We provide additional break up of Titan proof size into three components: outer evaluation (IOPP), inner evaluation (CSP), and Merkle paths. Table 3 reports their sizes (in KiB) as a function of the number of variables m.

7.2 SNARK using Spartan and Titan

We describe a SNARK, based on Microspartan [ZSC24] PIOP (variant of the Spartan PIOP for R1CS from [Set20]), slightly redesigned for Titan PCS. Our SNARK quadratically improves proof size and verification complexity of the original scheme from [Set20] (the transparent variant Spartan$_{\text{DL}}$) from $O(\sqrt{n})$ to $O(\sqrt[4]{n})$. Concretely, we show below that the proof sizes we obtain are about 3× smaller at $m = 23$.

Table 2: Comparison of polynomial commitment schemes grouped by number of variables m.

m	Scheme	Commit (s)	Eval (s)	Verify (s)	Size (KiB)
18	Brakedown	0.26	0.59	0.667	5921.9
	Dory	8.75	3.87	0.186	14.9
	Hyrax	4.41	0.04	0.04	16.2
	Kopis	16.58	1.802	0.549	26.6
	Titan	**6.31**	**0.23**	**0.043**	**24.4**
	WHIR	0.23	0.419	0.002	241.8
20	Brakedown	1.13	1.12	1.128	9325.9
	Dory	29.45	8.32	0.204	16.4
	Hyrax	15.19	0.10	0.075	32.2
	Kopis	64.42	3.79	1.089	43.7
	Titan	**14.02**	**0.69**	**0.053**	**26.3**
	WHIR	1.10	1.90	0.003	271.8
22	Brakedown	5.53	2.43	2.078	15961.4
	Dory	105.25	19.18	0.222	18.0
	Hyrax	54.69	0.30	0.144	64.2
	Kopis	220.92	9.21	2.182	76.8
	Titan	**44.38**	**1.42**	**0.059**	**34.5**
	WHIR	4.70	8.00	0.003	293.8
24	Brakedown	29.73	5.65	4.112	29059.8
	Dory	376.25	50.73	0.240	19.5
	Hyrax	196.63	0.96	0.276	128.2
	Kopis	860.33	26.00	3.957	141.9
	Titan	**153.92**	**3.74**	**0.080**	**40.9**
	WHIR	21.60	33.90	0.003	322.2
26	Brakedown	137.01	15.71	8.198	55084.3
	Dory	1374.66	156.46	0.260	21.0
	Hyrax	727.20	3.38	0.537	256.2
	Kopis	2449.69	78.00	6.500	271.1
	Titan	**552.11**	**10.87**	**0.118**	**49.8**
	WHIR	93.50	144.38	0.004	346.8



(a) Verification time

(b) Proof size

Figure 9: Performance comparison: Verification Time and Proof Size

In particular, we reduce size of auxiliary polynomial oracles at the cost of slightly increased sum-check communication. We provide detailed PIOP in Section 7.3. Here, we give an

Table 3: Break-up of Titan Proof Size

m	Outer Eval (KiB)	Inner Eval (KiB)	Merkle Paths (KiB)
18	8.43	2.63	13.33
20	8.43	4.75	13.06
22	12.5	8.75	13.24
24	17.06	9.0	15.16
26	24.25	9.25	15.31

overview of how Titan PCS is involved in the compilation of PIOP into a SNARK.

For $m, n \in \mathbb{N}$ and field $\mathbb{F}$, the relation $\mathcal{R}1\mathcal{CS}_{m,n,\mathbb{F}}$ consists of tuples $(A, B, C; \mathbf{z})$ with $A, B, C \in \mathbb{F}^{m \times n}$ and $\mathbf{z} \in \mathbb{F}^n$ satisfying $A\mathbf{z} \circ B\mathbf{z} = C\mathbf{z}$[8]. We consider A, B and C to be sparse matrices with $O(m + n)$ non-zero entries. Without loss of generality, we can assume $m = n$, which can be ensured by zero-padding vectors and matrices. We describe our variant of Spartan PIOP in Section 7.3, which closely follows presentation in [Set20]. The PIOP in Section 7.3 requires $\mathcal{P}$ to provide following evaluations for $\log N$-variate polynomials (where N is the total number of non-zero entries in A, B and C combined).

- Evaluations of index polynomials (trusted) $\widetilde{row}, \widetilde{col}, \widetilde{val}_A, \widetilde{val}_B$ and $\widetilde{val}_C$ at $\mathbf{r} \in \mathbb{F}^{\log N}$.
- Evaluations of witness polynomial $\widetilde{z}$ at $\mathbf{r}_y \in \mathbb{F}^{\log N}$.
- Evaluations of second round polynomials $\widetilde{p}, \widetilde{q}, \widetilde{m}_r, \widetilde{m}_c$ at $\mathbf{r} \in \mathbb{F}^{\log N}$.
- Evaluation of third round polynomial $\widetilde{g}$ at $\mathbf{r}$.

In the above, polynomials $\widetilde{z}$, $\widetilde{m}_r$ and $\widetilde{m}_c$ have "short" coefficients. We implement following standard optimisations to reduce proof size and verification complexity.

Oracle Aggregation: We aggregate four second round polynomials into a $(\log N + 2)$-variate polynomial $\widetilde{w}$. Then queries to constituent polynomials at $\mathbf{r}$ reduce to queries to $\widetilde{w}$ at $(\mathbf{r}, 0, 0), \ldots, (\mathbf{r}, 1, 1)$. We use the standard trick of reducing distinct queries to $\widetilde{w}$ to a single query to $\widetilde{w}$ at $\mathbf{r}' \in \mathbb{F}^{\log N+2}$ via a degree-2 sumcheck [CBBZ23]. The prover thus sends oracle $[\![\widetilde{w}]\!]$ to the aggregate polynomial $\widetilde{w}$.

Oracle Batching: In this case, we batch the claims $\widetilde{f}_i(\mathbf{r}) = v_i$, $i \in [k]$ using a random linear combination to the claim $\sum_{i=1}^{k} \gamma^{i-1} \widetilde{f}_i(\mathbf{r}) = \sum_{i=1}^{k} \gamma^{i-1} v_i$. We simulate queries to $[\![\sum_{i=1}^{k} \gamma^{i-1} \widetilde{f}_i]\!]$ by querying each constituent oracle. Note that instead of sending partially evaluated polynomials $\widetilde{a}_i(\mathbf{Y}) = \widetilde{f}_i(\mathbf{r}_x, \mathbf{Y})$ for each $i \in [k]$, $\mathcal{P}$ can simply send the batched evaluation $\sum_{i=1}^{k} \gamma^{i-1} \widetilde{a}_i(\mathbf{Y})$. This allows us to use just one instance of the CSP for evaluation of the partial polynomial. We use oracle batching for index polynomials $\widetilde{row}, \widetilde{col}, \widetilde{val}_A, \widetilde{val}_B$ and $\widetilde{val}_C$. We also note that querying trusted polynomials incurs less queries, as soundness for each query is $\approx \rho$ (the rate of the RS encoding). We also batch oracles $[\![\widetilde{z}]\!]$ and $[\![\widetilde{g}]\!]$. Since these are queried on distinct points, we first invoke a degree-2 sumcheck to reduce the queries to the same point.

In Table 4, we evaluate the performance of the SNARK alongwith Spartan$_{\text{DL}}$ [Set20]. For Spartan$_{\text{DL}}$, benchmarks we use the available implementation at [Mic]. We use identical method of generating synthetic R1CS matrices, as in the reference implementation [Mic]. We note that the implementation [Mic] implements a zero-knowledge variant which incurs additional prover overheads. The implementations also use different curves (Ristretto vs Pasta), so the comparison (specifically of prover time) is not entirely apples-to-apples. However, for larger polynomial sizes, the asymptotic advantage of underlying PCS Titan ($\sqrt[4]{n}$ vs $\sqrt{n}$) in

[8]It is customary to consider $\mathbf{z} = (1, \mathbf{x}, \mathbf{w})$ with $\mathbf{x}$ denoting public inputs. We ignore this decomposition for simplicity

verification and proof size starts to manifest itself. For SNARK using Titan, the most expensive operation is committing to auxiliary polynomials with large coefficients, followed by the computing sumcheck messages and evaluation proofs. The proof sizes are relatively short in our SNARK, as group oracles for index polynomials can be locally stored by the verifier (being $O(\sqrt{n})$ in size), and thus queries to them do not incur communication. The overhead of evaluations of index polynomials is also very low in the SNARK using Titan, being $O(\sqrt{n})$, largely using MSMs.

Table 4: Performance comparison of Spartan variants for R1CS instances of size $N = 2^{20}, 2^{21}, 2^{22}$ and 2^{23}. $*$ The implementation of $\text{Spartan}_{\text{DL}}$ is zero-knowledge, while our SNARK is not.

N	Scheme	Prover* (s)	Verifier (ms)	Proof Size (KB)
2^{20}	$\text{Spartan}_{\text{DL}}$	32.5	93	180
	Spartan+Titan	39	105	93
2^{21}	$\text{Spartan}_{\text{DL}}$	67	121	252
	Spartan+Titan	68	118	102
2^{22}	$\text{Spartan}_{\text{DL}}$	124	173	290
	Spartan+Titan	138	124	120
2^{23}	$\text{Spartan}_{\text{DL}}$	245	230	429
	Spartan+Titan	240	139	137

7.3 SNARK using Spartan and Titan:Details

Let A, B and C be $m \times n$ matrices over the field $\mathbb{F}$ defining the R1CS relation as in Section 7.2. Formally, let n_A, n_B and n_C be number of non-zero entries in matrices A, B and C respectively. Let $N = n_A + n_B + n_C$ and let M be the matrix obtained by row-wise concatenation of A, B and C. Let $\{(\mathsf{row}_i, \mathsf{col}_i, \mathsf{val}_i)\}_{i=1}^{N}$ be a canonical ordering of the non-zero tuples in the sparse representation of M, where we assume that first n_A tuples come from A, next n_B come from B and the remaining n_C come from C. Without loss of generality, we zero-pad matrices A, B and C to size $N \times N$. We define vectors $\mathbf{row}, \mathbf{col}, \mathbf{val}_A, \mathbf{val}_B, \mathbf{val}_C \in \mathbb{F}^N$ as follows:

- For $i = 1, \ldots, N$: $\mathbf{row}[\,i\,] = \mathsf{row}_i$, $\mathbf{col}[\,i\,] = \mathsf{col}_i$.
- For $i = 1, \ldots, n_A$: $\mathbf{val}_A[\,i\,] = \mathsf{val}_i$.
- For $i = 1, \ldots, n_B$: $\mathbf{val}_B[\,i\,] = \mathsf{val}_{i+n_A}$.
- For $i = 1, \ldots, n_C$: $\mathbf{val}_C[\,i\,] = \mathsf{val}_{i+n_A+n_B}$.

For all the above vectors, the positions not explicitly specified are assumed to be 0. Let $\widetilde{row}, \widetilde{col}$, $\widetilde{val}_A$, $\widetilde{val}_B$ and $\widetilde{val}_C$ denote the MLEs of above vectors. We note that for each $K \in \{A, B, C\}$, the MLE $\widetilde{K}$ of K is given by:

$$\begin{aligned}\widetilde{K}(\mathbf{x}, \mathbf{y}) &= \sum_{i=1}^{N} \sum_{j=1}^{N} K(i, j) \cdot \widetilde{eq}(\mathbf{x}, \langle i \rangle) \cdot \widetilde{eq}(\mathbf{y}, \langle j \rangle) \\ &= \sum_{i=1}^{N} \mathsf{val}_K[\,i\,] \cdot \widetilde{eq}(\mathbf{x}, \mathbf{row}[\,i\,]) \cdot \widetilde{eq}(\mathbf{y}, \mathbf{col}[\,j\,]) \end{aligned} \tag{19}$$

In the above, we identify $x \in \mathbb{F}$ with $\langle x \rangle$ to simplify notation. Let $\mathbf{z} \in \mathbb{F}^n$ be R1CS witness satisfying $A\mathbf{z} \circ B\mathbf{z} = C\mathbf{z}$. As outlined in [Set20, Section 5], the check $A\mathbf{z} \circ B\mathbf{z} = C\mathbf{z}$ reduces via the application of sum-check protocol to the oracle queries $\widetilde{A}(\mathbf{r}_x, \mathbf{r}_y)$, $\widetilde{B}(\mathbf{r}_x, \mathbf{r}_y)$, $\widetilde{C}(\mathbf{r}_x, \mathbf{r}_y)$ and $\widetilde{z}(\mathbf{r}_y)$ for some $\mathbf{r}_x, \mathbf{r}_y \in \mathbb{F}^{\log N}$. Let $\mathbf{T}_x$ and $\mathbf{T}_y$ denote tables of size N given by $\mathbf{T}_x[\,i\,] = \widetilde{eq}(\mathbf{r}_x, \langle i \rangle)$ and $\mathbf{T}_y[\,i\,] = \widetilde{eq}(\mathbf{r}_y, \langle i \rangle)$ for $i \in [N]$. Using, Equation (19), we can write:

$$\widetilde{K}(\mathbf{r}_x, \mathbf{r}_y) = \sum_{\mathbf{x} \in B_{\log N}} \widetilde{val}_K(\mathbf{x}) \cdot \widetilde{p}(\mathbf{x}) \cdot \widetilde{q}(\mathbf{x}) \text{ for } K \in \{A, B, C\} \tag{20}$$

where $\widetilde{p}$ and $\widetilde{q}$ are multilinear polynomials defined by $\widetilde{p}(\mathbf{x}) = \mathbf{T}_x[\,\widetilde{row}(\mathbf{x})\,]$ and $\widetilde{q}(\langle i \rangle) = \mathbf{T}_y[\,\widetilde{col}(\mathbf{x})\,]$. The correctness of lookups to compute polynomials $\widetilde{p}$ and $\widetilde{q}$ can be reduced to a sum-check involving auxiliary polynomial oracles corresponding to multiplicities of elements of the tables in the target vector. Using the approach of [Hab22], the prover sends polynomial oracles $[\![\widetilde{m}_r]\!]$, $[\![\widetilde{m}_c]\!]$ and shows the equality of rational functions:

$$\sum_{\mathbf{x} \in B_{\log N}} \frac{\widetilde{m}_r(\mathbf{x})}{X + \widetilde{id}(\mathbf{x}) \cdot Y + \widetilde{eq}(\mathbf{x}, \mathbf{r}_x)} = \sum_{\mathbf{x} \in B_{\log N}} \frac{1}{X + \widetilde{row}(\mathbf{x}) \cdot Y + \widetilde{p}(\mathbf{x})}$$
$$\sum_{\mathbf{x} \in B_{\log N}} \frac{\widetilde{m}_c(\mathbf{x})}{X + \widetilde{id}(\mathbf{x}) \cdot Y + \widetilde{eq}(\mathbf{x}, \mathbf{r}_y)} = \sum_{\mathbf{x} \in B_{\log N}} \frac{1}{X + \widetilde{col}(\mathbf{x}) \cdot Y + \widetilde{q}(\mathbf{x})}$$

The verifier probabilistically checks the above at $X = \alpha$, $Y = \beta$ for $\alpha, \beta \leftarrow \mathbb{F}$. The verifier sends $\gamma \leftarrow \mathbb{F}$ to batch the above checks. The prover sends oracle $\widetilde{g}$ to the polynomial interpolating the rational function

$$\widetilde{g}(\mathbf{x}) = \widetilde{m}_r(\mathbf{x})/\widetilde{d}_1(\mathbf{x}) - 1/\widetilde{d}_2(\mathbf{x}) + \gamma \widetilde{m}_c(\mathbf{x})/\widetilde{d}_3(x) - \gamma/\widetilde{d}_4(x)$$

where $\widetilde{d}_1(x), \widetilde{d}_2(x), \widetilde{d}_3(x)$ and $\widetilde{d}_4(x)$ denote the four denominators after setting $X = \alpha$ and $Y = \beta$, i.e,

$$\widetilde{d}_1(\mathbf{x}) = \alpha + \beta \cdot \widetilde{id}(\mathbf{x}) + \widetilde{eq}(\mathbf{x}, \mathbf{r}_x), \quad \widetilde{d}_2(\mathbf{x}) = \alpha + \beta \cdot \widetilde{row}(\mathbf{x}) + \widetilde{p}(\mathbf{x})$$
$$\widetilde{d}_3(\mathbf{x}) = \alpha + \beta \cdot \widetilde{id}(\mathbf{x}) + \widetilde{eq}(\mathbf{x}, \mathbf{r}_y), \quad \widetilde{d}_4(\mathbf{x}) = \alpha + \beta \cdot \widetilde{col}\mathbf{x} + \widetilde{q}(\mathbf{x})$$

To check correctness of $\widetilde{g}$, the verifier sends another challenge $\boldsymbol{\eta} \leftarrow \mathbb{F}^{\log N}$. The checks reduce to the following sum-check instances:

$$\sum_{\mathbf{x} \in B_{\log N}} \widetilde{eq}(\mathbf{x}, \boldsymbol{\eta}) \cdot \left(\widetilde{t}(\mathbf{x}) - \widetilde{g}(\mathbf{x}) \widetilde{d}_1(\mathbf{x}) \widetilde{d}_2(\mathbf{x}) \widetilde{d}_3(\mathbf{x}) \widetilde{d}_4(\mathbf{x}) \right) = 0$$
$$\sum_{\mathbf{x} \in B_{\log N}} \widetilde{g}(\mathbf{x}) = 0 \tag{21}$$

where we define $\widetilde{t}(\mathbf{x}) = \widetilde{m}_r(\mathbf{x}) \widetilde{D}_1(\mathbf{x}) - \widetilde{D}_2(\mathbf{x}) + \gamma \widetilde{m}_c(\mathbf{x}) \widetilde{D}_3(\mathbf{x}) - \gamma \widetilde{D}_4(\mathbf{x})$ and $\widetilde{D}_i(\mathbf{x}) = \prod_{j \neq i} \widetilde{d}_j(\mathbf{x})$. Using a random challenge, all the sum-check instances in Equations (20), and (21) can be aggregated into a single sum-check instance of the form

$$\overline{G}(\widetilde{row}, \widetilde{col}, \widetilde{val}_A, \widetilde{val}_B, \widetilde{val}_C, \widetilde{z}, \widetilde{p}, \widetilde{q}, \widetilde{m}_r, \widetilde{m}_c, \widetilde{g}, \widetilde{id}, \widetilde{eq}(\cdot, \mathbf{r}_x), \widetilde{eq}(\cdot, \mathbf{r}_y)) = \sigma \tag{22}$$

where $\overline{G}$ is a multivariate polynomial of total degree ≤ 6.

A Titan PCS in the Random Oracle Model

A.1 Complete Protocol

We now describe Titan PCS as a polynomial commitment scheme in the Random Oracle Model. Rather than composing the two interactive parts and compiling the composition, we compile each component separately and define the evaluation protocol as the sequential composition of the two resulting non-interactive arguments.

Compiled outer evaluation IOP. Let

$$(\mathcal{P}^{\mathrm{FS}}_{\mathrm{outer}}, \mathcal{V}^{\mathrm{FS}}_{\mathrm{outer}}) := \mathsf{FS}[\Pi_{\mathrm{outer}}]$$

be the non-interactive argument in the ROM obtained by compiling the IOP Π_{outer} of Section 5.1 using the BCS transformation [BCS16], for the relation

$$\mathcal{R}^{\mathsf{RO}}_{\mathrm{outer}} := \left\{ \left((\mathsf{pp}, \mathsf{rt}, \mathbf{x}, C, \delta);\ \widetilde{G}\right) : \begin{array}{l} \mathsf{rt} = \mathsf{MerkleTree}^{\mathsf{RO}}(G) \ \wedge\ \Delta(G, \widetilde{G}) < \delta \\ \wedge\ \widetilde{G} \in \mathsf{RS}[\mathbb{G}, \mathsf{L}_1, \ell_p] \ \wedge\ \widetilde{G}(\mathbf{x}) = C \end{array} \right\} \tag{23}$$

We write $\mathcal{L}^{\mathsf{RO}}_{\mathrm{outer}}$ for the language of statements admitting a witness in $\mathcal{R}^{\mathsf{RO}}_{\mathrm{outer}}$, i.e.

$$\mathcal{L}^{\mathsf{RO}}_{\mathrm{outer}} := \left\{ (\mathsf{pp}, \mathsf{rt}, \mathbf{x}, C) \ :\ \exists\, \widetilde{G} \text{ s.t. } \left((\mathsf{pp}, \mathsf{rt}, \mathbf{x}, C, \delta); \widetilde{G}\right) \in \mathcal{R}^{\mathsf{RO}}_{\mathrm{outer}} \right\}. \tag{24}$$

Here the oracle $[\![G]\!]$ of the IOP is realized as the Merkle commitment $\mathsf{rt} = \mathsf{MerkleTree}^{\mathsf{RO}}(G)$ as described in Section 3.5.1, and rt is part of the statement; the prover's messages consist of the authentication paths for the t_1 queried positions together with the folding messages of the iteration.

Compiled inner evaluation. Similarly, let

$$(\mathcal{P}^{\mathrm{FS}}_{\mathrm{inner}}, \mathcal{V}^{\mathrm{FS}}_{\mathrm{inner}}) := \mathsf{FS}[\Pi_{\mathrm{inner}}]$$

be the non-interactive argument in the ROM for the relation $\mathcal{R}_{\mathrm{inner}}$ of Lemma 5.2, obtained by compiling the $(3, \ldots, 3)$-special-sound interactive protocol Π_{inner} of Section 5.2 with the Fiat–Shamir transformation. We assume throughout that the two protocols use independent random oracles $\mathsf{RO}_{\mathrm{outer}}$ and $\mathsf{RO}_{\mathrm{inner}}$, which in practice can be from a single oracle RO by standard domain separation.

The evaluation protocol. We construct the non-interactive protocol $\Pi_{\mathbf{Eval}}$ in the ROM for the relation $\mathcal{R}_{\mathbf{Eval}}$ of Definition 3.3 as the sequential composition of $\mathsf{FS}[\Pi_{\mathrm{outer}}]$ and $\mathsf{FS}[\Pi_{\mathrm{inner}}]$, as described in Figure 10. We summarise the resulting construction of Titan PCS, now as a tuple of algorithms with access to a random oracle, in Figure 11. The properties are formally captured in Theorem A.1.

Before we proceed with the proof, we formally define the soundness guarantees of the compiled non-interactive arguments and the *random point evaluation* game for the non-interactive PCS. Let pp denote public parameters defined in Figure 3.

Lemma A.1 ($\mathsf{FS}[\Pi_{\mathrm{outer}}]$, from [BCS16])**.** *For any Q-query PPT adversary $\mathcal{A}$, we have:*

$$\Pr\left[\begin{array}{c} \mathcal{V}^{\mathrm{FS},\mathsf{RO}}_{\mathrm{outer}}((\mathsf{pp}, \mathsf{rt}, \mathbf{x}, C, \pi) = 1 \\ \wedge \\ (\mathsf{pp}, \mathsf{rt}, \mathbf{x}, C) \notin \mathcal{L}^{\mathsf{RO}}_{\mathrm{outer}} \end{array} \;\middle|\; \begin{array}{c} \mathsf{RO} \leftarrow U(\lambda) \\ (\mathsf{rt}, \mathbf{x}, C) \leftarrow \mathcal{A}^{\mathsf{RO}}(\mathsf{pp}) \end{array} \right] \leq \varepsilon_{\mathrm{outer}}(Q)$$

where $\varepsilon_{\mathrm{outer}}(Q) := Q \cdot (1-\delta)^{t_1} + O(Q^2/2^\lambda)$, t_1 denotes the number of queries made by the IOP verifier, while δ as before denotes the proximity parameter set to $(1-\rho_1)/2$.

Since, Π_{inner} is a $(\mu_1, \ldots, \mu_{k_2})$-special sound protocol for $\mu_i = 3$, the Fiat-Shamir compilation $\mathsf{FS}[\Pi_{\text{inner}}]$, using results for multi-round interactive protocols in [AFK23] implies:

Lemma A.2 ($\mathsf{FS}[\Pi_{\text{inner}}]$, from [AFK23])**.** *For any Q-query PPT adversary $\mathcal{A}$, there exists an expected polynomial time extractor* Ext *such that the following holds:*

$$\Pr\left[\begin{array}{c|c} \mathcal{V}_{\text{inner}}^{\text{FS},\mathsf{RO}}((\mathsf{pp}, C, \mathbf{y}, \sigma), \pi) = 1 \\ \wedge \\ ((C, \mathbf{y}, \sigma); \mathbf{a}) \notin \mathcal{R}_{\text{inner}} \end{array} \;\middle|\; \begin{array}{c} \mathsf{RO} \leftarrow U(\lambda) \\ (C, \mathbf{y}, \sigma, \pi) \xleftarrow{\mathsf{tr}} \mathcal{A}^{\mathsf{RO}}(\mathsf{pp}) \\ \mathbf{a} \leftarrow \mathsf{Ext}(\mathcal{A}, \mathsf{pp}, \mathsf{tr}, C, \mathbf{y}, \sigma, \pi) \end{array}\right] \leq \varepsilon_{\text{inner}}(Q)$$

where:

$$\varepsilon_{\text{inner}}(Q) \;\leq\; (Q+1) \cdot \left(3^{k_2} - 1\right) \cdot \left(\varepsilon_{\text{CSP}} + 3^{k_2} \rho_2^{t_2}\right).$$

with parameters k_2, t_2, ρ_2 as defined in protocol Π_{outer}.

Finally, we define the relation $\mathcal{R}_{\text{eval}}^{\mathsf{RO}}$ and *random point evaluation* in the ROM as follows. Both relations below are *parametrized by the random oracle* as the commitment rt is a Merkle root, and $\mathsf{MerkleTree}^{\mathsf{RO}}$ is itself RO-keyed. The evaluation relation is the ROM analogue of $\mathcal{R}_{Ev}$ of Definition 3.3, obtained by replacing the function oracle $[\![f]\!]$ with the Merkle root rt. Writing it out in full, with $\mathbf{z} = (\mathbf{x}, \mathbf{y})$ parsed as in Section 4.3:

$$\mathcal{R}_{\text{eval}}^{\mathsf{RO}} := \left\{ \left((\mathsf{pp}, \mathsf{rt}, \mathbf{x}, \mathbf{y}, \sigma);\ \widetilde{f}\right) \;:\; \begin{array}{ll} \text{(i)} & \widetilde{f} \in \mathbb{F}^{\leq 1}[X_1, \ldots, X_m] \ \wedge\ \widetilde{f}(\mathbf{x}, \mathbf{y}) = \sigma \\ \text{(ii)} & \widetilde{G} = \mathsf{Com}(\widetilde{f}\,;\, \mathbf{g}) \\ \text{(iii)} & \mathsf{rt} = \mathsf{MerkleTree}^{\mathsf{RO}}(G') \ \wedge\ \Delta(G', \widetilde{G}) < \delta \end{array} \right\} \tag{25}$$

Next, we define the random point evaluation game in the Random Oracle Model (ROM).

Definition A.1 (Random Point Evaluation in ROM)**.** *We say that a PPT adversary $\mathcal{A}$ wins the random point evaluation game* $\mathsf{RPE}(\lambda)$ *against an extractor* Ext *with rewinding access to $\mathcal{A}$with probability at least ϵ, if:*

$$\Pr\left[\begin{array}{c|c} \mathcal{V}_{\text{outer}}^{\text{FS},\mathsf{RO}_1}(\mathsf{pp}, \mathsf{rt}, \mathbf{x}, C, \pi_1) = 1 \\ \wedge \\ \mathcal{V}_{\text{inner}}^{\text{FS},\mathsf{RO}_2}(\mathsf{pp}, C, \mathbf{y}, \sigma, \pi_2) = 1 \\ \wedge \\ (\mathsf{pp}, \mathsf{rt}, \mathbf{x}, \mathbf{y}, \sigma; \widetilde{f}) \notin \mathcal{R}_{\text{eval}}^{\mathsf{RO}_1} \end{array} \;\middle|\; \begin{array}{c} \mathsf{RO}_1, \mathsf{RO}_2 \leftarrow U(\lambda) \\ \mathsf{rt}, \mathsf{st} \xleftarrow{\mathsf{tr}_1} \mathcal{A}_1^{\mathsf{RO}_1, \mathsf{RO}_2}(\mathsf{pp}) \\ (\mathbf{x}, \mathbf{y}) \leftarrow \mathbb{F}^p \times \mathbb{F}^q \\ \pi := (C, \pi_1, \pi_2, \sigma) \xleftarrow{\mathsf{tr}_2} \mathcal{A}_2^{\mathsf{RO}_1, \mathsf{RO}_2}(\mathsf{st}, \mathbf{x}, \mathbf{y}) \\ \widetilde{f} \leftarrow \mathsf{Ext}(\mathcal{A}_2(\mathsf{st}, \cdot), \mathsf{tr}_1, \mathsf{tr}_2, \mathsf{rt}, \mathbf{x}, \mathbf{y}, \sigma, \pi) \end{array}\right] \geq \epsilon$$

Intuitively, a successful adversary in an $\mathsf{RPE}(\lambda)$ game must succeed against $\mathrm{FS}[\Pi_{\text{outer}}]$ and $\mathrm{FS}[\Pi_{\text{inner}}]$ on statements sharing a common commitment C, which is what links the two stages of evaluation of $\widetilde{f}$; the partial evaluation opens to C and simultaneously evaluates to σ at the remaining coordinates.

Theorem A.1. *The scheme Titan PCS consisting of algorithms* ($\mathbf{Setup}^{\mathsf{RO}}$, $\mathbf{Commit}^{\mathsf{RO}}$, $\mathbf{Open}^{\mathsf{RO}}$, $\mathbf{Eval}^{\mathsf{RO}}$) *as in Figure 11 for polynomials in $\mathbb{F}^{\leq 1}[X_1, \ldots, X_m]$ satisfies* completeness*,* binding*, and* knowledge soundness with random point evaluation *in the ROM. For an adversary making at most Q queries to* RO*, the knowledge soundness error is*

$$\varepsilon_{\text{RPE}} \;\leq\; (p+1)\big(\varepsilon_{\text{outer}}(Q) + \varepsilon_{\text{inner}}(Q)\big) + \varepsilon_{\text{DLOG}} + \frac{p \log p}{|\mathbb{F}|}$$

where $\varepsilon_{\text{outer}}(Q) \leq Q \cdot (1-\delta)^{t_1} + O(Q^2/2^\lambda)$ is the soundness error of $\mathsf{FS}[\Pi_{\text{outer}}]$ from Lemma A.1 and $\varepsilon_{\text{inner}}(Q) \leq (Q+1)(3^{k_2} - 1)(\varepsilon_{\text{CSP}} + 3^{k_2}\rho_2^{t_2})$ is the knowledge soundness error of $\mathsf{FS}[\Pi_{\text{inner}}]$ from Lemma A.2. Here t_1 and t_2 denote the number of consistency queries in Π_{outer} and Π_{inner} respectively, while ε_{CSP} and $\varepsilon_{\text{DLOG}}$ denote the soundness of standard CSP and the adversary's success probability in breaking discrete-log in $\mathbb{G}$ respectively. In particular ε_{RPE} degrades from the interactive setting only by factors polynomial in Q and p.

- **Public Parameters**: pp as in Figure 3 and $\delta := (1-\rho_1)/2$.
- **Random Oracles**: $\mathsf{RO}_{\text{outer}}$, $\mathsf{RO}_{\text{inner}}$, obtained from RO by domain separation.
- **Common Input**: $\mathbf{z} \in \mathbb{F}^m$, $\sigma \in \mathbb{F}$, $\mathsf{rt} = \mathsf{MerkleTree}^{\mathsf{RO}}(G)$.
- **Prover's Input**: $\widetilde{f} \in \mathbb{F}^{\leq 1}[X_1, \ldots, X_m]$, $\widetilde{G} = \mathsf{Com}(\widetilde{f}\,; \mathbf{g})$, the word G.
- **Evaluation Protocol ($\Pi_{\mathbf{Eval}}$), non-interactive**: Parse $\mathbf{z}$ as $(\mathbf{x}, \mathbf{y})$ with $\mathbf{x} \in \mathbb{F}^{\ell_p}$ and $\mathbf{y} \in \mathbb{F}^{\ell_q}$.

  $\mathcal{P}_{\mathbf{Eval}}$ outputs the proof $\pi = (C, \pi_{\text{outer}}, \pi_{\text{inner}})$, where:

  1. $C = \mathsf{Com}(\widetilde{f}(\mathbf{x}, \mathbf{Y})\,; \mathbf{g})$ is the commitment to the partial evaluation and $\mathbf{a} \in \mathbb{F}^q$ its coefficient vector, i.e. $\widetilde{f}(\mathbf{x}, \mathbf{Y}) = \sum_{i=1}^{q} a_i \cdot \widetilde{eq}(\mathbf{Y}, \langle i \rangle)$.
  2. $\pi_{\text{outer}} \leftarrow \mathcal{P}^{\text{FS}}_{\text{outer}}\big((\mathsf{pp}, \mathsf{rt}, \mathbf{x}, C, \delta);\ \widetilde{G}\big)$, a proof for $\mathcal{R}_{\text{outer}}$ of (23).
  3. $\pi_{\text{inner}} \leftarrow \mathcal{P}^{\text{FS}}_{\text{inner}}\big((\mathsf{pp}, C, \mathbf{y}, \sigma);\ \mathbf{a}\big)$, a proof for $\mathcal{R}_{\text{inner}}$ of Lemma 5.2.

  $\mathcal{V}_{\mathbf{Eval}}(\mathsf{pp}, \mathsf{rt}, \mathbf{z}, \sigma, \pi)$ parses $\pi = (C, \pi_{\text{outer}}, \pi_{\text{inner}})$ and returns

  $$\mathcal{V}^{\text{FS}}_{\text{outer}}\big((\mathsf{pp}, \mathsf{rt}, \mathbf{x}, C, \delta), \pi_{\text{outer}}\big) \ \wedge\ \mathcal{V}^{\text{FS}}_{\text{inner}}\big((\mathsf{pp}, C, \mathbf{y}, \sigma), \pi_{\text{inner}}\big).$$

  The commitment C is the only value shared between the two sub-proofs — it is the statement of the inner argument and part of the statement of the outer one — which is what makes the composition sequential.

Figure 10: Titan Evaluation Protocol in the ROM: sequential composition of $\mathsf{FS}[\Pi_{\text{outer}}]$ and $\mathsf{FS}[\Pi_{\text{inner}}]$

- $\mathbf{Setup}^{\mathsf{RO}}(1^\lambda, m)$: Outputs pp in Figure 3 as described in Section 4.1. Set $\delta := (1-\rho_1)/2$.
- $\mathbf{Commit}^{\mathsf{RO}}(\mathsf{pp}, \widetilde{f}) \to \mathsf{rt}$: Constructs the word G as in Section 4.2 and outputs the Merkle root $\mathsf{rt} = \mathsf{MerkleTree}^{\mathsf{RO}}(G)$. The prover retains G and $\widetilde{G} = \mathsf{Com}(\widetilde{f}\,; \mathbf{g})$ as decommitment state.
- $\mathbf{Open}^{\mathsf{RO}}(\mathsf{pp}, \mathsf{rt}; \widetilde{f}) \to b$: Computes $\widetilde{G} = \mathsf{Com}(\widetilde{f}\,; \mathbf{g})$ and the associated word G' as in Section 4.2, and outputs $b = 1$ if $\mathsf{rt} = \mathsf{MerkleTree}^{\mathsf{RO}}(G')$ and $\Delta(\widetilde{G}, G') < \delta$. Outputs $b = 0$ otherwise.
- $\mathbf{Eval}^{\mathsf{RO}}$: The non-interactive evaluation protocol, consisting of $\pi \leftarrow \mathcal{P}^{\mathsf{RO}}_{\mathbf{Eval}}(\mathsf{pp}, \mathsf{rt}, \mathbf{z}, \sigma; \widetilde{f})$ and $b \leftarrow \mathcal{V}^{\mathsf{RO}}_{\mathbf{Eval}}(\mathsf{pp}, \mathsf{rt}, \mathbf{z}, \sigma, \pi)$, is given by Figure 10.

Figure 11: Titan PCS in the ROM under Unique Decoding Regime. All algorithms have access to the random oracle RO, from which $\mathsf{RO}_{\text{outer}}$ and $\mathsf{RO}_{\text{inner}}$ are obtained by domain separation.

Proof. We skip completeness as it follows trivially by construction. To prove binding, suppose PPT adversary $\mathcal{A}$ outputs $(\mathsf{rt}, \widetilde{f_0}, \widetilde{f_1})$ in the binding game. If $\widetilde{f_0}$ and $\widetilde{f_1}$ are two distinct polynomials in $\mathbb{F}^{\leq 1}[X_1, \ldots, X_m]$ satisfying $\mathbf{Open}^{\mathsf{RO}}(\mathsf{pp}, \mathsf{rt}; \widetilde{f_i}) = 1$ for $i = 0, 1$, then both openings exhibit words G'_i with $\mathsf{MerkleTree}^{\mathsf{RO}}(G'_i) = \mathsf{rt}$; either $G'_0 \neq G'_1$, in which case $\mathcal{A}$ has found a Merkle collision, which occurs with probability $O(Q^2/2^\lambda)$, or $G'_0 = G'_1 =: G$ and by definition of $\mathbf{Open}^{\mathsf{RO}}$ we must have $\Delta(G, \widetilde{G_i}) < \delta$ for $\widetilde{G_i} = \mathsf{Com}(\widetilde{f_i}\,; \mathbf{g})$. Since $\delta < (1-\rho_1)/2$, by triangle inequality we must have $\Delta(\widetilde{G_0}, \widetilde{G_1}) < 1 - \rho_1$. As $\widetilde{G_0}$ and $\widetilde{G_1}$ are codewords (we identify polynomials with their respective encodings) in $\mathsf{RS}[\mathbb{G}, \mathsf{L}_1, \ell_p]$ which has minimum distance $1-\rho_1$, we must have $\widetilde{G_0} = \widetilde{G_1}$. Thus $(\widetilde{f_0}, \widetilde{f_1})$ break the "binding" of Pedersen commitment scheme, and hence any PPT adversary $\mathcal{A}$ outputs them with negligible probability.

We now prove knowledge-soundness with random point evaluation. Informally, the outer argument $\mathsf{FS}[\Pi_{\text{outer}}]$ binds the prover to a unique multilinear polynomial $\widetilde{G}$ with coefficients $G_1, \ldots, G_p \in \mathbb{G}$. Subsequently, for each evaluation claim for $\mathbf{z} = (\mathbf{x}, \mathbf{y})$, the prover shows knowledge of opening for a linear combination of G_i's determined by $\mathbf{x}$, specifically for $C = \sum_{i=1}^{p} \widetilde{eq}(\mathbf{x}, \langle i \rangle) \cdot G_i$. These openings for several random $\mathbf{x}$ can be combined to obtain openings

for each G_i, $i \in [t]$, which determine the "rows" of the polynomial $\widetilde{f}$. The formal proof follows.

Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ denote the adversary in the game of Definition A.1, winning with probability ϵ. Without loss of generality, we assume $\mathcal{A}$ is deterministic, which The extractor Ext receives rt and the first-stage transcript tr_1 from the game. It neither runs nor rewinds $\mathcal{A}_1$. Its rewinding access is to $\mathcal{A}_2(\mathsf{st}, \cdot)$ alone, which it invokes on uniformly sampled points $(\mathbf{x}_i, \mathbf{y}_i) \leftarrow \mathbb{F}^{\log p} \times \mathbb{F}^{\log q}$, retaining those invocations on which both verifiers accept:

$$\big(\mathsf{rt}, \mathbf{x}_i, \mathbf{y}_i, \sigma_i, \pi_i = (C_i, \pi_{1,i}, \pi_{2,i})\big), \quad i \in [p].$$

As in the proof of Theorem 5.1, Ext follows the *early abort* rule: It performs one invocation first, and *aborts immediately* if that invocation is not accepting; otherwise, it continues running $\mathcal{A}_2$ on fresh challenges till it finds $p-1$ further accepting invocations. All p retained outputs carry the same rt, since rt is fixed by the game before the invocations are forked. Again, as before this yields an expected polynomial time extractor Ext.

The committed word G is obtained *online* from tr_1: since tr_1 contains $\mathcal{A}_1$'s queries to RO_1 and $\mathsf{rt} = \mathsf{MerkleTree}^{\mathsf{RO}}(G)$, the Merkle-tree extraction of the BCS transformation reads G off that transcript, without rewinding or re-execution of $\mathcal{A}_1$.

Similar to the proof of Theorem 5.1, we carry the bulk of analysis conditioned on the following hypothesis: for each accepting run, the outer evaluation is correct, and that the inner extractor succeeds. To prove that the hypothesis holds execpt with negligible probability, we translate $\mathcal{A}$ violating the hypothesis into adversaries against Lemmas A.1 and A.2. In each case, the reduction samples one random oracle and *defers* the other random oracle to the target game.

Call a run of $\mathcal{A}$ an *accepting run* if both proofs verify, i.e.

$$\mathcal{V}_{\text{outer}}^{\text{FS},\mathsf{RO}_1}(\mathsf{pp}, \mathsf{rt}, \mathbf{x}, C, \pi_1) = 1 \quad \wedge \quad \mathcal{V}_{\text{inner}}^{\text{FS},\mathsf{RO}_2}(\mathsf{pp}, C, \mathbf{y}, \sigma, \pi_2) = 1,$$

which is exactly the acceptance condition of $\mathcal{V}_{\mathbf{Eval}}$ in Figure 10. Call an accepting run a *bad accepting run* if, in addition, the conditions of one of the two relations fail: either $(\mathsf{pp}, \mathsf{rt}, \mathbf{x}, C) \notin \mathcal{L}_{\text{outer}}^{\mathsf{RO}_1}$, or $\mathsf{Ext}_{\text{inner}}$ fails to return $\mathbf{a}$ with $\big((C, \mathbf{y}, \sigma); \mathbf{a}\big) \in \mathcal{R}_{\text{inner}}$. Thus an accepting run that is not bad implies the hypothesis. Note that a run in which some proof fails to verify is not an accepting run at all, and is simply discarded by Ext.

Claim A.1. *For a single run of $\mathcal{A}$ in the game of Definition A.1,*

$$\Pr[\textit{the run is a bad accepting run}] \;\le\; \varepsilon_{\text{outer}}(Q) + \varepsilon_{\text{inner}}(Q).$$

Proof of Claim A.1. A bad accepting run lies in the union of

$$E_{\text{outer}} := \Big[\mathcal{V}_{\text{outer}}^{\text{FS},\mathsf{RO}_1} \text{ accepts } \wedge\; (\mathsf{pp}, \mathsf{rt}, \mathbf{x}, C) \notin \mathcal{L}_{\text{outer}}^{\mathsf{RO}_1}\Big]$$

and the analogous event E_{inner} for the inner argument, since badness requires one of the two relations to fail while the corresponding verifier accepts. It therefore suffices to bound each.

For E_{outer}, define $\mathcal{B}_{\text{outer}}$ against the game of Lemma A.1: on receiving RO from that game, it samples $\mathsf{RO}_{\text{inner}} \leftarrow U(\lambda)$ itself, runs the experiment of Definition A.1 with $\mathsf{RO}_1 := \mathsf{RO}$ and $\mathsf{RO}_2 := \mathsf{RO}_{\text{inner}}$. Specifically, $\mathcal{B}_{\text{outer}}$ runs $\mathcal{A}_1$ followed by $\mathcal{A}_2$ on a uniformly sampled point $(\mathbf{x}, \mathbf{y})$ and outputs the statement $(\mathsf{rt}, \mathbf{x}, C)$ with proof π_1. The swap is distribution preserving: RO_1 and RO_2 are independent in the RPE game, so receiving one from the outer game and sampling the other internally reproduces the RPE oracle distribution exactly, and $\mathcal{B}_{\text{outer}}$ is a single run of $\mathcal{A}$, hence strictly PPT as Lemma A.1 requires. E_{outer} is precisely the event that $\mathcal{B}_{\text{outer}}$ wins, so $\Pr[E_{\text{outer}}] \le \varepsilon_{\text{outer}}(Q)$.

For E_{inner}, define $\mathcal{B}_{\text{inner}}$ symmetrically, receiving RO from the game of Lemma A.2, sampling $\mathsf{RO}_{\text{outer}}$ itself, and outputting $(C, \mathbf{y}, \sigma)$ with proof π_2. The same distribution-preserving

swap applies. Note that C is produced by $\mathcal{A}$ after querying its oracles rather than fixed in advance, so $\mathcal{B}_{\text{inner}}$ is an adaptive adversary. Since, Lemma A.2 holds against adaptive adversaries, it yields $\Pr[E_{\text{inner}}] \le \varepsilon_{\text{inner}}(Q)$. A union bound gives the claim. □

Write $\varepsilon_{\text{bad}} := \varepsilon_{\text{outer}}(Q) + \varepsilon_{\text{inner}}(Q)$. Claim A.1 applies to any single run of $\mathcal{A}$, and the analysis below invokes it for $p+1$ runs: the p invocations Ext retains, and the *challenge* run of Definition A.1, on the point $(\mathbf{x}, \mathbf{y})$ sampled by the game itself. The latter is a run distinct from the former, since Ext samples its own points, and the evaluation claim below needs it to be accepting and not bad. A union bound over these $p+1$ runs gives

$$\Pr\left[\text{some of the } p+1 \text{ runs is a bad accepting run}\right] \;\le\; (p+1)\cdot\varepsilon_{\text{bad}}, \tag{26}$$

so with probability at least $\epsilon - (p+1)\cdot\varepsilon_{\text{bad}}$ the p retained invocations and the challenge run are all accepting with none of them bad.

Assume henceforth that aforementioned $p+1$ runs are accepting and none is bad. Then for every $i \in [p]$ we have both $(\mathsf{pp}, \mathsf{rt}, \mathbf{x}_i, C_i) \in \mathcal{L}_{\text{outer}}^{\mathsf{RO}_1}$ and an opening $\mathbf{a}_i$ of C_i. In the unique decoding regime $\delta < (1-\rho_1)/2$ the outer witness is unique, so a single $\widetilde{G} \in \mathsf{RS}[\mathbb{G}, \mathsf{L}_1, \ell_p]$ with $\mathsf{rt} = \mathsf{MerkleTree}^{\mathsf{RO}}(G)$ and $\Delta(G, \widetilde{G}) < \delta$ satisfies:

$$C_i \;=\; \widetilde{G}(\mathbf{x}_i) \;=\; \sum_{j=1}^{p} \widetilde{eq}(\mathbf{x}_i, \langle j\rangle)\cdot G_j \quad \text{for all } i \in [p], \tag{27}$$

while the openings returned by $\mathsf{Ext}_{\text{inner}}$ satisfy

$$\sum_{j=1}^{q} a_{ij}\cdot g_j = C_i \quad\wedge\quad \sum_{j=1}^{q} a_{ij}\cdot \widetilde{eq}(\mathbf{y}_i, \langle j\rangle) = \sigma_i \quad \text{for all } i \in [p]. \tag{28}$$

where we write $G_1, \ldots, G_p$ for the (purported) hypercube evaluations of $\widetilde{G}$. To determine $\widetilde{G}$, the extractor Ext sets up the following system of equations with unknowns G_i, $i \in [p]$:

$$\begin{bmatrix} \widetilde{eq}(\mathbf{x}_1,\langle 1\rangle) & \widetilde{eq}(\mathbf{x}_1,\langle 2\rangle) & \cdots & \widetilde{eq}(\mathbf{x}_1,\langle p\rangle) \\ \widetilde{eq}(\mathbf{x}_2,\langle 1\rangle) & \widetilde{eq}(\mathbf{x}_2,\langle 2\rangle) & \cdots & \widetilde{eq}(\mathbf{x}_2,\langle p\rangle) \\ \vdots & \vdots & \ddots & \vdots \\ \widetilde{eq}(\mathbf{x}_p,\langle 1\rangle) & \widetilde{eq}(\mathbf{x}_p,\langle 2\rangle) & \cdots & \widetilde{eq}(\mathbf{x}_p,\langle p\rangle) \end{bmatrix} \begin{bmatrix} G_1 \\ G_2 \\ \vdots \\ G_p \end{bmatrix} = \begin{bmatrix} C_1 \\ C_2 \\ \vdots \\ C_p \end{bmatrix} \tag{29}$$

$$= \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1q} \\ a_{21} & a_{22} & \cdots & a_{2q} \\ \vdots & \vdots & \ddots & \vdots \\ a_{p1} & a_{p2} & \cdots & a_{pq} \end{bmatrix} \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_q \end{bmatrix} \tag{30}$$

As shown in Lemma 5.5 (reproduced from [ZCF24]) the matrix consisting of evaluations $\widetilde{eq}(\mathbf{x}_i, \langle j\rangle)$ for uniformly sampled $\mathbf{x}_i$ is invertible with overwhelming probability. Thus, Ext can compute $p \times q$ matrix $\mathbf{F}$ such that $\mathbf{F}\cdot\mathbf{g} = \mathbf{G}$. The extractor outputs the unique multilinear polynomial $\widetilde{f}$ satisfying $\widetilde{f}(\langle i\rangle, \langle j\rangle) = F_{ij}$ for $i \in [p]$, $j \in [q]$.

We now verify that this $\widetilde{f}$ meets conditions (i)–(iii) of $\mathcal{R}_{\text{eval}}^{\mathsf{RO}}$ in (25). Condition (ii) holds by construction: the j-th row of $\mathbf{F}$ satisfies $\sum_i F_{ji}\, g_i = G_j$, so $G_j = \mathsf{Com}(\widetilde{f_j}\,; \mathbf{g})$, and thus $\widetilde{G} = \mathsf{Com}(\widetilde{f}\,; \mathbf{g})$. Due to δ being less than the unique decoding radius, we see that the unique $\widetilde{G}$ which witnesses the $p+1$ statements in $\mathcal{L}_{\text{outer}}$ for the accepting transcripts, also satisfies Equation 29. Except with probability $p\log p/|\mathbb{F}|$ where the $\widetilde{eq}$-matrix is degenerate, the solution to the system of equations is unique, and hence the coefficients $G_1, \ldots, G_p$ obtained by solving the system are the coefficients of $\widetilde{G}$. Thus (iii) holds. Finally, we need to show that

$\widetilde{f}(\mathbf{x}, \mathbf{y}) = \sigma$. As before, C is the commitment to the polynomial $\widetilde{f}(\mathbf{x}, \mathbf{Y})$ and the polynomial $\widetilde{a}(\mathbf{Y})$ with extracted coefficients $a_1, \ldots, a_q$ (due to the fact that the main transcript is also not bad). If $\widetilde{f}(\mathbf{x}, \mathbf{Y}) \neq \widetilde{a}(\mathbf{Y})$, the pair breaks the binding of Pedersen commitment and thus yields a non-trivial discrete-log relation among the generators $\mathbf{g}$. The probability of this occurance is therefore at most ϵ_{DLOG}. When $\widetilde{f}(\mathbf{x}, \mathbf{Y}) = \widetilde{a}(\mathbf{Y})$, we have $\widetilde{f}(\mathbf{x}, \mathbf{y}) = \widetilde{a}(\mathbf{y}) = \sigma$. Thus $\widetilde{f}$ satisfies all the conditions required of the witness for $\mathcal{R}^{\mathsf{RO}}_{\mathrm{eval}}$. The failure cases under the conditioning hypothesis account for probability of at most $p \log p / |\mathbb{F}| + \epsilon_{\mathrm{DLOG}}$, i.e.,

$$\Pr[\mathsf{Fail} \ \wedge \ \text{none of the } p+1 \text{ runs is bad}] \ \leq \ \varepsilon_{\mathrm{DLOG}} + \frac{p \cdot \log p}{|\mathbb{F}|}$$

Adding back the conditioning, we have shown

$$\Pr[\mathsf{Fail}] \ \leq \ (p+1)\, \varepsilon_{\mathrm{outer}}(Q) + (p+1)\, \varepsilon_{\mathrm{inner}}(Q) + \varepsilon_{\mathrm{DLOG}} + \frac{p \log p}{|\mathbb{F}|} \tag{31}$$

which is the bound in the statement. □

Acknowledgements

We thank Ariel Gabizon for helpful discussions and pointers. We also thank Hadas Zeilberger for explaining the subtle differences between Basefold and WHIR.

References

[AC20] Thomas Attema and Ronald Cramer. Compressed Σ-protocol theory and practical application to plug & play secure algorithmics. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 513–543. Springer, Cham, August 2020.

[ACFY24] Gal Arnon, Alessandro Chiesa, Giacomo Fenzi, and Eylon Yogev. STIR: Reed-solomon proximity testing with fewer queries. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part X*, volume 14929 of *LNCS*, pages 380–413. Springer, Cham, August 2024.

[ACFY25] Gal Arnon, Alessandro Chiesa, Giacomo Fenzi, and Eylon Yogev. WHIR: Reed-solomon proximity testing with super-fast verification. In Serge Fehr and Pierre-Alain Fouque, editors, *EUROCRYPT 2025, Part IV*, volume 15604 of *LNCS*, pages 214–243. Springer, Cham, May 2025.

[AFK23] Thomas Attema, Serge Fehr, and Michael Klooß. Fiat-shamir transformation of multi-round interactive proofs (extended version). *Journal of Cryptology*, 36(4):36, October 2023.

[AGL+23] Arasu Arun, Chaya Ganesh, Satya V. Lokam, Tushar Mopuri, and Sriram Sridhar. Dew: A transparent constant-sized polynomial commitment scheme. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *PKC 2023, Part II*, volume 13941 of *LNCS*, pages 542–571. Springer, Cham, May 2023.

[AHIV17] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 2087–2104. ACM Press, October / November 2017.

[ALM+92] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and hardness of approximation problems. In *33rd FOCS*, pages 14–23. IEEE Computer Society Press, October 1992.

[Ark] Arkworks-rs. poly-commit Library. `https://github.com/arkworks-rs/poly-commit`.

[AS92] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs; A new characterization of NP. In *33rd FOCS*, pages 2–13. IEEE Computer Society Press, October 1992.

[BBHR18a] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Fast reed-solomon interactive oracle proofs of proximity. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *ICALP 2018*, volume 107 of *LIPIcs*, pages 14:1–14:17. Schloss Dagstuhl, July 2018.

[BBHR18b] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Report 2018/046, 2018.

[BCC+16] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 327–357. Springer, Berlin, Heidelberg, May 2016.

[BCHO22] Jonathan Bootle, Alessandro Chiesa, Yuncong Hu, and Michele Orrù. Gemini: Elastic SNARKs for diverse environments. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 427–457. Springer, Cham, May / June 2022.

[BCR+19] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. Aurora: Transparent succinct arguments for R1CS. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 103–128. Springer, Cham, May 2019.

[BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 31–60. Springer, Berlin, Heidelberg, October / November 2016.

[BCS21] Jonathan Bootle, Alessandro Chiesa, and Katerina Sotiraki. Sumcheck arguments and their applications. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 742–773, Virtual Event, August 2021. Springer, Cham.

[BFLS91] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *23rd ACM STOC*, pages 21–31. ACM Press, May 1991.

[BGKS20] Eli Ben-Sasson, Lior Goldberg, Swastik Kopparty, and Shubhangi Saraf. DEEP-FRI: Sampling outside the box improves soundness. In Thomas Vidick, editor, *ITCS 2020*, volume 151, pages 5:1–5:32. LIPIcs, January 2020.

[BMM+21] Benedikt Bünz, Mary Maller, Pratyush Mishra, Nirvan Tyagi, and Psi Vesely. Proofs for inner pairing products and applications. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part III*, volume 13092 of *LNCS*, pages 65–97. Springer, Cham, December 2021.

[BMSS25] Benedikt Bünz, Tushar Mopuri, Alireza Shirzad, and Sriram Sridhar. DewTwo: A transparent PCS with quasi-linear prover, logarithmic verifier and 4.5KB proofs from falsifiable assumptions. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part VI*, volume 16005 of *LNCS*, pages 549–583. Springer, Cham, August 2025.

[CBBZ23] Binyi Chen, Benedikt Bünz, Dan Boneh, and Zhenfei Zhang. HyperPlonk: Plonk with linear-time prover and high-degree custom gates. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part II*, volume 14005 of *LNCS*, pages 499–530. Springer, Cham, April 2023.

[CHM+20] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Psi Vesely, and Nicholas P. Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 738–768. Springer, Cham, May 2020.

[CS25] Elizabeth Crites and Alistair Stewart. On reed–solomon proximity gaps conjectures. Cryptology ePrint Archive, Paper 2025/2046, 2025.

[EG25a] Liam Eagen and Ariel Gabizon. MERCURY: A multilinear polynomial commitment scheme with constant proof size and no prover FFTs. Cryptology ePrint Archive, Report 2025/385, 2025.

[EG25b] Liam Eagen and Ariel Gabizon. Revisiting the IPA-sumcheck connection. Cryptology ePrint Archive, Paper 2025/1325, 2025.

[FGL+96] Uriel Feige, Shafi Goldwasser, László Lovász, Shmuel Safra, and Mario Szegedy. Interactive proofs and the hardness of approximating cliques. *Journal of the ACM (JACM)*, 43(2):268–292, 1996.

[GLH+24] Yanpei Guo, Xuanming Liu, Kexi Huang, Wenjie Qu, Tianyang Tao, and Jiaheng Zhang. DeepFold: Efficient multilinear polynomial commitment from reed-solomon code and its application to zero-knowledge proofs. Cryptology ePrint Archive, Report 2024/1595, 2024.

[GLS+23] Alexander Golovnev, Jonathan Lee, Srinath T. V. Setty, Justin Thaler, and Riad S. Wahby. Brakedown: Linear-time and field-agnostic SNARKs for R1CS. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part II*, volume 14082 of *LNCS*, pages 193–226. Springer, Cham, August 2023.

[GMN22] Nicolas Gailly, Mary Maller, and Anca Nitulescu. SnarkPack: Practical SNARK aggregation. In Ittay Eyal and Juan A. Garay, editors, *FC 2022*, volume 13411 of *LNCS*, pages 203–229. Springer, Cham, May 2022.

[GPPS24] Chaya Ganesh, Sikhar Patranabis, Shubh Prakash, and Nitin Singh. GAPP: Generic aggregation of polynomial protocols. Cryptology ePrint Archive, Report 2024/1685, 2024.

[GPS25] Chaya Ganesh, Sikhar Patranabis, and Nitin Singh. Samaritan: Linear-time prover SNARK from new multilinear polynomial commitments. Cryptology ePrint Archive, Report 2025/419, 2025.

[GWC19] Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. PLONK: Permutations over Lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953, 2019.

[Hab22] Ulrich Haböck. Multivariate lookups based on logarithmic derivatives. Cryptology ePrint Archive, Report 2022/1530, 2022.

[Hab25] Ulrich Haböck. A note on mutual correlated agreement for reed-solomon codes. Cryptology ePrint Archive, Paper 2025/2110, 2025.

[KPV22] Assimakis A. Kattis, Konstantin Panarin, and Alexander Vlasov. RedShift: Transparent SNARKs from list polynomial commitments. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 1725–1737. ACM Press, November 2022.

[KT23] Tohru Kohrita and Patrick Towa. Zeromorph: Zero-knowledge multilinear-evaluation proofs from homomorphic univariate commitments. Cryptology ePrint Archive, Report 2023/917, 2023.

[KZG10] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 177–194. Springer, Berlin, Heidelberg, December 2010.

[Lee21] Jonathan Lee. Dory: Efficient, transparent arguments for generalised inner products and polynomial commitments. In Kobbi Nissim and Brent Waters, editors, *TCC 2021, Part II*, volume 13043 of *LNCS*, pages 1–34. Springer, Cham, November 2021.

[MBKM19] Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2111–2128. ACM Press, November 2019.

[Mic] Microsoft. Spartan: High Speed zkSNARKs Without Trusted Setup. `https://github.com/microsoft/Spartan`.

[PST13] Charalampos Papamanthou, Elaine Shi, and Roberto Tamassia. Signatures of correct computation. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 222–242. Springer, Berlin, Heidelberg, March 2013.

[Qua] Quarks ZK Developers. quarks-zk Rust Crate. `https://crates.io/crates/quarks-zk/0.1.5`.

[RRR16] Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Constant-round interactive proofs for delegating computation. In Daniel Wichs and Yishay Mansour, editors, *48th ACM STOC*, pages 49–62. ACM Press, June 2016.

[SB25] István András Seres and Péter Burcsi. Behemoth: Transparent polynomial commitment scheme with constant opening proof size and verifier time. In Abderrahmane Nitaj, Svetla Petkova-Nikova, and Vincent Rijmen, editors,

AFRICACRYPT 25, volume 15651 of *LNCS*, pages 343–369. Springer, Cham, July 2025.

[Set20] Srinath Setty. Spartan: Efficient and general-purpose zkSNARKs without trusted setup. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 704–737. Springer, Cham, August 2020.

[SL20] Srinath Setty and Jonathan Lee. Quarks: Quadruple-efficient transparent zkSNARKs. Cryptology ePrint Archive, Report 2020/1275, 2020.

[Wiz] WizardOfMenlo. WHIR Reference Implementation. `https://github.com/WizardOfMenlo/whir`.

[WTs+18] Riad S. Wahby, Ioanna Tzialla, abhi shelat, Justin Thaler, and Michael Walfish. Doubly-efficient zkSNARKs without trusted setup. In *2018 IEEE Symposium on Security and Privacy*, pages 926–943. IEEE Computer Society Press, May 2018.

[ZCF24] Hadas Zeilberger, Binyi Chen, and Ben Fisch. BaseFold: Efficient field-agnostic polynomial commitment schemes from foldable codes. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part X*, volume 14929 of *LNCS*, pages 138–169. Springer, Cham, August 2024.

[Zha25] Yuncong Zhang. Revisiting sum-check-based polynomial commitment schemes. Cryptology ePrint Archive, Paper 2025/2249, 2025.

[ZSC24] Jiaxing Zhao, Srinath Setty, and Weidong Cui. MicroNova: Folding-based arguments with efficient (on-chain) verification. Cryptology ePrint Archive, Report 2024/2099, 2024.