SubLogarithmic Linear Time SNARKs from Improved Sumcheck

Sikhar Patranabis[1], Nitin Singh[2], and Sayani Sinha[3]

[1,2]IBM Research India
[3]IIT Kharagpur, India

Abstract

We present HybridSpartan and HybridPlonk – the first SNARKs that simultaneously achieve linear-time prover, sublogarithmic proof size, logarithmic verifier, and also feature updatable setups. Our constructions are provably secure in the Random Oracle Model (ROM) and the Algebraic Group Model (AGM). As a core technical contribution (possibly of independent interest), we reduce the communication complexity of the classical sumcheck protocol for multivariate polynomials from logarithmic to sublogarithmic, while retaining linear prover complexity. For degree d multivariate polynomials in μ variables which can be decomposed into ℓ multilinear polynomials, our protocol achieves $O(\ell + d \log\log n)$ communication and $O(n)$ prover cost for $n = 2^\mu$. Our protocol leverages recently proposed multilinear polynomial commitment schemes (PCS) with linear-time prover and constant proof size.

Multivariate sumcheck is a key ingredient in the design of several prover-efficient SNARKs, such as Spartan (Crypto '20), HyperPlonk (Eurocrypt '23), MicroSpartan (S&P '25), Hyrax (S&P '18), Libra (Crypto '19), Gemini (Eurocrypt '22), Virgo (S&P '20) etc. All of these SNARKs incur $\Omega(\log n)$ proof size, with the smallest concrete proof sizes ranging from 5KB-10KB for circuit sizes in the range of 2^{20}-2^{30}.

We compile variants of the Spartan and HyperPlonk multilinear PIOPs with our improved sumcheck to realize two new SNARKs, namely HybridSpartan and HybridPlonk, that support R1CS and Plonkish constraints, respectively. Both of them achieve $O(n)$ prover, $O(\log\log n)$ proof size, and $O(\log n)$ verifier, while avoiding proof recursion and non-black-box use of cryptographic primitives. We implement HybridSpartan and HybridPlonk, and compare their performances with several state-of-the-art prover-efficient SNARKs. For circuits of size 2^{30}, HybridSpartan and HybridPlonk achieve proof sizes of 1.9 KB and 2.4 KB, respectively, which are $2.4 - 3.6\times$ smaller than the most compact prover-efficient SNARKs, while achieving comparable prover costs.

Contents

1 Introduction **3**

1.1 Our Contributions 4

1.2 Comparison with Related Work 6

2 Preliminaries **7**

2.1 Succinct Argument of Knowledge 8

2.2 Polynomial Commitment Scheme 8

2.3 The KZG PCS 9

2.4 Algebraic Group Model (AGM) 9

2.5 The q-DLOG Assumption 10

2.6 Polynomial IOP 10

3 Background and Technical Overview **11**

3.1 Background 11

3.2 Improved Sumcheck Argument with Sublogarithmic Rounds 12

4 Sublogarithmic Multivariate Sumcheck Argument **13**

4.1 Phase-1: Univariate sumcheck 15

4.2 Phase-2: Consistency of Multilinear Restrictions 16

4.3 Phase-3: Consistency of Univariate Evaluations 17

4.4 Putting Everything Together 19

5 HybridSpartan **29**

5.1 Background: MicroSpartan 29

5.2 HybridSpartan via Improved Sumcheck 31

5.3 Concrete Proof Size of HybridSpartan 31

6 HybridPlonk **32**

6.1 Background: HyperPlonk 32

6.2 HybridPlonk via Improved Sumcheck 33

6.3 Concrete Proof Size of HybridPlonk 34

7 Implementation and Evaluation **34**

A Ethical Considerations **41**

1 Introduction

A Succinct Non-interactive ARgument of Knowledge (SNARK) allows a prover to convince a verifier about the integrity of a computation such that the size of the proof and the work done by the verifier in checking the proof is much smaller than the size of the computation. Concretely, for an NP relation $\mathcal{R}$, a SNARK proof π corresponding to a statement $\mathbb{x}$ convinces a verifier (non-interactively) about the existence of witness $\mathbb{w}$ such that $\mathcal{R}(\mathbb{x}, \mathbb{w}) = 1$. Succinctness requires that the size of π and the verification complexity are both $O_\lambda(\text{polylog}(|\mathbb{x}| + |C|))$, where $|\mathbb{x}|$ is the size of the statement $\mathbb{x}$, $|C|$ is the number of gates in the circuit C that verifies the relation $\mathcal{R}(\mathbb{x}, \mathbb{w}) = 1$, and λ is the security parameter [1].

SNARKs were initially introduced as theoretical concepts in the seminal contributions of Micali [Mic94] and Kilian [Kil92]. However, they have since been refined in a long line of works [Gro10, Lip12, BCCT12, BCI+13, GGPR13, PHGR13, BCG+13, Lip13, BCTV14], and have recently seen real-world deployment in several applications [BCG+14, Lab17, rol21, BCJ+24, LVA+25, LKKO20, ZFZS20, SDP22].

Models for SNARKs. Efficient SNARKs typically require a Structured Reference String (SRS) to be generated at setup, and are often proven secure in an idealized model (e.g., Random Oracle Model or ROM, Generic Group Model or GGM, Algebraic Group Model or AGM). Setup for SNARKs in the SRS model can be broadly classified into three categories: (i) *transparent* (setup only uses public randomness and verifier randomness consists of only *public coins*), (ii) *trusted* (require a randomized, circuit-dependent preprocessing phase with secret random coins), and (iii) *universal and updatable* (a *one-time* setup can be used to prove statements about any computation; moreover, there is a mechanism to update by contributing to the randomness of the SRS, and the SRS is trusted as long as at least one of the updates is honest). While the first practical SNARKs were in the trusted SRS model [PHGR13, GGPR13], several recent (pairing-based) SNARKs are in the updatable SRS model [MBKM19, GWC19, CHM+20, Set20, BCHO22, CBBZ23, XZZ+19, ZSCZ25, GPS25], which is also the model that we use in this paper.

SNARKs from PIOP and PCS. Several modern SNARKs [WTs+18, GWC19, XZZ+19, CHM+20, Set20, SL20, BCHO22, XZS22, CBBZ23, GLS+23] are designed by compiling an information-theoretically secure Polynomial Interactive Oracle Proof (PIOP) using a cryptographic Polynomial Commitment Scheme (PCS). In a PIOP, the prover provides the verifier with oracle access to a set of polynomials, and the verifier queries these polynomial oracles at random challenge points to obtain evaluations, eventually outputting accept or reject. Compiling a PIOP using a PCS yields a public-coin succinct argument, where the polynomial oracles are realized via *commitments* to polynomials that a verifier can query for evaluations, together with proofs that the evaluations are indeed consistent with the commitments. The interactive argument is further compiled using Fiat-Shamir [FS87] to obtain a SNARK in the ROM.

Univariate and Multilinear PCS. SNARKs based on the above approach can be broadly classified into two categories: those realized by compiling univariate PIOPs using univariate PCS (such as Sonic [MBKM19], Marlin [CHM+20], Plonk [GWC19], and Pari [DMS24]), and those realized by compiling multilinear PIOPs using multilinear PCS (such as Hyrax [WTs+18], Libra [XZZ+19], Spartan [Set20], Kopis [SL20], Xiphos [SL20], Gemini [BCHO22], Orion [XZS22], Brakedown [GLS+23] and HyperPlonk [CBBZ23]). Standard pairing-based SNARKs employing univariate polynomial commitments, such as KZG [KZG10], achieve constant proof size and constant verification complexity, but typically incur $O(n \log n)$ prover-side field operations, where n is the size of the computation (often represented by the size of the circuit or the number of R1CS constraints) [2]. This becomes a bottleneck for applications requiring large prover computations [rol21, Lab17], and has sparked increased interest in designing *prover-efficient* SNARKs.

[1] Throughout the paper, all of our asymptotics are stated for growing instance size, and have implicit dependence on a fixed security parameter λ. Hence, we simply write $O(\cdot)$ instead of $O_\lambda(\cdot)$ for simplicity of exposition.

[2] The $O(n \log n)$ prover cost discussed here pertains to the pairing-based univariate-PCS/SNARK paradigm, which is more relevant to our construction. Other paradigms, notably FRI-based polynomial commitments and STARKs, offer different prover/communication trade-offs and are not the target of this comparison.

Prover-Efficient SNARKs. Pairing-based SNARKs employing multilinear PIOPs and multilinear PCS [WTs+18, XZZ+19, Set20, SL20, CBBZ23, BCHO22, XZS22, GLS+23] are prover-efficient in the sense that they achieve $O(n)$ prover [3]. However, they incur $O(\log n)$ proof size. With growing computation sizes reaching $\approx 2^{30}$, even the best known constructions incur concrete proof sizes $\approx$ 8-10 KB [4]. In comparison, Plonk [GWC19] achieves constant proof size of 0.61 KB. Testudo [CGG+23] achieves constant proof size by recursively composing sumcheck verification and inner pairing-product verification with a Groth16 [Gro16] SNARK. However, such a recursive composition requires non-native operations (such as target group scalar multiplications) to be encoded inside arithmetic circuits which makes it practically challenging, limits the choice of elliptic curves to instantiate the scheme, and prevents the possibility of proving security in the ROM. In this paper, we ask:

Can we design a linear-time prover SNARK with sublogarithmic proof size, without relying on proof recursion?

Multivariate Sumcheck. Several prover-efficient SNARKs [WTs+18, XZZ+19, Set20, ZXZS20, BCHO22, CBBZ23] are designed using PIOPs for *multivariate sumcheck*. These schemes inherit their logarithmic proof size from the logarithmic communication complexity of the classical sumcheck argument for multivariate polynomials [LFKN90]. Consequently, though recent works [GPS25, EG25] have proposed pairing-based multilinear PCS with linear-time prover and constant proof size, the communication complexity of multivariate sumcheck remains a major barrier towards achieving prover-efficient SNARKs with sublogarithmic proof size. This leads to the following question:

Can we design a multivariate sumcheck argument with linear-time prover and sublogarithmic communication?

1.1 Our Contributions

We answer both of the above questions in the affirmative. We exhibit a new multivariate sumcheck argument with linear-time prover and sublogarithmic communication. We then leverage this protocol to design two SNARKs – HybridSpartan (for R1CS constraints) and HybridPlonk (for Plonkish constraints) – the first SNARKs in the updatable SRS setting that simultaneously achieves $O(n)$ prover time, $O(\log \log n)$ proof size , and $O(\log n)$ verifier. The underlying interactive arguments are knowledge-sound in the Algebraic Group Model (AGM), inherited from the polynomial commitment schemes used in our construction; applying the Fiat–Shamir transform yields SNARKs whose security is established in the combined ROM + AGM setting. Next, we expand on our contributions.

Improved Multivariate Sumcheck. We leverage two recent multilinear PCS proposals, namely Samaritan [GPS25] and Mercury [EG25] to design a new sumcheck argument for multivariate polynomials with significantly smaller communication complexity than the classical sumcheck argument [LFKN90]. For degree d multivariate polynomials in μ variables which can be decomposed into ℓ multilinear polynomials, we exhibit a new multivariate sumcheck argument with $O(\ell + d \log \log n)$ communication for $n = 2^\mu$. As described in Section 4, the proposed argument retains the $O(n)$ prover cost (where the precise constant depends on ℓ, d and the multivariate form). Thus we improve on the $O(\log n)$ communication inherent in all existing applications of the multivariate sumcheck argument.

Our improved sumcheck argument specifically exploits the algebraic correspondence between univariate and multilinear polynomial commitments underlying Samaritan and Mercury, together with their constant-size multilinear evaluation proofs with linear prover time. We expand upon these requirements and the resulting scope of our proposal in Section 3.2.

[3] Throughout this paper, our notion of a linear-time prover follows the pairing-based cost model where a size-n multiexponentiation or multi-scalar multiplication (MSM), which typically dominates the prover cost, is counted as $O(n)$ work, with implicit dependence on security parameter λ. This differs from hash/code-based linear-prover systems such as Brakedown [GLS+23] and Orion [XZS22], which avoid MSMs.

[4] Throughout this paper, we report proof sizes assuming a BLS12-381 curve-based implementation, unless specified otherwise.

Table 1: Comparison of pairing-based SNARKs with linear-time prover. Here, n denotes the number of R1CS constraints (or the number of gates in a circuit), and $(\mathbb{F}, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ denote the field and groups underlying a bilinear pairing. $\mathbb{P}$ denotes a pairing evaluation. All concrete proof sizes are reported with respect to the BLS12-381 curve, except MicroSpartan [ZSCZ25] and HybridSpartan, which are reported for the BN254 curve. The proof size for Libra depends on the circuit depth d.

SNARK	Prover Time	Verifier Time	Proof Size	$n = 2^{20}$	Setup
Dory [Lee21]	$O(n)\,\mathbb{G}_1$	$O(\log n)\,\mathbb{G}_T$	$O(\log n)\,\mathbb{G}_T$	24KB	Transparent
Kopis [SL20]		$O(\sqrt{n})\,\mathbb{G}_T$	$O(\log n)\,\mathbb{G}_T$	39KB	
Xiphos [SL20]		$O(\log n)\,\mathbb{G}_T$	$O(\log n)\,\mathbb{G}_T$	61KB	
Garuda [DMS24]		$O(\log n)\,\mathbb{P}$	$O(\log n)\,\mathbb{G}_1$	$\approx$ 10KB	Trusted
Spartan [Set20] + KZG [KZG10]		$O(\log^2 n)\,\mathbb{F}$, $O(\log n)\,\mathbb{G}_1$	$O(\log^2 n)\,\mathbb{F}$	234KB	Updatable
Gemini [BCHO22]		$O(\log n)\,\mathbb{G}_1$	$O(\log n)\,\mathbb{G}_1$	18KB	
HyperPlonk [CBBZ23] + PST [PST13]		$O(\log n \log(\log n))\,\mathbb{F}$, $O(\log n)\,\mathbb{P}$	$O(\log n)\,\mathbb{G}_1$	5.5KB	
HyperPlonk + HyperKZG [BCHO22]		$O(\log n \log(\log n))\,\mathbb{F}$, $O(\log n)\,\mathbb{G}_1$	$O(\log n)\,\mathbb{G}_1$	6.1KB	
Libra [XZZ+19]		$O(d \log n)\,\mathbb{F}$, $O(\log n)\,\mathbb{G}_1$	$O(d \log n)\,\mathbb{G}_1$	–	
MicroSpartan [ZSCZ25] (BN254)		$O(\log n)\,\mathbb{G}_1$	$O(\log n)\,\mathbb{G}_1$	5KB	
Samaritan [GPS25]		$O(\log n)\,\mathbb{F}$, $O(1)\,\mathbb{G}_1$	$O(\log n)\,\mathbb{F}$, $O(1)\,\mathbb{G}_1$	6.2KB	
HyperPlonk + Samaritan [GPS25]		$O(\log n \log(\log n))\,\mathbb{F}$, $O(\log n)\,\mathbb{G}_1$	$O(\log n)\,\mathbb{F}$, $O(1)\,\mathbb{G}_1$	4.2KB	
HybridSpartan (this work, BN254)		$O(\log n)\,\mathbb{F}$, $O(\log n)\,\mathbb{G}_1$	$O(\log \log n)\,\mathbb{F}$, $O(1)\,\mathbb{G}_1$	1.9KB	
HybridPlonk (this work)		$O(\log n)\,\mathbb{F}$, $O(\log n)\,\mathbb{G}_1$	$O(\log \log n)\,\mathbb{F}$, $O(1)\,\mathbb{G}_1$	2.4KB	

Sublogarithmic Linear Time SNARKs. We apply our improved sumcheck argument to obtain the *first* linear-time prover SNARKs with sublogarithmic proof sizes supporting either R1CS or Plonkish constraints. In Section 5, we compile recent improvements to the Spartan multilinear PIOP [Set20, ZSCZ25] with our improved sumcheck argument to realize HybridSpartan, that supports R1CS constraints. Next, in Section 6, we compile an optimized variant of the HyperPlonk [CBBZ23] multilinear PIOP with our improved sumcheck argument to realize HybridPlonk, that supports Plonkish constraints.

Both SNARKs feature *updatable setup* and *sublogarithmic proof size* of $O(\log \log n)$, while preserving $O(n)$ prover time and $O(\log n)$ verifier. They also avoid proof recursion and non-black-box use of cryptographic primitives; after applying the Fiat–Shamir transform, their security is established in the combined ROM + AGM setting.

We note that the asymptotic improvements to the proof sizes of HybridSpartan and HybridPlonk are significant at instance sizes of practical interest, without sacrificing prover and verifier efficiency substantially. We compare the proof sizes of HybridSpartan and HybridPlonk with those for Microspartan [ZSCZ25] and HyperPlonk compiled with Samaritan [GPS25], which are the state of the art prover-efficient SNARKs with the smallest proof sizes. For HybridSpartan, the proof size over the BN254 curve is 1.88KB at circuit sizes of 2^{30}. This is $\approx 3.6\times$ smaller than the proof size of Microspartan [ZSCZ25] over BN254 for the same circuit size [5]. For HybridPlonk, the proof size over the BLS12-381 curve is 2.44KB at circuit sizes of 2^{30}. This is $\approx 2.4\times$ smaller than the proof size of HyperPlonk compiled with Samaritan [GPS25] over BLS12-381 for the same circuit size. Prior SNARKs with smaller proof sizes than HybridSpartan and HybridPlonk are based on univariate PCS, and are not prover-efficient as they inherently incur $O(n \log n)$ prover cost due to polynomial multiplications. We present a detailed comparison of HybridSpartan and HybridPlonk with existing SNARKs in Section 1.2 and Table 1.

We remark that the asymptotic reduction in proof sizes does not asymptotically increase either prover or verifier complexity: both HybridSpartan and HybridPlonk retain $O(n)$ prover time and $O(\log n)$ verification while reducing the proof size from $O(\log n)$ to $O(\log \log n)$. Concretely, this compactness comes with a moderate verification-time overhead. For example, in our benchmarks, for $n = 2^{20}$, HybridSpartan has a 1.9KB-sized proof compared with approximately 5KB for MicroSpartan, while verification for HybridSpartan takes approximately 18ms versus 8ms for MicroSpartan. Thus, our constructions are particularly suitable for applications where proof transmission or storage is an important cost component, rather than uniformly lower verification cost. This trade-off can be attractive in applications where proofs must be transmitted or persistently stored, e.g., in blockchain and rollup settings, where proof data itself contributes significantly to the system cost.

[5] The authors of [ZSCZ25] reported proof sizes for Microspartan over the BN254 curve. Hence, we also report proof sizes for HybridSpartan over BN254 for a fair comparison.

Evaluation. We implement HybridSpartan and HybridPlonk (as well as Samaritan PCS from [GPS25]), on top of the popular Arkworks [ac22] ecosystem. Our implementation is available at: `https://github.com/SayaniSinha97/HybridSNARK`. We provide a thorough empirical evaluation of both HybridSpartan and HybridPlonk, and compare them with relevant recent works in Section 7.

Extension to zkSNARK. Zero-knowledge, while a desirable guarantee, is not the primary focus of this work. While we focus on succinctness, it is possible to achieve zero-knowledge variants of HybridSpartan and HybridPlonk following standard techniques (such as [CHM+20, CBBZ23]), albeit with some additional prover cost.

1.2 Comparison with Related Work

Pairing-based Prover-Efficient SNARKs. Table 1 presents a detailed comparison of HybridSpartan and HybridPlonk against known pairing-based prover-efficient SNARKs. HybridSpartan and HybridPlonk are the only ones with sublogarithmic proof sizes, and also achieve the smallest concrete proof sizes.

Non-Pairing-based Prover-Efficient SNARKs. Several works have proposed (plausibly post-quantum) SNARKs with transparent setup and linear prover from hash-based assumptions (e.g., FRI [BBHR18], Orion [XZS22], Brakedown [GLS+23], BaseFold [ZCF24], STIR [ACFY24], WHIR [ACFY25], and Blaze [BCF+25b]) and lattice-based assumptions (e.g., LaBRADOR [BS23], Greyhound [NS24] and [CMNW24]). Asymptotically, all of these constructions incur superlogarithmic proof sizes and verification overheads. In comparison, HybridPlonk incurs sublogarithmic proof size and logarithmic verification overhead (but is not post-quantum due to its reliance on pairings). The concrete proof sizes of known hash-based SNARKs [BBHR18, XZS22, GLS+23, ZCF24, ACFY24, ACFY25, BCF+25b] are greater than 200KB for $n = 2^{20}$ (two orders of magnitude larger than the proof sizes of HybridSpartan and HybridPlonk). The concrete proof sizes of state-of-the-art lattice-based SNARKs [BS23, NS24, CMNW24] are around 50KB for $n = 2^{20}$ (nearly 20× larger than HybridSpartan and HybridPlonk).

SNARKs with SubLogarithmic Proof Size. Several early SNARK constructions [Gro10, Lip12, BCCT12, BCI+13, GGPR13, PHGR13, BCG+13, Lip13, BCTV14] have constant proof size and verification complexity. All of these SNARKs incur at least $O(n \log n)$ prover time. More recently, SNARKs with constant proof size and verification complexity have been achieved by compiling univariate PIOPs using univariate PCS. Some prominent examples are Sonic [MBKM19], Marlin [CHM+20], Plonk [GWC19], and Pari [DMS24]. While these SNARKs have concretely smaller proof sizes and verification time than HybridSpartan and HybridPlonk, they incur significantly larger $O(n \log n)$ prover overheads.

Recent Improvements to Sumcheck. Three recent works [LMN25, ARZR25, BCF+25a], using very different techniques to ours, have described sumcheck PIOPs with sublogarithmic communication. None of the aforementioned works report implementation of their PIOP compiled with a polynomial commitment scheme, so we cannot concretely compare with them. Another relevant approach is Gruen's *univariate skip* technique [Gru24], which skips multiple sumcheck rounds by replacing several multilinear variables with a single higher-degree univariate variable, thereby inducing a different prover–verifier trade-off. In contrast, our goal is to reduce the asymptotic communication of sumcheck while retaining linear prover time and logarithmic verifier time.

In particular, the PIOP in [LMN25] considers an instance of sumcheck over a multivariate polynomial with degree d in each variable, where the prover cost to compute round messages scales as $\mu(d+1)^{\mu/2}$, μ being the number of variables. If the SNARK PIOP involves sumcheck of degree $d \geq 3$, the prover effort becomes superlinear as $(d+1)^{\mu/2} = \Omega(2^{\mu})$. This is true for the multilinear PIOPs for currently known multilinear SNARKs such as Spartan ($d = 3$), Hyperplonk ($d = 5$), etc. There are other plausible challenges in adapting multilinear PCS such as Samaritan [GPS25] or Mercury [EG25] to multivariate polynomials involved in the sumcheck PIOP while retaining linear prover.

Concurrent to our work, two recent works [ARZR25, BCF+25a] propose $O(\log^* n)$ round sumcheck PIOPs with $O(n)$ prover work. While [ARZR25] works specifically over binary fields, [BCF+25a] generalizes to fields of large prime order. The sumcheck PIOP in [BCF+25a] supports arbitrary products of polynomials and with different time and space tradeoffs. The polynomial oracles in [BCF+25a] are also multivariate with $\log^* n$ variables, with different degree bounds. To compile the PIOP into an argument with $O(\log^* n)$ size, the authors present a polynomial commitment scheme for such polynomials by adapting the PST [PST13] scheme. Note that unlike the *structured reference string* (SRS) used by our scheme which consists of monomial encodings, $[x^i]$, the SRS for [BCF+25a], as for [PST13] is structured as multinomial encodings $[x_1^{e_1} \cdots x_k^{e_k}]$. While the former SRS can be updated sequentially using the *power of tau* protocol in [BGM17] and offers 1 *out of N* security against malicious contributions, the latter requires a general multiparty computation involving all the parties for secure generation. In practice, this limits the updatability of the SRS.

2 Preliminaries

This section introduces the notations used throughout and presents some preliminary background material.

Notation. We denote the set of integers $\{1, \ldots, n\}$ by $[n]$ for $n \in \mathbb{N}$, and $\mathbb{F}$ to denote a prime field of order p. We denote by λ a security parameter. We use negl to denote a negligible function: for any integer $c > 0$, there exists $n \in \mathbb{N}$, such that $\forall\ x > n$, $\mathsf{negl}(x) \leq 1/x^c$. We assume a bilinear group generator BG which on input λ outputs parameters for the protocols. Specifically $\mathsf{BG}(1^\lambda)$ outputs $(\mathbb{F}, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, g_t)$ where: $\mathbb{F} = \mathbb{F}_p$ is a prime field of super-polynomial size in λ, with $p = \lambda^{\omega(1)}$; $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ are groups of order p, and e is an efficiently computable non-degenerate bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$; Generators g_1, g_2 are uniformly chosen from $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively and $g_t = e(g_1, g_2)$. We write groups $\mathbb{G}_1$ and $\mathbb{G}_2$ additively, and use the shorthand notation $[x]_1$ and $[x]_2$ to denote group elements $x \cdot g_1$ and $x \cdot g_2$ respectively for $x \in \mathbb{F}$. We implicitly assume that all the setup algorithms for the protocols invoke BG to generate descriptions of groups and fields over which the protocol is instantiated. We will use sets $\mathbb{F}, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ to specify the type of operations, where additionally, we have $\mathbb{P}$ to denote pairings and $\mathbb{M}$ to denote multiexponentiation.

Sets. For $\mu \in \mathbb{N}$, we use B_μ to denote the set $\{0,1\}^\mu$. For $\mathbf{y} = (y_1, \ldots, y_\mu) \in B_\mu$, we use the notation $\mathsf{id}_\mu(\mathbf{y})$ to denote the integer $1 + \sum_{i=1}^{\mu} y_i 2^{i-1}$, and similarly for an integer i, we use $\langle\, i \,\rangle_\mu$ to denote the μ-bit binary decomposition of $i - 1$. We note that the surjective maps $\mathsf{id}_\mu : B_\mu \to [2^\mu]$ and $\langle\, \cdot \,\rangle_\mu : [\,2^\mu\,] \to B_\mu$ are inverses of each other. We will drop the subscript μ when it is clear from the context.

Polynomials and Multilinear Extensions. We use $\mathbb{F}^{\leq 1}[X_1, \ldots, X_\mu]$ to denote the set of μ-variate multilinear polynomials over the field $\mathbb{F}$. We define 2μ-variate polynomial

$$\widetilde{eq}_\mu(\mathbf{x}, \mathbf{y}) = \prod_{i=1}^{\mu} (x_i y_i + (1 - x_i)(1 - y_i))$$

for $\mathbf{x} \in \mathbb{F}^\mu$ and $\mathbf{y} \in \mathbb{F}^\mu$. The polynomials $\{\widetilde{eq}_\mu(x, \langle\, i \,\rangle) : i \in [2^\mu]\}$ are linearly independent over $\mathbb{F}$ and form the Lagrange basis polynomials for the set B_μ. For $\mathbf{x}, \mathbf{y} \in B_\mu$, $\widetilde{eq}_\mu(\mathbf{x}, \mathbf{y}) = 1$ if $\mathbf{x} = \mathbf{y}$, and is 0 otherwise. For a function $f : B_\mu \to \mathbb{F}$, the (unique) polynomial $\widetilde{f}(\mathbf{x}) = \sum_{i=1}^{2^\mu} f(\langle\, i \,\rangle) \widetilde{eq}_\mu(\mathbf{x}, \langle\, i \,\rangle)$ is called the *multilinear extension* (MLE) of the function f.

We also naturally view vectors $\mathbf{f} \in \mathbb{F}^{2^\mu}$ as functions $f : B_\mu \to \mathbb{F}$, and define MLE of the vector as that of the implied function. For clarity of notation, we will denote multilinear polynomials as $\widetilde{f}$ (with a tilde), its associated coefficient vector of evaluations at B_μ as $\mathbf{f}$, and the univariate polynomial with $\mathbf{f}$ as the coefficient vector (in power basis $1, X, \ldots, X^{n-1}$ for $n = 2^\mu$) as $\hat{f}(X)$. Thus, the univariate and multilinear polynomials sharing the coefficient vector $\mathbf{f}$ in the respective bases are denoted as $\hat{f}$ and $\widetilde{f}$ respectively.

2.1 Succinct Argument of Knowledge

Let $\mathcal{R}$ be a NP-relation and $\mathcal{L}$ be the corresponding NP-language, where $\mathcal{L} = \{x : \exists\ w \text{ such that } (x, w) \in \mathcal{R}\}$. Here, a prover $\mathcal{P}$ aims to convince a verifier $\mathcal{V}$ that $x \in \mathcal{L}$ by proving that it knows a witness w for a public statement x such that $(x, w) \in \mathcal{R}$. An interactive argument of knowledge for a relation $\mathcal{R}$ consists of a PPT algorithm Setup that takes as input the security parameter λ, and outputs the public parameters pp, and a pair of interactive PPT algorithms $\langle \mathcal{P}, \mathcal{V} \rangle$, where $\mathcal{P}$ takes as input (pp, x, w) and $\mathcal{V}$ takes as input (pp, x). An interactive argument of knowledge $\langle \mathcal{P}, \mathcal{V} \rangle$ must satisfy completeness and knowledge soundness.

Definition 2.1 (Completeness)**.** *For all security parameter $\lambda \in \mathbb{N}$ and statement x and witness w such that $(x, w) \in \mathcal{R}$, we have*

$$\Pr\left(b = 1 \ : \ \begin{matrix} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ b \leftarrow \langle \mathcal{P}(w), \mathcal{V} \rangle(\mathsf{pp}, x) \end{matrix}\right) = 1.$$

Definition 2.2 (Knowledge Soundness)**.** *For any PPT malicious prover $\mathcal{P}^* = (\mathcal{P}_1^*, \mathcal{P}_2^*)$, there exists a PPT algorithm $\mathcal{E}$ such that the following probability is negligible:*

$$\Pr\left(\begin{matrix} b = 1 \wedge \\ (x, w) \notin \mathcal{R} \end{matrix} \ : \ \begin{matrix} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ (x, \mathsf{st}) \leftarrow \mathcal{P}_1^*(1^\lambda, \mathsf{pp}) \\ b \leftarrow \langle \mathcal{P}_2^*(\mathsf{st}), \mathcal{V} \rangle(\mathsf{pp}, x) \\ w \leftarrow \mathcal{E}^{\mathcal{P}_2^*}(\mathsf{pp}, x) \end{matrix}\right).$$

A *succinct* argument of knowledge $\langle \mathcal{P}, \mathcal{V} \rangle$ for a relation $\mathcal{R}$, must satisfy completeness and knowledge soundness and additionally be *succinct*, that is, the communication complexity between prover and verifier, as well as the verification complexity is bounded by $\mathsf{poly}(\lambda, \log |w|)$.

2.2 Polynomial Commitment Scheme

A polynomial commitment scheme (PCS) introduced in [KZG10] allows a prover to open evaluations of the committed polynomial succinctly. A PCS over $\mathbb{F}$ is a tuple $\mathsf{PC} = (\mathsf{Setup}, \mathsf{Com}, \mathsf{Open}, \mathsf{Eval})$ where:

- $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, n, \{D_i\}_{i \in [n]})$. On input security parameter λ, number of variables n and upper bounds $D_i \in \mathbb{N}$ on the degree of each variable X_i for a n-variate polynomial, Setup generates public parameters pp.
- $(C, \tilde{\mathbf{c}}) \leftarrow \mathsf{Com}(\mathsf{pp}, f(\mathbf{X}), \mathbf{d})$. On input the public parameters pp, and a n-variate polynomial $f(X_1, \cdots, X_n) \in \mathbb{F}[X_1, \ldots, X_n]$ with degree at most $\deg(X_i) = d_i \leq D_i$ for all i, Com outputs a commitment to the polynomial C, and additionally an opening hint $\tilde{\mathbf{c}}$.
- $b \leftarrow \mathsf{Open}(\mathsf{pp}, f(\mathbf{X}), \mathbf{d}, C, \tilde{\mathbf{c}})$. On input the public parameters pp, the commitment C and the opening hint $\tilde{\mathbf{c}}$, a polynomial $f(X_1, \cdots, X_n)$ with $d_i \leq D_i$, Open outputs a bit indicating accept or reject.
- $b \leftarrow \mathsf{Eval}(\mathsf{pp}, C, \mathbf{d}, \mathbf{x}, v; f(\mathbf{X}))$. A public coin interactive protocol

  $$\langle P_{\mathsf{eval}}(f(\mathbf{X})), V_{\mathsf{eval}} \rangle(\mathsf{pp}, C, \mathbf{d}, \mathbf{x}, v)$$

  between a PPT prover and a PPT verifier. The parties have public parameters pp, commitment C, degree d, evaluation point x, and claimed evaluation v as common input. The prover has, in addition, the opening $f(X_1, \cdots, X_n)$ of C, with $\deg(X_i) \leq d_i$. At the end of the protocol, the verifier outputs 1 indicating accepting the proof or outputs 0 indicating rejecting the proof.

A polynomial commitment scheme must satisfy completeness, binding and extractability.

Definition 2.3 (Completeness)**.** *For all polynomials $f(X_1, \cdots, X_n) \in \mathbb{F}[X_1, \ldots, X_n]$ with degree $\deg(X_i) = d_i \leq D_i$, for all $(x_1, \ldots, x_n) \in \mathbb{F}^n$,*

$$\Pr\left[\ b = 1 \ \middle| \ \begin{matrix} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, n, \{D_i\}_{i \in [n]}), \\ (C, \tilde{\mathbf{c}}) \leftarrow \mathsf{Com}(\mathsf{pp}, f(\mathbf{X}), \mathbf{d}), \\ v \leftarrow f(\mathbf{x}), \\ b \leftarrow \mathsf{Eval}(\mathsf{pp}, C, \mathbf{d}, \mathbf{x}, v; f(\mathbf{X})) \end{matrix} \ \right] = 1.$$

Definition 2.4 (Binding)**.** *A polynomial commitment scheme* PC *is binding if for all PPT* $\mathcal{A}$*, the following probability is negligible in* λ*:*

$$\Pr\left(\begin{array}{l|l} \mathsf{Open}(\mathsf{pp}, f_0, \mathbf{d_0}, C, \tilde{\mathbf{c}}_\mathbf{0}) = 1 \wedge & \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, n, \mathbf{D}) \\ \mathsf{Open}(\mathsf{pp}, f_1, \mathbf{d_1}, C, \tilde{\mathbf{c}}_\mathbf{1}) = 1 \wedge & (C, f_0, f_1, \tilde{\mathbf{c}}_0, \\ f_0 \neq f_1 & \quad \tilde{\mathbf{c}}_1, \mathbf{d_0}, \mathbf{d_1}) \leftarrow \mathcal{A}(\mathsf{pp}) \end{array}\right).$$

Definition 2.5 (Knowledge Soundness)**.** *For any PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, there exists a PPT algorithm* $\mathcal{E}$ *such that the following probability is negligible in* λ*:*

$$\Pr\left(\begin{array}{c} b = 1 \wedge \\ \mathcal{R}_{\mathsf{Eval}}(\mathsf{pp}, C, \mathbf{x}, \ : \\ v; \tilde{f}, \tilde{\mathbf{c}}) = 0 \end{array} \middle| \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, n, \{D_i\}_{i\in[n]}) \\ (C, \mathbf{d}, \mathbf{x}, v, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{pp}) \\ (\tilde{f}, \tilde{\mathbf{c}}) \leftarrow \mathcal{E}^{\mathcal{A}_2}(\mathsf{pp}, C, d) \\ b \leftarrow \langle \mathcal{A}_2(\mathsf{st}), V_{\mathsf{eval}} \rangle (\mathsf{pp}, C, \mathbf{d}, \mathbf{x}, v) \end{array}\right).$$

where the relation $\mathcal{R}_{\mathsf{Eval}}$ *is defined as follows:*

$$\mathcal{R}_{\mathsf{Eval}} = \left\{ \left((\mathsf{pp}, C, \mathbf{d}, \mathbf{x} \in \mathbb{F}^n, v \in \mathbb{F}); (f(\mathbf{X}), \widetilde{\mathbf{c}}) \right) : \mathsf{Open}(\mathsf{pp}, f, \mathbf{d}, C, \widetilde{\mathbf{c}}) = 1 \ \wedge \ v = f(\mathbf{x}) \right\}.$$

Fiat-Shamir. An interactive protocol is *public-coin* if the verifier's messages are uniformly random strings. Public-coin protocols can be transformed into non-interactive arguments in the Random Oracle Model (ROM) by using the Fiat-Shamir (FS) [FS87] heuristic to derive the verifier's messages as the output of a Random Oracle. All protocols in this work are public-coin interactive protocols in the structured reference string (SRS) model where both the parties have access to a SRS, that are then compiled into non-interactive arguments using FS.

2.3 The KZG PCS

The KZG univariate PCS was introduced in [KZG10]. We denote the KZG scheme by the tuple of PPT algorithms (KZG.Setup,KZG.Commit, KZG.Prove, KZG.Verify) as defined below.

Definition 2.6 (KZG PCS)**.** *Let* $(\mathbb{F}, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, g_t)$ *be output of bilinear group generator* $\mathsf{BG}(1^\lambda)$*.*

- KZG.Setup *on input* $(1^\lambda, d)$*, where* d *is the degree bound, samples uniform* $\tau \leftarrow \mathbb{F}$ *and outputs*

$$\mathsf{srs} = (\{[\tau]_1, \ldots, [\tau^d]_1\}, \{[\tau]_2, \ldots, [\tau^d]_2\}.$$

- KZG.Commit *on input* $(\mathsf{srs}, p(X))$*, where* $p(X) \in \mathbb{F}_{\leq d}[X]$*, outputs* $C = [p(\tau)]_1$*.*
- KZG.Prove *on input* $(\mathsf{srs}, p(X), \alpha)$*, where* $p(X) \in \mathbb{F}_{\leq d}[X]$ *and* $\alpha \in \mathbb{F}$*, outputs* (v, π) *such that* $v = p(\alpha)$ *and* $\pi = [q(\tau)]_1$*, for* $q(X) = \frac{p(X) - p(\alpha)}{X - \alpha}$*.*
- KZG.Verify *on input* $(\mathsf{srs}, C, v, \alpha, \pi)$*, outputs* 1 *if the following equation holds, and* 0 *otherwise:* $e(C - v[1]_1 + \alpha\pi, [1]_2) \stackrel{?}{=} e(\pi, [\tau]_2)$*.*

2.4 Algebraic Group Model (AGM)

We analyze the security of our protocols in the Algebraic Group Model (AGM) introduced in [FKL18]. An adversary $\mathcal{A}$ is called *algebraic* if every group element output by $\mathcal{A}$ is accompanied by a representation of that group element in terms of all the group elements that $\mathcal{A}$ has seen so far (input and output). In the AGM, an adversary $\mathcal{A}$ is restricted to be *algebraic*, which in our SRS-based protocol means a PPT algorithm satisfying the following: Given $\mathsf{srs} = (\mathsf{srs}_1, \mathsf{srs}_2)$, whenever $\mathcal{A}$ outputs an element $A \in \mathbb{G}_i, i \in 1, 2$, it is accompanied by its representation, i.e, $\mathcal{A}$ also outputs a vector $\mathbf{v}$ over $\mathbb{F}$ such that $A = \langle \mathbf{v}, \mathsf{srs}_i \rangle$.

2.5 The q-DLOG Assumption

Definition 2.7 (q-DLOG Assumption)**.** *The q-DLOG assumption with respect to $\mathcal{G}$ holds if for all λ and for all PPT $\mathcal{A}$, the following probability is negligible in λ:*

$$\Pr\left(\begin{array}{c} \tau = \tau' \\ \tau' \leftarrow \mathcal{A}(1^\lambda, \mathsf{pp}) \end{array} : \begin{array}{c} (\mathbb{F}, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, g_t) \leftarrow \mathsf{BG}(1^\lambda), \\ \tau \leftarrow \mathbb{F}, \\ \mathsf{pp} := ([\tau]_1, [\tau^2]_1, \ldots, [\tau^q]_1, \\ [\tau]_2, [\tau^2]_2, \ldots, [\tau^q]_2) \end{array}\right)$$

The KZG univariate PCS is shown to be evaluation binding assuming the hardness of q-DLOG (Definition 2.7) and knowledge sound in the Algebraic Group Model (AGM). At a high level, AGM [FKL18] considers *algebraic* adversaries that are algorithms $\mathcal{A}$ such that every group element output by $\mathcal{A}$ is accompanied by a representation of that group element in terms of all the group elements that $\mathcal{A}$ has seen so far (input and output).

2.6 Polynomial IOP

A modular approach for designing efficient succinct arguments is to (i) first, construct an Interactive Oracle Proof (IOP) that is an information-theoretic protocol in an idealized model, (ii) compile the information-theoretic protocol via a cryptographic compiler to obtain a cryptographic argument system. In a Polynomial IOP (PIOP), the prover provides oracle access to a set of polynomials, and the verifier accepts or rejects by checking certain identities over the polynomials output by the prover and possibly public polynomials known to the verifier. A PIOP is compiled into a succinct argument of knowledge by using a *polynomial commitment scheme* to realize the polynomial oracles. Many recent constructions of zkSNARKs [BFS20, CHM$^+$20, GWC19] follow this approach where the information theoretic object is a PIOP and the cryptographic compiler is a polynomial commitment scheme.

Definition 2.8 (Polynomial IOP)**.** *A polynomial IOP is a public-coin interactive proof for a relation $\mathcal{R} = (\mathbb{x}, \mathbb{w})$. $\mathcal{R}$ is an oracle relation such that $\mathbb{x}$ consists of* oracles *to μ-variate polynomials over $\mathbb{F}$. These oracles can be queried at arbitrary points in $\mathbb{F}^\mu$ to evaluate the polynomial at these points. In every round in the protocol, the prover sends multi-variate polynomial oracles. The verifier in every round sends a random challenge. At the end of the protocol, the verifier (with oracle access to all the polynomial oracles sent so far) and given its own randomness outputs accept/reject. A PIOP satisfies completeness and knowledge-soundness.*

We use the following results about PIOPs and their compilation.

Lemma 2.1 ([CHM$^+$20, BFS20, CBBZ23])**.** *If a PIOP is sound for an oracle relation $\mathcal{R}$ with soundness error δ, then it is knowledge sound for $\mathcal{R}$ with knowledge error δ and the extractor running in time polynomial in the witness size.*

Indexed Relations. An indexed relation $\mathcal{R}$ is a set of triples $(\mathbb{i}, \mathbb{x}, \mathbb{w})$ where $\mathbb{i}$ is the index, $\mathbb{x}$ is the instance, and $\mathbb{w}$ is the witness. Typically, an NP relation $((C, \mathbb{x}); \mathbb{w})$ given by $C(\mathbb{x}, \mathbb{w}) = 1$ for an arithmetic circuit C is expressed as an indexed relation with index $\mathbb{i}$ encoding the description of the circuit C. A PIOP for indexed relation $\mathcal{R}$ is specified by indexer $\mathcal{I}$, prover $\mathcal{P}$ and verifier $\mathcal{V}$ which operate as follows:

- *Offline Phase*: The indexer $\mathcal{I}$ receives the index $\mathbb{i}$ as input and outputs a set of polynomials encoding $\mathbb{i}$.
- *Online Phase*: The prover $\mathcal{P}(\mathbb{i}, \mathbb{x}, \mathbb{w})$ interacts with the verifier $\mathcal{V}([\![\mathbb{i}]\!], \mathbb{x})$, i.e, the verifier has oracle access to the polynomials output by $\mathcal{I}$ in addition to any oracles in the statement $\mathbb{x}$.

Lemma 2.2 ([CHM$^+$20, BFS20, CBBZ23])**.** *Let $\mathcal{R}$ be a relation over $\mathbb{F}$. Let $PIOP = (\mathcal{I}, \mathcal{P}, \mathcal{V})$ be a PIOP over $\mathbb{F}$ for $\mathcal{R}$ with negligible soundness error, and let*

$$\mathsf{PC} = (\mathsf{Setup}, \mathsf{Com}, \mathsf{Open}, \mathsf{Eval})$$

be a polynomial commitment scheme over $\mathbb{F}$ that satisfies completeness, binding and extractability. Then there exists a compiler that compiles the PIOP using PC to obtain a public-coin argument of knowledge $\Pi = (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ for $\mathcal{R}$. If the PIOP is for an indexed relation, the resulting argument system is a preprocessing argument system.

This succinct argument system with a public-coin verifier is finally transformed into a SNARK via Fiat-Shamir.

3 Background and Technical Overview

In this section, we present a high-level overview of our approach. We begin by introducing some background material that is necessary to understand the overview.

3.1 Background

Multivariate Sumcheck. Let $f(X_1, \ldots, X_\mu) \in \mathbb{F}[X_1, \ldots, X_\mu]$. Consider the claim:

$$\sum_{x_1 \in \{0,1\}} \sum_{x_2 \in \{0,1\}} \cdots \sum_{x_\mu \in \{0,1\}} f(x_1, \ldots, x_\mu) = y.$$

This claim takes time $O(|B_\mu|)$ to verify. The sumcheck protocol [LFKN90] allows a verifier $\mathcal{V}$ to outsource this computation to a prover $\mathcal{P}$, where the prover $\mathcal{P}$ sends a number of field elements that is logarithmic in the size of the hypercube, and the verifier $\mathcal{V}$ needs to evaluate f at a single point. We recall the classical sumcheck protocol from [LFKN90] for multilinear polynomials.

- **Common Input**: $\mathcal{V}$ is given oracle access $[\![\widetilde{f}]\!]$ to the multilinear polynomial $\widetilde{f} \in \mathbb{F}^{\leq 1}[X_1, \ldots, X_\mu]$ and the claimed sum $v = \sum_{\mathbf{x} \in B_\mu} f(\mathbf{x})$. The prover and verifier interact in μ rounds $1, \ldots, \mu$ as:
- As its i^{th} message, $\mathcal{P}$ sends univariate polynomial $g_i(X_i)$ such that $g_i(X_i) = \sum_{\mathbf{x}' \in B_{\mu-i}} f(\mathbf{r}, X_i, \mathbf{x}')$, where $\mathbf{r} = (r_1, \ldots, r_{i-1})$ is the vector of challenges sent by the verifier in the previous rounds. Note that $\mathbf{r}$ is not defined for $i = 1$, and so we ignore it for $i = 1$.
- $\mathcal{V}$ checks $g_i(1) + g_i(0) = g_{i-1}(r_i)$ for $i > 1$ and $g_1(1) + g_1(0) = v$ for $i = 1$. The verifier aborts if the check fails.
- If $i < \mu$, $\mathcal{V}$ sends $r_i \leftarrow \mathbb{F}$ to the prover.
- For $i = \mu$, $\mathcal{V}$ samples $r_\mu \leftarrow \mathbb{F}$ and checks $g_\mu(r_\mu) = \widetilde{f}(r_1, \ldots, r_\mu)$ by querying the oracle $[\![f]\!]$.

The above protocol has communication complexity $O(\mu)$, with the prover $\mathcal{P}$ computing $O(2^\mu)$ $\mathbb{F}$-operations.

Multilinear PCS with Constant Proof Size. We recall recent constructions [GPS25, EG25] of multilinear polynomial commitment schemes with constant proof size. The constructions use the following isomorphism between μ-variate multilinear polynomials and univariate polynomials $\mathbb{F}[X]$ of degree $< n$ for $n = 2^\mu$. Define the isomorphism of the $\mathbb{F}$-vector spaces as:

$$\varphi : \mathbb{F}^{\leq 1}[X_1, \ldots, X_\mu] \longrightarrow \mathbb{F}^{<n}[X]$$
$$\sum_{i=1}^{n} f_i \widetilde{eq}_\mu((X_1, \ldots, X_\mu), \langle\, i \,\rangle) \mapsto \sum_{i=1}^{n} f_i X^{i-1}.$$

For $\widetilde{f} \in \mathbb{F}^{\leq 1}[X_1, \ldots, X_\mu]$, we use $\hat{f}$ to denote the univariate polynomial $\varphi(\widetilde{f})$. We associate a multilinear polynomial $\widetilde{f}$ with the univariate polynomial $\hat{f}$ according to the above isomorphism. Subsequently, we treat

the KZG commitment C to the polynomial $\hat{f}$ as the commitment to the multilinear polynomial $\widetilde{f}$. The constructions in [GPS25, EG25] show elegant protocols to prove evaluations $\widetilde{f}(\mathbf{z}) = v$ for $\mathbf{z} \in \mathbb{F}^\mu$ and $v \in \mathbb{F}$ with constant proof size and $O_\lambda(n)$ prover cost. Specifically, we will use the Samaritan from [GPS25] which achieves constant proof size of 368 bytes over the BLS12-381 curve.

3.2 Improved Sumcheck Argument with Sublogarithmic Rounds

In this subsection, we present an overview of our main technical contribution, namely, an improved sumcheck argument with sublogarithmic communication and linear prover. For the sake of illustration, consider the claim $\sum_{\mathbf{x} \in B_\mu} \widetilde{a}(\mathbf{x})\widetilde{b}(\mathbf{x}) = v$ for μ-variate multilinear polynomials $\widetilde{a}$ and $\widetilde{b}$. Let $n = 2^\mu$. Using the sumcheck argument described in Section 3.1, the prover can prove the above claim in $O(\log n)$ rounds of communication while doing $O(n)$ work. We ask:

Can we obtain a sumcheck argument with fewer communication rounds?

Our Approach. A possible approach is to use the *univariate sumcheck* protocol introduced in [BCR+19], where instead of considering multilinear polynomial evaluations over the hypercube, we consider univariate polynomial evaluations over a multiplicative subgroup of $\mathbb{F}$ of size n. Specifically, letting $\mathbb{H} = \{1, \omega, \ldots, \omega^{n-1}\}$ be the multiplicative subgroup of $\mathbb{F}$ of size n with $\omega \in \mathbb{F}$ as a generator, we define univariate polynomials $\hat{a}, \hat{b} \in \mathbb{F}^{<n}[X]$ such that $\hat{a}(\omega^{i-1}) = \widetilde{a}(\langle\, i \,\rangle)$ and $\hat{b}(\omega^{i-1}) = \widetilde{b}(\langle\, i \,\rangle)$. Then, the equivalent claim to prove is $\sum_{h \in \mathbb{H}} \hat{a}(h)\hat{b}(h) = v$. The univariate sumcheck protocol [BCR+19], when used with the KZG PCS allows the identity to be checked using $O(1)$ communication. However, the prover work is $O(n \log n)$ due to the need to compute polynomial product $\hat{a}(X) \cdot \hat{b}(X)$. The next question we ask is:

Can we reduce communication while retaining $O(n)$ prover work?

We give an affirmative answer to this question, achieving communication of $O(\log \log n)$ with linear prover effort. We achieve this by using a hybrid of the two sumcheck techniques. At a high level, our approach is as follows:

- We initially proceed with the multilinear sumcheck protocol for $O(\log \log n)$ rounds, at the end of which we are left with a claim over hypercube of size $m = n/\log n$, involving multilinear polynomials of size m.
- By identifying the multilinear polynomials of size m with univariate polynomials of degree $< m$, we derive an equivalent univariate sumcheck claim.
- Using the univariate sumcheck protocol, and the KZG PCS, we can then prove the claim with $O(1)$ communication and $O(m \log m)$ prover work, which is $O(n)$ since $m = n/\log n$.

Challenges. While the above approach gives us a template for a sumcheck argument in $O(\log \log n)$ rounds, we run into two main challenges. First, we need to prove that the multilinear polynomials of size $m = n/\log n$ are suitable restrictions of the original multilinear polynomials of size n. Additionally, we need to prove that the polynomials involved in the univariate sumcheck are consistent with the multilinear polynomials in the reduced sumcheck instance. This, in turn, requires proving that a pair of polynomials share the same coefficients in the power and Lagrange bases.

Using Multilinear PCS with Special Properties. We address both challenges by crucially leveraging the following key properties of Samaritan [GPS25] and Mercury [EG25]:

- Both schemes provide a "dual commitment": under the isomorphism φ defined in Section 3.1, the same commitment simultaneously serves a commitment to a univariate polynomial $\hat{f}$ and the corresponding

multilinear polynomial $\widetilde{f} = \varphi^{-1}(\hat{f})$. In particular, our construction exploits the specific algebraic correspondence between the univariate and multilinear representations underlying these schemes, rather than treating the multilinear PCS as a generic black box.

- Both schemes support $O(1)$-sized multilinear evaluation proofs with linear prover time.

The first property allows us to efficiently link the univariate and multilinear representations without introducing separate commitments, while the second allows us to establish the requisite consistency checks without compromising our target communication and prover complexities. Consequently, the treatment in the rest of the paper is specific to Samaritan and Mercury, rather than an abstract multilinear PCS satisfying a generic interface.

To the best of our knowledge, CHOPIN [BHKM26], which appeared recently, is the only other multilinear PCS satisfying similar properties. We leave investigating whether its algebraic structure is compatible with our design template as an interesting direction for future work.

4 Sublogarithmic Multivariate Sumcheck Argument

We now formalize the hybrid sumcheck approach outlined in Section 3.2. We begin by formally describing the multilinear sumcheck relation for which we propose an interactive argument.

The Multivariate Sumcheck Relation. Let $\widetilde{G}(X_1, \ldots, X_\ell)$ be a multivariate polynomial of degree d over a field $\mathbb{F}$. Let $\widetilde{f}_1, \ldots, \widetilde{f}_\ell$ be μ-variate multilinear polynomials over $\mathbb{F}$. Let $v \in \mathbb{F}$ be a value. Let B_μ denote the hypercube of size $n = 2^\mu$. Let R be a multivariate sumcheck relation (parameterized by n, d, ℓ) defined with respect to a statement $\mathsf{st} = (\widetilde{\mathbf{G}}, v, B_\mu)$ and a witness $\mathsf{wit} = \{\widetilde{f}_i\}_{i\in[\ell]}$ such that :

$$R\left((\widetilde{\mathbf{G}}, v, B_\mu); \{\widetilde{f}_i\}_{i\in[\ell]}\right) = 1 \quad \text{if} \quad \sum_{\mathbf{x}\in B_\mu} \widetilde{G}(\widetilde{f}_1(\mathbf{x}), \ldots, \widetilde{f}_\ell(\mathbf{x})) = v. \tag{1}$$

Throughout, for simplicity of exposition, we write $\rho = \log\log n$, $\kappa = \mu - \rho$, and $m = 2^\kappa$, implicitly rounding ρ to an integer. More precisely, one may set $\rho = \lceil \log \mu \rceil$, in which case $m = \Theta(n/\log n)$; this does not affect any of our asymptotic bounds.

End-to-End Sumcheck Argument. Figure 1 describes the interactive protocol describing our sumcheck argument for the relation R in Equation (1) with sublogarithmic communication and linear prover. Throughout, we use uPCS and mPCS to denote a homomorphic univariate PCS and a homomorphic multilinear PCS, respectively. For concrete instantiations, $(\mathsf{uPCS}, \mathsf{mPCS})$ either refers to the pair (KZG, Samaritan) or to the pair (KZG, Mercury). Figure 1 builds upon the outline in Section 3.2, adapted to the generalized sumcheck instance in Equation (1). Along the way we also incorporate certain optimizations aimed to reduce the concrete communication cost. The protocol is divided into four phases:

- **Phase-0: Classical sumcheck.** Initially, the prover $\mathcal{P}$ and the verifier $\mathcal{V}$ proceed with the classical multivariate sumcheck protocol (described in Section 3.1) for $\rho = \log\log n$ rounds, at the end of which they are left with a claim over hypercube of size $m = n/\log n$, involving κ-variate multilinear polynomials of size m.
- **Phase-1: Univariate sumcheck.** $\mathcal{P}$ and $\mathcal{V}$ establish the reduced sumcheck claim using a univariate PIOP over a domain of size $m = n/\log n$. This phase reduces verification of the sum to checking evaluations of the univariate polynomials $\hat{q}_i$ at a random point.
- **Phase-2: Consistency of multilinear restrictions.** Next, $\mathcal{P}$ and $\mathcal{V}$ establish that the multilinear polynomials underlying the reduced instance are consistent with the original polynomials after fixing the challenges from the first $\rho = \log\log n$ rounds of multivariate sumcheck. The checks for the ℓ constituent polynomials are batched using a random linear combination.

Let $n = 2^\mu$, $\rho = \log \mu$, $\kappa = \mu - \rho$, $m = 2^\kappa$. We present our sumcheck argument as an interactive protocol below, where the prover $\mathcal{P}$ proves to the verifier $\mathcal{V}$ the following multivariate sumcheck relation:

$$\sum_{\mathbf{x} \in B_\mu} \widetilde{G}(\widetilde{f}_1(\mathbf{x}), \ldots, \widetilde{f}_\ell(\mathbf{x})) = v.$$

Throughout, we use uPCS and mPCS to denote a homomorphic univariate PCS and a homomorphic multilinear PCS, respectively with public parameter pp, as per the definition of PCS in Section 2.2. For concrete instantiations, $(\mathsf{uPCS}, \mathsf{mPCS})$ either refers to the pair (KZG, Samaritan) or to the pair (KZG, Mercury).

Phase-0: Classical Sumcheck

1: $\mathcal{P}$ sends $(C_i, \tilde{\mathbf{c}}_i) = \mathsf{mPCS.Com}(\mathsf{pp}, \widetilde{f}_i(\cdot), \mu), \forall i \in [\ell]$.
2: $\mathcal{P}$ and $\mathcal{V}$ interact for $\rho = \log \log n$ rounds of classical multivariate sumcheck such that the following relation is left to be proved, for $\mathbf{r} = (r_1, \ldots, r_\rho) \in \mathbb{F}^\rho$: $\sum_{\mathbf{x} \in B_\kappa} \widetilde{G}(\widetilde{f}_1(\mathbf{r}, \mathbf{x}), \ldots, \widetilde{f}_\ell(\mathbf{r}, \mathbf{x})) = \bar{v}$.

Phase-1: Univariate PIOP

3: Let $\widetilde{p}_i(\cdot) = \widetilde{f}_i(\mathbf{r}, \cdot)\ \forall i \in [\ell]$.
4: $\mathcal{P}$ computes $\hat{q}_i(X) = \mathsf{ifft}(\hat{p}_i(X))\ \forall i \in [\ell]$.
5: $\mathcal{P}$ computes $\hat{g}(X)$ and $\hat{h}(X)$, with $\deg(\hat{h}) \leq m - 2$ such that $\hat{G}(X) = (X^m - 1)\hat{g}(X) + X\hat{h}(X) + \bar{v}/m$.
6: $\mathcal{P}$ sends $(C_g, \tilde{\mathbf{c}}_g) = \mathsf{uPCS.Com}(\mathsf{pp}, \hat{g}(X), \deg(\hat{g}))$ and $(C_h, \tilde{\mathbf{c}}_h) = \mathsf{uPCS.Com}(\mathsf{pp}, \hat{h}(X), \deg(\hat{h}))$.
7: $\mathcal{V}$ sends $z \leftarrow \mathbb{F}$.
8: $\mathcal{P}$ sends the values $v_i = \hat{q}_i(z)$, $\forall i \in [\ell]$.
9: $\mathcal{V}$ computes $\hat{G}(z) = \widetilde{G}(v_1, \cdots, v_\ell)$, and assuming $\hat{a}(X) = (z^m - 1)\hat{g}(X) + z\hat{h}(X)$, $\mathcal{V}$ homorphically computes the commitment to $\hat{a}(X)$ as $C'_1 = (z^m - 1) \cdot C_g + z \cdot C_h$.

Phase-2: Prove Consistency between $\widetilde{p}_i(\cdot)$ and $\widetilde{f}_i(\mathbf{r}, \cdot)\ \forall i \in [\ell]$

10: $\mathcal{V}$ sends $\gamma \leftarrow \mathbb{F}$.
11: $\mathcal{P}$ computes $\hat{p}(X) = \sum_{i=1}^{\ell} \gamma^{i-1} \hat{p}_i(X)$ and $\hat{p}_{\mathsf{ext}}(X)$ as described in Section 4.
12: $\mathcal{P}$ sends $(C_p, \tilde{\mathbf{c}}_p) = \mathsf{uPCS.Com}(\mathsf{pp}, \hat{p}(X), m)$ and $(C_{p_{\mathsf{ext}}}, \tilde{\mathbf{c}}_{p_{\mathsf{ext}}}) = \mathsf{mPCS.Com}(\mathsf{pp}, \widetilde{p}_{\mathsf{ext}}(\cdot), \mu) = \mathsf{uPCS.Com}(\mathsf{pp}, \hat{p}_{\mathsf{ext}}(X), n)$.
13: $\mathcal{V}$ sends $\mathbf{y} \leftarrow \mathbb{F}^\kappa$.
14: $\mathcal{V}$ computes combined mPCS commitment C_f for $\widetilde{f}(\cdot) = \sum_{i=1}^{\ell} \gamma^{i-1} \widetilde{f}_i(\cdot)$ as $C_f = \sum_{i=1}^{\ell} \gamma^{i-1} C_i$ and then computes $C'_2 = C_{p_{\mathsf{ext}}} - C_f$.
15: $\mathcal{V}$ sends $\eta \leftarrow \mathbb{F}$.
16: $\mathcal{P}$ sends $v_\eta = \hat{p}(\eta^s)$.

Phase-3: Prove evaluations $v_i = \hat{q}_i(z)\ \forall i \in [\ell]$

17: $\mathcal{P}$ computes $\hat{A}_z(X)$ as described in Equation (10) (Section 4.3), and sends $(C_A, \tilde{\mathbf{c}}_A) = \mathsf{uPCS.Com}(\mathsf{pp}, \hat{A}_z(X), \deg(\hat{A}_z))$.
18: $\mathcal{V}$ sends $\delta, \epsilon \leftarrow \mathbb{F}$.
19: $\mathcal{P}$ computes $\hat{S}(X)$ satisfying Equation (13) (described in Section 4.3), and sends $(C_S, \tilde{\mathbf{c}}_S) = \mathsf{uPCS.Com}(\mathsf{pp}, \hat{S}(X), \deg(\hat{S}))$.
20: $\mathcal{V}$ sends $r \leftarrow \mathbb{F}^*$.
21: $\mathcal{P}$ sends the values $v_{\mathsf{Ainv}} = \hat{A}_z(1/r)$, $v_{\mathsf{pinv}} = \hat{p}(1/r)$, $v_{\mathsf{Sinv}} = \hat{S}(1/r)$.
22: Let $\hat{B}_z(X)$ be the "inverse-coefficient" polynomial of $\hat{A}_z(X)$ which is easily evaluated by $\mathcal{V}$ (see Equation (11) for details).
23: $\mathcal{P}$ computes $v_{\mathsf{Binv}} = \hat{B}_z(\delta/r)$.
24: $\mathcal{V}$ sends $r' \leftarrow \mathbb{F}$.
25: Let $\hat{b}(X) = v_{\mathsf{Ainv}} \cdot \hat{p}(X) + (v_{\mathsf{pinv}} + \epsilon \cdot v_{\mathsf{Binv}}) \cdot \hat{A}_z(X) - r \cdot \hat{S}(X)$, and $\hat{c}(X) = \hat{p}(X) + r' \cdot \hat{A}_z(X) + r'^2 \cdot \hat{S}(X)$.
26: $\mathcal{V}$ computes $v' = \sum_{i=1}^{\ell} \gamma^{i-1} v_i$, $v_{\mathsf{Binv}} = \hat{B}_z(\delta/r)$ and $v_{\mathsf{B}} = \hat{B}_z(\delta r)$.
27: $\mathcal{V}$ computes $C'_3 = v_{\mathsf{Ainv}} \cdot C_p + (v_{\mathsf{pinv}} + \epsilon \cdot v_{\mathsf{Binv}}) \cdot C_A - r \cdot C_S$.
28: $\mathcal{V}$ computes $C'_4 = C_p + r' \cdot C_A + r'^2 \cdot C_S$.

Verifier Checks

29: $\mathsf{uPCS.Eval}(\mathsf{pp}, C'_1, \deg(\hat{a}), z, \big(\hat{G}(z) - \bar{v}/m\big); \hat{a}(X)) = 1$
30: $\mathsf{mPCS.Eval}(\mathsf{pp}, C'_2, \mu, (\mathbf{r}, \mathbf{y}), 0_{\mathbb{F}}; \widetilde{p}_{\mathsf{ext}}(\cdot) - \widetilde{f}(\cdot)) = 1$
31: $\mathsf{uPCS.Eval}(\mathsf{pp}, C_{p_{\mathsf{ext}}}, n, \eta, v_\eta \cdot \frac{\eta^s - 1}{\eta - 1}; \hat{p}_{\mathsf{ext}}(X)) = 1$
32: $\mathsf{uPCS.Eval}(\mathsf{pp}, C_p, m, \eta^s, v_\eta; \hat{p}(X)) = 1$
33: $\mathsf{uPCS.Eval}(\mathsf{pp}, C'_3, \deg(\hat{b}), r, 2\big(\frac{mv'}{z^m - 1} + \epsilon \cdot \frac{\delta^m - 1}{\delta - 1}\big) + (1/r) \cdot v_{\mathsf{Sinv}} - \epsilon \cdot v_{\mathsf{Ainv}} \cdot v_{\mathsf{B}}; \hat{b}(X)) = 1$
34: $\mathsf{uPCS.Eval}(\mathsf{pp}, C'_4, \deg(\hat{c}), 1/r, v_{\mathsf{pinv}} + r' \cdot v_{\mathsf{Ainv}} + r'^2 \cdot v_{\mathsf{Sinv}}; \hat{c}(X)) = 1$
35: $\mathcal{V}$ checks $\deg(\hat{h}(X)) \leq (m - 2)$ with a pairing check in the presence of C_h.

Figure 1: End-to-End Protocol for Our Sumcheck Argument with Sublogarithmic Communication and Linear Prover

- **Phase-3: Consistency of univariate evaluations.** Finally, $\mathcal{P}$ and $\mathcal{V}$ establish that the evaluations of the univariate polynomials $\hat{q}_i$ used in Phase-1 are consistent with the corresponding restricted multilinear polynomials authenticated in Phase-2. Observe that this completes the linkage between the original multivariate sumcheck instance and the univariate sumcheck claim.

In the subsequent subsections, we describe each of the phases 1 through 3 in detail. To improve readability, Figure 1 stacks the verifier checks at the end of the protocol description; these checks should be understood as being performed at the appropriate points after the requisite prover messages and verifier challenges have been generated.

4.1 Phase-1: Univariate sumcheck

As discussed above, to prove the claim in Equation (1), in phase 0, the prover and the verifier proceed with $\rho = \log\log n$ rounds of the classical multivariate sumcheck protocol described in Section 3.1. Assuming $\mathbf{r} = (r_1, \ldots, r_\rho)$ is the set of challenges sent by the verifier so far, the reduced claim is of the form

$$\sum_{\mathbf{x} \in B_\kappa} \widetilde{G}(\widetilde{f}_1(\mathbf{r}, \mathbf{x}), \ldots, \widetilde{f}_\ell(\mathbf{r}, \mathbf{x})) = \bar{v}, \tag{2}$$

for some $\bar{v} \in \mathbb{F}$. At this stage, suppose that the prover commits to the κ-variate multilinear polynomials $\widetilde{p}_i, i \in [\ell]$ defined by $\widetilde{p}_i(\mathbf{x}) = \widetilde{f}_i(\mathbf{r}, \mathbf{x})$ for $\mathbf{x} \in B_\kappa$. Let $\hat{p}_1(X), \ldots, \hat{p}_\ell(X)$ denote the corresponding univariate polynomials of degree less than m, such that j^{th} coefficient of $\hat{p}_i$ is $\widetilde{p}_i(\langle\, j \,\rangle)$, $\forall i \in [\ell], j \in [m]$.

Using a Univariate PIOP. Our idea is to verify the reduced claim in Phase-1 using a univariate PIOP over the much smaller domain of size m. Specifically, we choose the domain to be the set $\mathbb{H}$ consisting of the m roots of unity $\{1, \omega, \ldots, \omega^{m-1}\}$. To this end, the prover computes polynomials $\hat{q}_i(X)$, $i \in [\ell]$ such that evaluations of $\hat{q}_i$ over the set $\mathbb{H}$ match the monomial coefficients of polynomial $\hat{p}_i$. Let $\{\mu_k^{\mathbb{H}}(X)\}_{k=1}^m$ denote the Lagrange basis polynomials for the set $\mathbb{H}$ satisfying $\mu_k^{\mathbb{H}}(\omega^{j-1}) = \delta_{kj}\ \forall j \in [m]$, where $\delta_{kj} = 1$ if $k = j$, and $\delta_{kj} = 0$ otherwise. We define linear operator ifft on the set $\mathbb{F}^{<m}[X]$ as:

$$\mathsf{ifft}\left(\sum_{i=1}^{m} a_i X^{i-1}\right) = \sum_{i=1}^{m} a_i \mu_i^{\mathbb{H}}(X). \tag{3}$$

In terms of the above operator, we have $\hat{q}_i = \mathsf{ifft}(\hat{p}_i)$ for all $i \in [\ell]$. Assuming that the polynomials $\hat{p}_i$ and $\hat{q}_i$ are constructed honestly, we can reduce the sumcheck claim in Equation (2) to a univariate sumcheck claim as below:

$$\begin{aligned}\sum_{\mathbf{x} \in B_\kappa} \widetilde{G}(\widetilde{p}_1(\mathbf{x}), \ldots, \widetilde{p}_\ell(\mathbf{x})) = \bar{v} &\Leftrightarrow \sum_{i \in [m]} \widetilde{G}(\widetilde{p}_1(\langle\, i \,\rangle), \ldots, \widetilde{p}_\ell(\langle\, i \,\rangle)) = \bar{v} \\ &\Leftrightarrow \sum_{i \in [m]} \widetilde{G}(\hat{q}_1(\omega^{i-1}), \ldots, \hat{q}_\ell(\omega^{i-1})) = \bar{v}. \end{aligned} \tag{4}$$

The final claim uses the fact that $\hat{q}_i(\omega^{j-1}) = \widetilde{p}_i(\langle\, j \,\rangle)$ for all $i \in [\ell]$ and $j \in [m]$. Let $\hat{G}(X) = \widetilde{G}(\hat{q}_1(X), \ldots, \hat{q}_\ell(X))$. Then the final claim is essentially equivalent to the claim $\sum_{i \in [m]} \hat{G}(\omega^{i-1}) = \bar{v}$. Now, we can use the *univariate sumcheck* to establish this claim. As shown in [BCR+19], it requires the prover to send polynomial oracles $\hat{g}$ and $\hat{h}$, with $\deg(\hat{h}) \leq m - 2$ such that:

$$\hat{G}(X) = (X^m - 1) \cdot \hat{g}(X) + X \cdot \hat{h}(X) + \bar{v}/m. \tag{5}$$

The prover commits to polynomials $\hat{g}(X)$ and $\hat{h}(X)$. To check the identity, the verifier queries the polynomials $\hat{q}_i, i \in [\ell]$, $\hat{g}$, and $\hat{h}$ at a random point and enforces the degree bound on $\hat{h}$. The above polynomials can be computed and committed in $O(\mathcal{C}(\widetilde{G}, d, \ell) \cdot m \log m) = O(\mathcal{C}(\widetilde{G}, d, \ell) \cdot n)$ $\mathbb{F}$ operations and $O(dm + \ell m) = O((d + \ell)n/\log n)$ $\mathbb{G}_1$ operations, where $\mathcal{C}(\widetilde{G}, d, \ell)$ is a constant dependent on the multivariate form $\widetilde{G}$, d

and ℓ. Note that most $\mathbb{F}$-operations are incurred in computing polynomial products corresponding to high degree terms in $\widetilde{G}$. In a typical application we expect d to be a small constant (e.g., $d = 5$ for HyperPlonk) and thus computing each term of $\widetilde{G}$ takes $\approx O(m \log m)$ operations. The entire computation of $\hat{G}$ then takes $O(||\widetilde{G}||m \log m)$ operations, which is $O(||\widetilde{G}||n)$ where $||\widetilde{G}||$ denotes the number of monomial terms in $\widetilde{G}$.

Note that in the univariate PIOP in Equation (5), the query to $\hat{G}$ at point $z \leftarrow \mathbb{F}$ is simulated using queries to $\hat{q}_i$, $i \in [\ell]$ as:

$$\hat{G}(z) = G(\hat{q}_1(z), \ldots, \hat{q}_\ell(z))$$

Summary and Optimizations. We now summarize how the above idea is translated into Phase-1 (lines 3-9) of Figure 1, albeit with some additional optimizations to reduce communication. Recall that, after the completion of first $\rho = \log \log n$ rounds of classical multivariate sumcheck protocol in Phase-0, the prover $\mathcal{P}$ holds ℓ multilinear polynomials of κ variables, denoted with $\widetilde{p}_i(\cdot) = \widetilde{f}_i(\mathbf{r}, \cdot)\ \forall i \in [\ell]$, and the verifier $\mathcal{V}$ holds a value $\bar{v} \in \mathbb{F}$. The challenges of $\mathcal{V}$ for the first ρ rounds are denoted together with $\mathbf{r} \in \mathbb{F}^\rho$. Each $\widetilde{p}_i(\cdot)$ is also viewed as $\hat{p}_i(X) = \sum_{j=1}^{m} \hat{p}_{i,j} X^{j-1}$ such that $\hat{p}_{i,j} = \widetilde{p}_i(\langle\, j \,\rangle)$.

At this point, $\mathcal{P}$ first computes $\hat{q}_i(X) = \mathsf{ifft}(\hat{p}_i(X))\ \forall i \in [\ell]$. The goal of the univariate PIOP in Phase-1 of Figure 1 is to check the following claim:

$$\hat{G}(\hat{q}_1(X), \ldots, \hat{q}_\ell(X)) = (X^m - 1)\hat{g}(X) + X\hat{h}(X) + \bar{v}/m.$$

where polynomials $\hat{q}_i$'s are as defined earlier, and $\hat{g}$, $\hat{h}$ are auxiliary polynomials such that $\deg(h) < m - 1$. To this end, $\mathcal{P}$ sends commitments C_g and C_h to polynomials $\hat{g}$ and $\hat{h}$ respectively. Subsequently, $\mathcal{V}$ sends $z \leftarrow \mathbb{F}$ as the evaluation query to the polynomials. The prover then responds with purported evaluations $v_i = \hat{q}_i(z)$, $i \in [\ell]$. In Figure 1, we frequently request polynomial evaluations in a *staggered* manner to optimize the number of evaluation proofs. This optimization, often referred to as *linearization* trick, is commonly employed in univariate PIOPs in the AGM.

4.2 Phase-2: Consistency of Multilinear Restrictions

In this phase, the prover establishes to the verifier the correct *linkage* between the polynomials $\widetilde{f}_i$ and their corresponding restrictions $\widetilde{p}_i = \widetilde{f}_i(\mathbf{r}, \cdot)$ for $i \in [\ell]$ (corresponding to the left half of the diagram in Equation (6)). For concreteness, let C_i be the mPCS commitments to polynomials $\widetilde{f}_i$ for $i \in [\ell]$. Let $z \leftarrow \mathbb{F}$ be a random verifier challenge to check Equation (5), and let v_i, $i \in [\ell]$ be purported evaluations $\hat{q}_i(z)$ which are sent by the prover. We first note that the commitments C_i, $i \in [\ell]$, together with the sumcheck challenges $\mathbf{r}$ implicitly determine the evaluations $\hat{q}_i(z)$. This happens because $\widetilde{f}_i(\mathbf{r}, \cdot)$ determines their corresponding univariate polynomials $\hat{p}_i$, which in turn determine their ifft counterparts $\hat{q}_i$ respectively. Schematically, the lineage from $\{\widetilde{f}_i\}_{i=1}^{\ell}$ and $\mathbf{r}$ to $\{\hat{q}_i\}_{i=1}^{\ell}$ may be depicted as:

$$\begin{array}{ccccccc} \widetilde{f}_i & \overset{\mathbf{r}}{\rightleftharpoons} & \widetilde{p}_i & \overset{\varphi}{\rightleftharpoons} & \hat{p}_i & \overset{\mathsf{ifft/fft}}{\rightleftharpoons} & \hat{q}_i \\ & & \Big\updownarrow \gamma & & & & \\ \widetilde{f} & \overset{\mathbf{r}}{\rightleftharpoons} & \widetilde{p} & \overset{\varphi}{\rightleftharpoons} & \hat{p} & \overset{\mathsf{ifft/fft}}{\rightleftharpoons} & \hat{q} \end{array} \tag{6}$$

Moreover, since all linkages are homomorphic, for a random $\gamma \leftarrow \mathbb{F}$, the same lineage applies to polynomials obtained by batching the constituent polynomials using γ; specifically we have $\widetilde{f} = \sum_{i=1}^{\ell} \gamma^{i-1} \cdot \widetilde{f}_i$, and so forth for others. Clearly, for uniformly and independently sampled γ, it suffices to prove the above lineage over aggregated polynomial – which implies it holds for constituent polynomials except with negligible probability.

As the commitment scheme mPCS is homomorphic, the verifier can compute commitment C_f to the aggregated polynomial $\widetilde{f} = \sum_{i=1}^{\ell} \gamma^{i-1} \widetilde{f}_i$ as $C_f = \sum_{i=1}^{\ell} \gamma^{i-1} \cdot C_i$. The prover then sends the commitment C_p to the aggregated polynomial $\widetilde{p} = \sum_{i=1}^{\ell} \gamma^{i-1} \widetilde{p}_i$. To show that $\widetilde{p}$ is a restriction of $\widetilde{f}$ by challenges $\mathbf{r}$, the verifier samples $\mathbf{y} \leftarrow \mathbb{F}^\kappa$ and queries $\widetilde{p}$ and $\widetilde{f}$ at points $\mathbf{y}$ and $(\mathbf{r}, \mathbf{y})$ respectively. It accepts if the two evaluations are equal. Using the multilinear PCS in [GPS25, EG25], the preceding check incurs $O(m + n)$ cost to the prover and results in $O(1)$ communication.

Additional Optimizations. Note that in a straightforward approach, the prover needs to send two evaluation proofs corresponding to κ-variate multilinear polynomial $\widetilde{p}(\cdot)$ at $\mathbf{y}$ and μ-variate multilinear polynomial $\widetilde{f}(\cdot)$ at $(\mathbf{r}, \mathbf{y})$, respectively. However, we reduce it to a single multilinear PCS evaluation proof by (i) extending $\widetilde{p}(\cdot)$ into a μ-variate multilinear polynomial $\widetilde{p}_{\mathsf{ext}}(\cdot)$, and (ii) batching $\widetilde{p}_{\mathsf{ext}}(\cdot)$ and $\widetilde{f}(\cdot)$ as a single μ-variate multilinear polynomial. This reduces the concrete communication cost further. We provide details of this optimization in Section 4. The linkage of $\widetilde{p}$ to $\hat{p}$ is automatic, as the commitment C_p for the polynomial $\widetilde{p}$ is also a commitment to its univariate counterpart $\hat{p}$. The aggregated polynomial $\hat{p}$ can now be used to "authenticate" the evaluations v_i, $i \in [\ell]$. We observe that $\hat{q}(z) = \sum_{i=1}^{\ell} \gamma^{i-1} v_i$. Since $\hat{q} = \mathsf{ifft}(\hat{p})$, we can write this as:

$$\sum_{j=1}^{m} p_j \mu_j(z) = v' \tag{7}$$

where $v' = \sum_{i=1}^{\ell} \gamma^{i-1} \cdot v_i$. The PIOP to check Equation (7) requires new techniques, which we defer to Phase-3.

Summary. We now summarize how the above idea is translated into Phase-2 (lines 10-16) of Figure 1. The verifier $\mathcal{V}$ starts this phase by sending the batching challenge $\gamma \leftarrow \mathbb{F}$. The prover $\mathcal{P}$ responds with the commitment C_p, purported to be the commitment to the polynomial $\widetilde{p} = \sum_{i=1}^{\ell} \gamma^{i-1} \cdot \widetilde{p}_i$. $\mathcal{V}$ homomorphically computes commitment $C_f = \sum_{i=1}^{\ell} \gamma^{i-1} C_i$ to the polynomial $\widetilde{f} = \sum_{i=1}^{\ell} \gamma^{i-1} \widetilde{f}_i$.

To check that $\widetilde{p}$ corresponds to restriction of $\widetilde{f}$ by challenges $\mathbf{r}$, $\mathcal{V}$ samples $\mathbf{y} \in \mathbb{F}^{\kappa}$ to check $\widetilde{p}(\mathbf{y}) \stackrel{?}{=} \widetilde{f}(\mathbf{r}, \mathbf{y})$. We now define the μ-variate polynomial $\widetilde{p}_{\mathsf{ext}}$ that vacuously extends $\widetilde{p}$ to B_μ as $\widetilde{p}_{\mathsf{ext}}(X_1, \ldots, X_\mu) = \widetilde{p}(X_{\rho+1}, \ldots, X_\mu)$, i.e, $\widetilde{p}_{\mathsf{ext}}$ essentially ignores the first ρ variables. Thus, we can obtain the query $\widetilde{p}(\mathbf{y})$ as $\widetilde{p}_{\mathsf{ext}}(\mathbf{r}, \mathbf{y})$. Let $\hat{p}_{\mathsf{ext}}(X)$ be the univariate polynomial of degree $< n$ corresponding to μ-variate polynomial $\widetilde{p}_{\mathsf{ext}}$. We need to relate it to the polynomial $\hat{p}(X)$ of degree at most m corresponding to $\widetilde{p}$. It can be seen that if $(p_1, \ldots, p_m)$ are the evaluations of $\widetilde{p}$ over B_κ, the evaluations of $\widetilde{p}_{\mathsf{ext}}$ over B_μ are:

$$\underbrace{p_1, \ldots, p_1}_{s \text{ times}}, \underbrace{p_2, \ldots, p_2}_{s \text{ times}}, \ldots, \underbrace{p_m, \ldots, p_m}_{s \text{ times}}$$

where $s = 2^\rho$. This holds because the evaluation of $\widetilde{p}_{\mathsf{ext}}$ is independent of the first ρ variables. The relation describing the corresponding univariate polynomials $\hat{p}_{\mathsf{ext}}(X)$ and $\hat{p}(X)$ is:

$$\begin{aligned} \hat{p}_{\mathsf{ext}}(X) &= \sum_{i=1}^{m} p_i X^{s(i-1)} (1 + X + \cdots + X^{s-1}) \\ &= \hat{p}(X^s) \cdot (X^s - 1)/(X - 1) \end{aligned} \tag{8}$$

Accordingly, $\mathcal{P}$ sends the commitment $C_{p_{\mathsf{ext}}}$ to polynomial $\hat{p}_{\mathsf{ext}}$, and $\mathcal{V}$ checks the identity in Equation (8). In particular, to enable the identity check, $\mathcal{V}$ sends a random $\eta \in \mathbb{F}$, and in response $\mathcal{P}$ sends (i) $v_\eta = \hat{p}(\eta^s)$ along with a uPCS evaluation proof of $\hat{p}(X)$ at $X = \eta^s$, and (ii) a uPCS evaluation proof of $\hat{p}_{\mathsf{ext}}(X)$ at $X = \eta$. These two checks by $\mathcal{V}$ are summarized in lines 31 and 32 of Figure 1. Additionally, we can now batch the multilinear queries for $\widetilde{p}_{\mathsf{ext}}$ and $\widetilde{f}$ into a single query (verified in line 30 of Figure 1).

4.3 Phase-3: Consistency of Univariate Evaluations

In this phase, the prover establishes to the verifier that the evaluations of the univariate polynomials $\hat{q}_i$ used in Phase-1 are consistent with the corresponding restricted multilinear polynomials authenticated in Phase-2. Put together with Phase-2, this allows the prover to establish to the verifier the correct overall linkage between the responses $\hat{q}_i(z)$ and the polynomials $\widetilde{f}_i$ involved in the multilinear sumcheck.

A bit more concretely, in phase-3, the verifier needs to authenticate all the evaluations $v_i = \hat{q}_i(z)$ for each $i \in [\ell]$ that were sent by the prover in the first phase. To this end, the verifier uses the polynomial $\hat{p}(X)$, effectively establishing the linkage in the right half of diagram in Equation (6). However, instead of

authenticating polynomials $\hat{q}_i(X)$, it suffices for the verifier to authenticate their evaluations at z. If the checks in the previous two phases succeed, the verifier knows that the opening $\hat{p}(X)$ to the commitment C_p must satisfy $\hat{q}(X) = \mathsf{ifft}(\hat{p}(X))$ for correctly computed polynomial $\hat{q}$. In terms of Lagrange basis polynomials, this is equivalent to:

$$\hat{q}(z) = \sum_{j=1}^{m} p_j \mu_j^{\mathbb{H}}(z),$$

where p_j is the j^{th} coefficient of $\hat{p}$, and $\mu_j^{\mathbb{H}}(z) = \frac{\omega^{j-1}(z^m-1)}{m(z-\omega^{j-1})}$. Since $\hat{q}(z)$ is already determined as $\hat{q}(z) = \sum_{i=1}^{\ell} \gamma^{i-1} v_i$ by the evaluations $v_1, \ldots, v_\ell$ sent by the prover, the above relation reduces to checking:

$$\sum_{j=1}^{m} p_j \cdot \frac{\omega^{j-1}(z^m-1)}{m(z-\omega^{j-1})} \stackrel{?}{=} v' \implies \sum_{j=1}^{m} p_j \cdot \frac{\omega^{j-1}}{z-\omega^{j-1}} \stackrel{?}{=} \frac{mv'}{z^m-1} \tag{9}$$

where $v' = \sum_{i=1}^{\ell} \gamma^{i-1} v_i$. At this point, we introduce some tools that will help us checking the identity in Equation (9).

Definition 4.1 (Products on Polynomials)**.** *Let $m \in \mathbb{N}$ and let $\hat{a}(X) = \sum_{j=1}^{m}$ and $\hat{b}(X) = \sum_{j=1}^{m} b_j X^{j-1}$ be polynomials in $\mathbb{F}^{<m}[X]$. We use $\langle \hat{a}, \hat{b} \rangle$ to denote the inner product $\sum_{i=1}^{m} a_j b_j$ of the coefficient vectors of the two polynomials. We also define the hadamard product $\hat{a} \circ \hat{b}$ as the polynomial $\hat{c}(X) = \sum_{i=1}^{m} c_i X^{i-1}$ where $c_i = a_i \cdot b_i$, in other words $\hat{a} \circ \hat{b}$ is the polynomial whose coefficient vector is the hadamard product of the coefficient vectors of respective polynomials.*

We also record following useful results from [EG25, Claim 4.1].

Lemma 4.1. *For polynomials $\hat{p} = \sum_{i=1}^{m} p_i X^{i-1}$ and $\hat{q} = \sum_{i=1}^{m} q_i X^{i-1}$ in $\mathbb{F}[X]$, let $\langle \hat{p}, \hat{q} \rangle$ denote the sum $\sum_{i=1}^{m} p_i q_i$. Then, $\langle \hat{p}, \hat{q} \rangle = v$ if and only if there exists a polynomial $\hat{S} \in \mathbb{F}[X]$, computable in $O(m \log m)$ field operations such that:*

$$\hat{p}(X) \cdot \hat{q}(1/X) + \hat{p}(1/X) \cdot \hat{q}(X) = 2v + X \cdot \hat{S}(X) + (1/X) \cdot \hat{S}(1/X)$$

The following variant of Lemma 4.1 will also be useful.

Lemma 4.2. *For polynomials $\hat{p} = \sum_{i=1}^{m} p_i X^{i-1}$ and $\hat{q} = \sum_{i=1}^{m} q_i X^{i-1}$, let $\hat{r} = \hat{p} \circ \hat{q}$ be the polynomial $\sum_{i=1}^{m} r_i X^{i-1}$ with $r_i = p_i q_i$ for all $i \in [m]$. Then, with overwhelming probability over the choice of $\delta \leftarrow \mathbb{F}$, the polynomials $\hat{p}, \hat{q}$ and $\hat{r}$ in $\mathbb{F}^{<m}[X]$ satisfy $\hat{r} = \hat{p} \circ \hat{q}$ if and only if $\langle \hat{p}(\delta X), \hat{q}(X) \rangle = \hat{r}(\delta)$. Equivalently, there exists a polynomial $\hat{S}$, computable in $O(m \log m)$ field operations such that:*

$$\hat{p}(\delta X) \cdot \hat{q}(1/X) + \hat{p}(\delta/X) \cdot \hat{q}(X) = 2\hat{r}(\delta) + X \cdot \hat{S}(X) + (1/X) \cdot \hat{S}(1/X)$$

Proof. The proof essentially follows from batching the checks $r_i = p_i q_i$ using the challenge δ. □

We also note that Lemmas 4.1 and 4.2 also imply efficient PIOPs for relations $\langle \hat{p}, \hat{q} \rangle = \sigma$ and $\hat{p} \circ \hat{q} = \hat{r}$, given commitments to the involved polynomials. To check the identity in Equation (9), $\mathcal{P}$ and $\mathcal{V}$ define the polynomial

$$\hat{A}_z(X) = \sum_{i=1}^{m} \frac{\omega^{j-1}}{z-\omega^{j-1}} \cdot X^{j-1} \tag{10}$$

The prover then sends commitment C_A to the polynomial $\hat{A}_z(X)$. In terms of polynomial products, the check in Equation (9) is equivalent to the check: $\langle \hat{p}, \hat{A}_z \rangle = mv'/(z^m-1)$.

At this point, the verifier still needs to ensure that the commitment C_A opens to the correct polynomial $\hat{A}_z$ as defined. To do so, we define "inverse-coefficient" polynomial

$$\begin{aligned} \hat{B}_z(X) = \sum_{j=1}^{m} \frac{z-\omega^{j-1}}{\omega^{j-1}} X^{j-1} &= z \sum_{i=1}^{m} (\omega^{-1}X)^{j-1} - \frac{X^m-1}{X-1} \\ &= z\Psi_{\omega^{-1}}(X) - \frac{X^m-1}{X-1}, \end{aligned} \tag{11}$$

where $\Psi_{\omega^{-1}}(X) = \prod_{i=1}^{\kappa}(1 + \omega^{-2^{i-1}} X^{2^{i-1}})$, We observe that $\hat{B}_z(X)$can be evaluated by the verifier at any point in $O(\kappa) = O(\log m)$ field operations. The description of polynomial $\hat{B}_z$ allows the verifier to check the correctness of purported polynomial $\hat{A}_z(X)$ committed by C_A using the hadamard identity:

$$\hat{A}_z \circ \hat{B}_z = \frac{X^m - 1}{X - 1}$$

Summary. We now summarize how the above idea translates to Phase-3 (lines 17-28 of Figure 1. Observe that we have effectively reduced checking the identity in Equation (9) to the prover $\mathcal{P}$ exhibiting a polynomial $\hat{A}_z(X)$ and proving:

$$\langle\, \hat{p}, \hat{A}_z \,\rangle = \frac{mv'}{z^m - 1}, \quad \hat{A}_z \circ \hat{B}_z = \frac{X^m - 1}{X - 1} \tag{12}$$

Now Lemmas 4.1 and 4.2 imply a univariate PIOP to check relation in Equation (12), wherein the verifier $\mathcal{V}$ sends a challenge $\delta \leftarrow \mathbb{F}$ and, subsequently, the prover $\mathcal{P}$ sends commitments to polynomials $\hat{S}_1(X)$ and $\hat{S}_2(X)$ satisfying:

$$\hat{p}(X) \cdot \hat{A}_z(1/X) + \hat{p}(1/X) \cdot \hat{A}_z(X) = \frac{2mv'}{z^m - 1} + X \cdot \hat{S}_1(X) + (1/X) \cdot \hat{S}_1(1/X),$$
$$\hat{A}_z(X) \cdot \hat{B}_z(\delta/X) + \hat{A}_z(1/X) \cdot \hat{B}_z(\delta X) = \frac{2(\delta^m - 1)}{\delta - 1} + X \cdot \hat{S}_2(X) + (1/X) \cdot \hat{S}_2(1/X)$$

Using another challenge $\epsilon \leftarrow \mathbb{F}$ from the verifier $\mathcal{V}$, the above can be batched into a single check:

$$\begin{aligned} &\hat{p}(X) \cdot \hat{A}_z(1/X) + \hat{p}(1/X) \cdot \hat{A}_z(X) + \epsilon\big(\hat{A}_z(X) \cdot \hat{B}_z(\delta/X) + \hat{A}_z(1/X) \cdot \hat{B}_z(\delta X)\big) \\ &\quad = 2\Big(\frac{mv'}{z^m - 1} + \epsilon \cdot \frac{\delta^m - 1}{\delta - 1}\Big) + X \cdot \hat{S}(X) + (1/X) \cdot \hat{S}(1/X). \end{aligned} \tag{13}$$

Now, instead of committing to $\hat{S}_1(X)$ and $\hat{S}_2(X)$, the prover $\mathcal{P}$ can commit to a single polynomial $\hat{S}(X)$. To check the polynomial identity in Equation 13, $\mathcal{V}$ sends a random $r \in \mathbb{F}^*$; in response, $\mathcal{P}$ sends (i) evaluations of $\hat{p}(X)$, $\hat{A}_z(X)$, and $\hat{S}(X)$ at $X = 1/r$ along with a single uPCS evaluation proof batched over a random $r' \in \mathbb{F}$, (ii) and a uPCS evaluation proof for a linear combination of the three aforementioned polynomials (denoted by $\hat{b}(X)$ in Figure 1) at $X = r$, where r' is another random challenge by $\mathcal{V}$. These two verifier checks correspond to lines 33-34 in Figure 1. This concludes the description of Phase-3 of our sumcheck argument.

Remark 4.1. *We emphasize that Phase-3 allows the verifier to check that each value v_i sent across by the prover is indeed the claimed evaluation $\hat{q}_i(z)$, **without** requiring the prover to explicitly commit to the $\hat{q}_i$ polynomials. Instead, we exploit the algebraic relation between $\hat{q}_i$ and the corresponding committed polynomial $\hat{p}_i$ to reduce the evaluation claims on the $\hat{q}_i$ polynomials to evaluation claims on the $\hat{p}_i$ polynomials. Avoiding explicit commitments to the $\hat{q}_i$ polynomials is important for preserving the low communication complexity of our sumcheck argument.*

4.4 Putting Everything Together

We now state the following theorem summarizing the sumcheck argument in Figure 1.

Theorem 4.1. *In the Algebraic Group Model (AGM), under the q-DLOG assumption (Definition 2.7), the protocol described in Figure 1, when instantiated with* uPCS = KZG *[KZG10] and* mPCS = Samaritan *[GPS25], is a succinct argument of knowledge for the multilinear sumcheck relation R in Equation* (1) *(parameterized by n, d, ℓ) that achieves completeness as per Definition 2.1, knowledge soundness as per Definition 2.2, and the following efficiency properties:*

- *Prover-to-verifier communication:* $O(\ell + d \log\log n)$.

- *Prover cost:* $O(c \cdot n)$ $\mathbb{F}$*-operations and* $O(n + \ell n/\log n)$ $\mathbb{G}_1$ *operations. for some constant* $c = \mathcal{C}(\widetilde{\mathbf{G}}, d, \ell)$ *dependent on the multivariate form* $\widetilde{\mathbf{G}}$*,* d *and* ℓ*.*
- *Verifier cost:* $O(\ell + \log n)$ $\mathbb{F}$*-operations and* $O(\ell)$ $\mathbb{G}_1$*-operations.*

We analyze the prover-to-verifier communication of the protocol, since this corresponds to the proof size of the non-interactive version of the argument after applying the Fiat-Shamir transform.

Proof. We present the proofs of completeness, knowledge soundness and efficiency below.

4.4.1 Completeness

It follows from the detailed analysis of the protocol phases in Sections 4.1, 4.2, and 4.3 that any honest prover $\mathcal{P}$ holding a valid witness $\mathsf{wit} = \{\widetilde{f}_i(\mathbf{x})\}_{i\in[\ell]}$ such that

$$\sum_{\mathbf{x}\in B_\mu} \widetilde{G}(\widetilde{f}_1(\mathbf{x}), \ldots, \widetilde{f}_\ell(\mathbf{x})) = v$$

passes all of the verification checks in Figure 1 with probability 1. Hence, completeness holds.

4.4.2 Knowledge Soundness

We now present a detailed proof of knowledge soundness. Let $\mathcal{P}^*$ be an algebraic PPT prover interacting with the honest verifier $\mathcal{V}$. We show that there exists a PPT extractor $\mathcal{E}$ such that, whenever $\mathcal{P}^*$ causes $\mathcal{V}$ to accept, $\mathcal{E}$ outputs multilinear polynomials $\{\widetilde{f}_i\}_{i\in[\ell]}$ satisfying the relation R in Equation (1), except with negligible probability.

Extraction of Committed Polynomials. Let $\mathcal{P}^*$ be an algebraic PPT prover. By the definition of the Algebraic Group Model (AGM), every group element output by $\mathcal{P}^*$ is accompanied by an algebraic representation in terms of previously seen group elements. Recursively expanding these representations with respect to the monomial KZG SRS allows an extractor $\mathcal{E}$ to associate with every commitment generated by $\mathcal{P}^*$ a polynomial whose coefficient vector is given by the corresponding algebraic representation.

In particular, from the initial commitments $\{C_i\}_{i\in[\ell]}$, the extractor obtains coefficient vectors $\{\mathbf{f}_i\}_{i\in[\ell]}$. Under the isomorphism φ^{-1} from Section 3.1, each $\mathbf{f}_i$ uniquely determines a μ-variate multilinear polynomial $\widetilde{f}_i$. We define $\mathsf{wit}^* = \{\widetilde{f}_i\}_{i\in[\ell]}$ to be the extracted witness.

Next, applying the same argument as above to the auxiliary prover-generated commitments

$$(C_g, C_h, C_p, C_{p_{\mathsf{ext}}}, C_A, C_S)$$

associates with them fixed polynomials

$$(\hat{g}, \hat{h}, \hat{p}, \hat{p}_{\mathsf{ext}}, \hat{A}_z, \hat{S}),$$

respectively. Moreover, the coefficient vector extracted from $C_{p_{\mathsf{ext}}}$ also determines the corresponding multilinear polynomial $\widetilde{p}_{\mathsf{ext}} = \varphi^{-1}(\hat{p}_{\mathsf{ext}})$. Note that each such polynomial is fixed at the point at which the corresponding commitment is sent by the prover $\mathcal{P}^*$.

Associating Polynomials with Verifier-derived Commitments. Observe that in Figure 1, $\mathcal{V}$ derives a set of commitments from the available prover-sent commitments. Let us denote it with

$$C_{\mathsf{derived}} = \{C'_1, C_f, C'_2, C'_3, C'_4\}.$$

Without loss of generality we consider a modified protocol where $\mathcal{P}^*$ sends these commitments to $\mathcal{V}$ and $\mathcal{V}$ checks their correctness. For e.g., if $\mathcal{V}$ derives cm as $\sum_{i=1}^{k} \gamma_i \mathsf{cm}_i$ for some scalars γ_i and commitments cm_i

(for $i \in [k]$) sent by $\mathcal{P}^*$, then the equivalent modification requires $\mathcal{P}^*$ to additionally send the commitment cm to $\mathcal{V}$; $\mathcal{V}$ then checks that $\mathsf{cm} = \sum_{i=1}^{k} \gamma_i \mathsf{cm}_i$, and rejects if the check fails. In this modified protocol, $\mathcal{E}$ directly extracts polynomials corresponding to the commitments in C_{derived}. If $\mathcal{E}$ extracts polynomials $a(X)$ and $a_1(X), \cdots, a_k(X)$ corresponding to commitments cm and $\mathsf{cm}_1, \cdots, \mathsf{cm}_k$, such that $\mathsf{cm} = \sum_{i=1}^{k} \gamma_i \mathsf{cm}_i$, but $a(X) \neq \sum_{i=1}^{k} \gamma_i a_i(X)$, then $\mathcal{E}$ aborts. If abort happens, then $a(X)$ and $\sum_{i=1}^{k} \gamma_i a_i(X)$ are two polynomials with same commitment cm, which violates the binding of the underlying polynomial commitment schemes. Therefore, $\mathcal{E}$ aborts with negligible probability.

Assuming that E does not abort, we conclude that the polynomials $\hat{a}(X)$, $\widetilde{f}(\mathbf{x})$, $\widetilde{p}_{\mathsf{ext}}(\mathbf{x}) - \widetilde{f}(\mathbf{x})$, $\hat{b}(X)$, and $\hat{c}(X)$ corresponding to the commitments C_1', C_f, C_2', C_3', and C_4', respectively, satisfy

$$\begin{aligned}
\hat{a}(X) &= (z^m - 1)\hat{g}(X) + z\hat{h}(X), \\
\widetilde{f}(\mathbf{x}) &= \sum_{i=1}^{\ell} \gamma^{i-1} \widetilde{f}_i(\mathbf{x}), \\
\widetilde{p}_{\mathsf{ext}}(\mathbf{x}) - \widetilde{f}(\mathbf{x}) &= \varphi^{-1}(\hat{p}_{\mathsf{ext}}(X)) - \widetilde{f}(\mathbf{x}), \\
\hat{b}(X) &= v_{\mathsf{Ainv}} \hat{p}(X) + (v_{\mathsf{pinv}} + \epsilon v_{\mathsf{Binv}}) \hat{A}_z(X) - r\hat{S}(X), \\
\hat{c}(X) &= \hat{p}(X) + r' \hat{A}_z(X) + r'^2 \hat{S}(X).
\end{aligned}$$

Defining event $\mathsf{Good}_{\mathsf{PCS}}$. We now define an event $\mathsf{Good}_{\mathsf{PCS}}$ that for each accepting evaluation proof, the extracted polynomial corresponding to the commitment satisfies the evaluation claim. Assuming that uPCS and mPCS satisfy soundness, we have

$$\Pr[\neg \mathsf{Good}_{\mathsf{PCS}}] \leq \mathsf{negl}(\lambda) \implies \Pr[\mathsf{Good}_{\mathsf{PCS}}] > 1 - \mathsf{negl}(\lambda).$$

When $\mathsf{uPCS} = \mathsf{KZG}$ and $\mathsf{mPCS} = \mathsf{Samaritan}$, the above holds by the q-DLOG assumption in the AGM.

In the rest of the proof, we condition on the event $\mathsf{Good}_{\mathsf{PCS}}$, and analyze the resulting randomized polynomial-identity checks.

Defining event E. At this point, we introduce an auxiliary event E for ease of exposition. We say that E occurs if, for each $i \in [\ell]$, $v_i = \hat{q}_i(z)$, i.e., v_i is the correct evaluation of $\hat{q}_i(X)$ at $X = z$, in Phase-1 of Figure 1.

Remark 4.2. *We remark that the event E is not covered by the event $\mathsf{Good}_{\mathsf{PCS}}$ defined above, since the prover $\mathcal{P}^\star$ is not required to explicitly commit to the $\hat{q}_i$ polynomials in Figure 1.*

We now state the following lemmas.

Lemma 4.3. *Conditioned on the intersection of the events $\mathsf{Good}_{\mathsf{PCS}}$ and E, the set of extracted witness polynomials $\{\widetilde{f}_i\}_{i \in [\ell]}$ satisfy the sumcheck relation in Equation 1, except with negligible probability.*

Proof. For every $i \in [\ell]$, let $\widetilde{p}_i(\mathbf{x}) = \widetilde{f}_i(\mathbf{r}, \mathbf{x})$ for $\mathbf{x} \in \mathbb{F}^\kappa$, where $\widetilde{f}_i$ is the extracted witness polynomial and $\mathbf{r} \in \mathbb{F}^\rho$ denotes the challenges generated during Phase-0. Let $\hat{p}_i \in \mathbb{F}^{<m}[X]$ denote the univariate polynomial corresponding to $\widetilde{p}_i$ under the isomorphism φ, and define $\hat{q}_i = \mathsf{ifft}(\hat{p}_i)$. Thus, the polynomials $\hat{q}_i$ are uniquely determined by the extracted witness and the Phase 0 challenges and they are fixed before the verifier samples z. We define,

$$D_1(X) = \hat{a}(X) - \widetilde{G}(v_1, \cdots, v_\ell) + \bar{v}/m,$$

where $\hat{a}(X) = (X^m - 1)\hat{g}(X) + X\hat{h}(X)$. Conditioned on the intersection of the events $\mathsf{Good}_{\mathsf{PCS}}$ and E, we get

$$D_1(z) = \hat{a}(z) - \widetilde{G}(\hat{q}_1(z), \cdots, \hat{q}_\ell(z)) + \bar{v}/m,$$

and the accepting evaluation check in line-29 of Figure 1 implies

$$D_1(z) = 0.$$

Since $\deg(\hat{q}_i) < m$ for every $i \in [\ell]$ and $\widetilde{G}$ has total degree at most d, we must have $\deg\Big(\widetilde{G}(\hat{q}_1, \ldots, \hat{q}_\ell)\Big) \leq d(m-1)$. Moreover, by construction of the univariate sumcheck PIOP and the degree check on $\hat{h}$ (line-35), the remaining terms in D_1 have degree at most $d(m-1)$. Hence $\deg(D_1) \leq d(m-1)$. Since $\hat{g}(X)$, $\hat{h}(X)$ are fixed before z is sampled, by the univariate Schwartz–Zippel bound, we obtain:

$$\Pr_z[D_1(z) = 0 \wedge D_1 \not\equiv 0] \leq \frac{d(m-1)}{|\mathbb{F}|},$$

which implies that the polynomial identity $D_1(X) = 0$ holds except with negligible probability. Therefore,

$$\widetilde{G}\big(\hat{q}_1(X), \ldots, \hat{q}_\ell(X)\big) = (X^m - 1)\hat{g}(X) + X\hat{h}(X) + \frac{\bar{v}}{m}.$$

Let $\mathbb{H} = \{1, \omega, \ldots, \omega^{m-1}\}$. Then, summing the above equation over $h \in \mathbb{H}$, the first term on the right-hand side vanishes because $h^m = 1$ for every $h \in \mathbb{H}$. Moreover, since $\deg(\hat{h}) \leq m-2$, the polynomial $X\hat{h}(X)$ has degree at most $m-1$ and has zero constant coefficient. By orthogonality over the multiplicative subgroup, we obtain

$$\sum_{h \in \mathbb{H}} \widetilde{G}\big(\hat{q}_1(h), \ldots, \hat{q}_\ell(h)\big) = \bar{v}.$$

By definition, $\hat{q}_i = \mathsf{ifft}(\hat{p}_i)$, and hence, $\hat{q}_i(\omega^{j-1}) = \widetilde{p}_i(\langle j \rangle) = \widetilde{f}_i(\mathbf{r}, \langle j \rangle)$ for every $i \in [\ell]$ and $j \in [m]$. Therefore, the above equation is precisely

$$\sum_{\mathbf{x} \in B^\kappa} \widetilde{G}\big(\widetilde{f}_1(\mathbf{r}, \mathbf{x}), \ldots, \widetilde{f}_\ell(\mathbf{r}, \mathbf{x})\big) = \bar{v},$$

which is exactly the reduced sumcheck claim of Equation 2.

Observe that the polynomial $F(\mathbf{X}) = \widetilde{G}\big(\widetilde{f}_1(\mathbf{X}), \ldots, \widetilde{f}_\ell(\mathbf{X})\big)$, has individual degree at most d, since every $\widetilde{f}_i$ is multilinear and $\widetilde{G}$ has total degree at most d. Therefore, by the standard round-by-round soundness analysis of the classical sumcheck protocol, if the original claim is false then the probability that the first ρ rounds terminate in a valid reduced claim of the form above is at most $\rho d/|\mathbb{F}|$. Hence, except with probability at most $\rho d/|\mathbb{F}|$, validity of the reduced sumcheck equation above implies

$$\sum_{\mathbf{x} \in B^\mu} \widetilde{G}\big(\widetilde{f}_1(\mathbf{x}), \ldots, \widetilde{f}_\ell(\mathbf{x})\big) = v,$$

as desired. Hence we conclude that the set of extracted polynomials satisfy the sumcheck relation of Equation 1 except with probability at most

$$\frac{d(m-1)}{|\mathbb{F}|} + \frac{\rho d}{|\mathbb{F}|} = \mathsf{negl}(\lambda),$$

where the probability is over random choices of $(z, \mathbf{r})$. This concludes the proof of Lemma 4.3. □

Lemma 4.4. *Conditioned on the event* $\mathsf{Good}_{\mathsf{PCS}}$, $\Pr[\neg \mathsf{E}] \leq \mathsf{negl}(\lambda)$.

Proof. We first show that the consistency checks in Phase-2 and Phase-3 authenticate the values $\{v_i\}_{i \in [\ell]}$ supplied by the prover in Phase-1. Specifically, the Phase-3 checks establish the claimed evaluation of the batched polynomial $\widehat{p}$ at z, while the Phase-2 checks link $\hat{p}$ to the γ-weighted combination of the restricted witness polynomials. Together, these imply $\sum_{i=1}^{\ell} \gamma^{i-1}(\widehat{q}_i(z) - v_i) = 0$. Finally, a Schwartz–Zippel argument over the random batching challenge γ shows that, except with negligible probability, $\widehat{q}_i(z) = v_i$ for every $i \in [\ell]$, establishing the event E.

We first show that the batched PCS check correctly authenticates the evaluations of the committed polynomials at $1/r$. Let

$$D_2(X) = \hat{c}(X) - (v_{\mathsf{pinv}} + r' \cdot v_{\mathsf{Ainv}} + r'^2 \cdot v_{\mathsf{Sinv}}).$$

Conditioned on $\mathsf{Good}_{\mathsf{PCS}}$, the successful verifier check in line 34 of Figure 1 implies $D_2(1/r) = 0$. By our argument on the extraction of verifier-derived commitments, the following holds due to C_4':

$$\hat{c}(X) = \hat{p}(X) + r' \cdot \hat{A}_z(X) + r'^2 \cdot \hat{S}(X).$$

Together, they yield

$$(\hat{p}(1/r) - v_{\mathsf{pinv}}) + r' \cdot (\hat{A}_z(1/r) - v_{\mathsf{Ainv}}) + r'^2 \cdot (\hat{S}(1/r) - v_{\mathsf{Sinv}}) = 0.$$

Since r' is sampled after $\mathcal{P}^*$ has sent the values v_{pinv}, v_{Ainv}, v_{Sinv} to $\mathcal{V}$, the following three identities hold simultaneously except with probability at most $\frac{2}{|\mathbb{F}|} = \mathsf{negl}(\lambda)$:

$$\hat{p}(1/r) = v_{\mathsf{pinv}}, \quad \hat{A}_z(1/r) = v_{\mathsf{Ainv}}, \quad \hat{S}(1/r) = v_{\mathsf{Sinv}}.$$

We next show that the Phase-3 check at the random challenge r enforces the corresponding identity between $\hat{p}$, $\hat{A}_z$ $\hat{B}_z$ and $\hat{S}$ as captured in Equation (13). Concretely, let

$$D_3(X) = \hat{b}(X) - \left(2\Big(\frac{mv'}{z^m - 1} + \epsilon \cdot \frac{\delta^m - 1}{\delta - 1}\Big) + (1/r) \cdot v_{\mathsf{Sinv}} - \epsilon \cdot v_{\mathsf{Ainv}} \cdot v_{\mathsf{B}}\right).$$

Conditioned on $\mathsf{Good}_{\mathsf{PCS}}$, the successful verifier check in line 33 of Figure 1 implies $D_3(r) = 0$. Now, due to verifier-derived commitment C_3',

$$\hat{b}(X) = v_{\mathsf{Ainv}} \cdot \hat{p}(X) + (v_{\mathsf{pinv}} + \epsilon \cdot v_{\mathsf{Binv}}) \cdot \hat{A}_z(X) - r \cdot \hat{S}(X).$$

Therefore $D_3(r) = 0$ implies,

$$v_{\mathsf{Ainv}} \cdot \hat{p}(r) + (v_{\mathsf{pinv}} + \epsilon \cdot v_{\mathsf{Binv}}) \cdot \hat{A}_z(r) - r \cdot \hat{S}(r) = 2\Big(\frac{mv'}{z^m - 1} + \epsilon \cdot \frac{\delta^m - 1}{\delta - 1}\Big) + (1/r) \cdot v_{\mathsf{Sinv}} - \epsilon \cdot v_{\mathsf{Ainv}} \cdot v_{\mathsf{B}}.$$

Now, in the above relation, we replace v_{pinv} with $\hat{p}(1/r)$, v_{Ainv} with $\hat{A}_z(1/r)$, and v_{Sinv} with $\hat{S}(1/r)$, since the correctness of these evaluations are already proven to hold with overwhelming probability:

$$\begin{aligned}
&\hat{A}_z(1/r) \cdot \hat{p}(r) + (\hat{p}(1/r) + \epsilon \cdot v_{\mathsf{Binv}}) \cdot \hat{A}_z(r) - r \cdot \hat{S}(r) \\
&\quad = 2\Big(\frac{mv'}{z^m - 1} + \epsilon \cdot \frac{\delta^m - 1}{\delta - 1}\Big) + (1/r) \cdot \hat{S}(1/r) - \epsilon \cdot \hat{A}_z(1/r) \cdot v_{\mathsf{B}}.
\end{aligned}$$

Further note that $\mathcal{V}$ itself computes $v_{\mathsf{Binv}} = \hat{B}_z(\delta/r)$ and $v_{\mathsf{B}} = \hat{B}_z(\delta r)$. After replacing v_{Binv} and v_{B}, and rearranging the terms we obtain:

$$\begin{aligned}
&\hat{p}(r) \cdot \hat{A}_z(1/r) + \hat{p}(1/r) \cdot \hat{A}_z(r) + \epsilon\Big(\hat{A}_z(r) \cdot \hat{B}_z(\delta/r) + \hat{A}_z(1/r) \cdot \hat{B}_z(\delta r)\Big) \\
&\quad - 2\left(\frac{mv'}{z^m - 1} + \epsilon \cdot \frac{\delta^m - 1}{\delta - 1}\right) - r \cdot \hat{S}(r) - (1/r) \cdot \hat{S}(1/r) = 0.
\end{aligned}$$

We now define

$$\begin{aligned}
D_3'(X) = &\hat{p}(X) \cdot \hat{A}_z(1/X) + \hat{p}(1/X) \cdot \hat{A}_z(X) + \epsilon\Big(\hat{A}_z(X) \cdot \hat{B}_z(\delta/X) + \hat{A}_z(1/X) \cdot \hat{B}_z(\delta X)\Big) \\
&- 2\left(\frac{mv'}{z^m - 1} + \epsilon \cdot \frac{\delta^m - 1}{\delta - 1}\right) - X\hat{S}(X) - (1/X) \cdot \hat{S}(1/X).
\end{aligned}$$

Further, to apply Schwartz-Zippel, we clear the negative powers in the above polynomial and obtain an ordinary polynomial of bounded degree as

$$D_3''(X) = X^{m-1} D_3'(X).$$

We have already established that $D_3'(r) = 0$, and hence, $D_3''(r) = r^{m-1} \cdot D_3'(r) = 0$. By construction,

$$\deg(\hat{p}), \deg(\hat{A}_z), \deg(\hat{B}_z) \leq m-1,$$

and the polynomial $\hat{S}$ arising from Lemmas 4.1 and 4.2 can be taken to satisfy $\deg(\hat{S}) \leq m-2$. Consequently, every monomial occurring in D_3' has exponent in the interval $[-(m-1), m-1]$. Therefore it follows that $\deg(D_3'') \leq 2m-2$. Since the random challenge r is sampled after the associated polynomials are fixed, the univariate Schwartz-Zippel bound gives,

$$\Pr_r[D_3''(r) = 0 \wedge D_3'' \not\equiv 0] \leq \frac{2m-2}{|\mathbb{F}|},$$

which implies that the polynomial identity $D_3''(X) = 0$ holds except with negligible probability. So,

$$\begin{aligned} X^{m-1}\Big(&\hat{p}(X) \cdot \hat{A}_z(1/X) + \hat{p}(1/X) \cdot \hat{A}_z(X) + \epsilon\Big(\hat{A}_z(X) \cdot \hat{B}_z(\delta/X) + \hat{A}_z(1/X) \cdot \hat{B}_z(\delta X)\Big) \\ &- 2\Big(\frac{mv'}{z^m - 1} + \epsilon \cdot \frac{\delta^m - 1}{\delta - 1}\Big)\Big) \\ &= X^m \cdot \hat{S}(X) + X^{m-2} \cdot \hat{S}(1/X). \end{aligned}$$

In the right hand side of the above polynomial identity, the minimum power of X in $X^m \cdot \hat{S}(X)$ is m, and the maximum power of X in $X^{m-2} \cdot \hat{S}(1/X)$ is $m-2$. Thus coefficient of X^{m-1} is zero in the right hand side. So, coefficient of X^{m-1} must be zero in the left hand side also. Hence, we write,

$$\begin{aligned} \mathsf{const}\Big(&\hat{p}(X) \cdot \hat{A}_z(1/X) + \hat{p}(1/X) \cdot \hat{A}_z(X) + \epsilon\Big(\hat{A}_z(X) \cdot \hat{B}_z(\delta/X) + \hat{A}_z(1/X) \cdot \hat{B}_z(\delta X)\Big)\Big) \\ &- 2\Big(\frac{mv'}{z^m - 1} + \epsilon \cdot \frac{\delta^m - 1}{\delta - 1}\Big)\Big) = 0, \end{aligned}$$

where $\mathsf{const}(\cdot)$ represents the constant term of the expression inside. Observe,

$$\mathsf{const}\big(\hat{p}(X) \cdot \hat{A}_z(1/X) + \hat{p}(1/X) \cdot \hat{A}_z(X)\big) = 2\sum_{i=0}^{m-1} \hat{p}[i] \cdot \hat{A}_z[i],$$

$$\mathsf{const}\big(\hat{A}_z(X) \cdot \hat{B}_z(\delta/X) + \hat{A}_z(1/X) \cdot \hat{B}_z(\delta X)\big) = 2\sum_{i=0}^{m-1} \delta^i \cdot \hat{A}_z[i] \cdot \hat{B}_z[i].$$

Therefore,

$$\Big(\sum_{i=0}^{m-1} \hat{p}[i] \cdot \hat{A}_z[i] - \frac{mv'}{z^m - 1}\Big) + \epsilon \cdot \Big(\sum_{i=0}^{m-1} \delta^i \cdot \hat{A}_z[i] \cdot \hat{B}_z[i] - \frac{\delta^m - 1}{\delta - 1}\Big) = 0.$$

Since $\hat{B}_z$ is computed by $\mathcal{V}$ itself, and $\hat{p}$ and $\hat{A}_z$ are sent by $\mathcal{P}^*$ before random challenge ϵ is received, the following two identities hold together with probability $(1 - \frac{1}{|\mathbb{F}|})$:

$$\sum_{i=0}^{m-1} \hat{p}[i] \cdot \hat{A}_z[i] = \frac{mv'}{z^m - 1}, \quad \sum_{i=0}^{m-1} \delta^i \cdot \hat{A}_z[i] \cdot \hat{B}_z[i] = \frac{\delta^m - 1}{\delta - 1}.$$

At this point, we show that the random challenge δ forces the committed polynomial $\hat{A}_z$ to have the prescribed coefficients determined by $\hat{B}_z$. To this end, we define

$$\begin{aligned} D_4(X) &= \sum_{i=0}^{m-1} (\hat{A}_z[i] \cdot \hat{B}_z[i]) \cdot X^i - \frac{X^m - 1}{X - 1}, \\ &= \sum_{i=0}^{m-1} (\hat{A}_z[i] \cdot \hat{B}_z[i]) \cdot X^i - \sum_{i=0}^{m-1} X^i. \end{aligned}$$

We have already established $D_4(\delta) = 0$. Note that $\deg(D_4) = m - 1$. Since δ is a random variable received after $\hat{A}_z$ is determined, by the univariate Schwartz-Zippel bound,

$$\Pr_{\delta}[D_4(\delta) = 0 \wedge D_4 \not\equiv 0] < \frac{m-1}{|\mathbb{F}|},$$

which implies the polynomial identity $D_4(X) = 0$. Hence,

$$\hat{A}_z[i] \cdot \hat{B}_z[i] = 1, \quad \forall i \in [0, m-1].$$

Recall that $\mathcal{V}$ computes $\hat{B}_z(X)$ such that $\hat{B}_z[i] = \frac{z-\omega^i}{\omega^i}$, which implies,

$$\hat{A}_z[i] = \frac{\omega^i}{z-\omega^i}, \quad \forall i \in [0, m-1].$$

Now, we replace the value of $\hat{A}_z[i]$ in the previously established equality $\sum_{i=0}^{m-1} \hat{p}[i] \cdot \hat{A}_z[i] = \frac{mv'}{z^m-1}$, to obtain

$$\begin{aligned}\sum_{i=0}^{m-1} \hat{p}[i] \cdot \frac{\omega^i}{z-\omega^i} = \frac{mv'}{z^m-1}, &\implies \sum_{i=1}^{m} \hat{p}[i-1] \cdot \frac{\omega^{i-1}(z^m-1)}{m(z-\omega^{i-1})} = v', \\ &\implies \sum_{i=1}^{m} \hat{p}[i-1] \cdot \mu_i^{\mathbb{H}}(z) = v',\end{aligned}$$

where $\{\mu_i^{\mathbb{H}}(z)\}_{i\in[m]}$ are the set of Lagrange basis polynomials evaluated at z over the multiplicative group $\mathbb{H} = \{1, \omega, \ldots, \omega^{m-1}\}$ of order m. Recall that $\mathcal{V}$ itself computes $v' = \sum_{j=1}^{\ell} \gamma^{j-1} v_j$. Hence,

$$\sum_{i=1}^{m} \hat{p}[i-1] \cdot \mu_i^{\mathbb{H}}(z) = \sum_{j=1}^{\ell} \gamma^{j-1} v_j.$$

At this point, we use the Phase-2 multilinear evaluation check to link $\widetilde{p}_{\mathsf{ext}}$ to the γ-weighted combination of the restricted extracted witness polynomials.Observe the evaluation checks performed in lines 30-32 in Figure 1. We define the following κ-variate multilinear polynomial:

$$D_5(\mathbf{Y}) = \widetilde{p}_{\mathsf{ext}}(\mathbf{r}, \mathbf{Y}) - \widetilde{f}(\mathbf{r}, \mathbf{Y}),$$

where $\mathbf{r} = (r_1, \cdots, r_\rho) \in \mathbb{F}^\rho$ are the verifier's random challenges for first ρ rounds of the classical multivariate sumcheck in Phase-0, and is fixed before D_5 is determined. Conditioned on $\mathsf{Good}_{\mathsf{PCS}}$, the accepting evaluation check in line-30 implies $D_5(\mathbf{y}) = 0$. Since $\mathbf{y} \in \mathbb{F}^\kappa$ is randomly sampled after D_5 is determined, by the multivariate Schwartz-Zippel bound,

$$\Pr_{\mathbf{y}}[D_5(\mathbf{y}) = 0 \wedge D_5 \not\equiv 0] \leq \frac{\kappa}{|\mathbb{F}|},$$

which implies that the multilinear identity $D_5(\mathbf{Y}) = 0$ holds except with negligible probability. Hence,

$$\widetilde{p}_{\mathsf{ext}}(\mathbf{r}, \mathbf{Y}) = \widetilde{f}(\mathbf{r}, \mathbf{Y}).$$

We next show that $\widehat{p}_{\mathsf{ext}}$ is exactly the coefficient-repetition extension of $\hat{p}$, thereby linking the univariate and multilinear representations used in Phase-2. Define $H_s(X) = \sum_{j=0}^{s-1} X^j$, and let

$$D_6(X) = \hat{p}_{\mathsf{ext}}(X) - \hat{p}(X^s) H_s(X).$$

Notice that $H_s(X) = (X^s - 1)/(X - 1)$ for $X \neq 1$, while the polynomial definition above is well-defined also at $X = 1$. Both $\hat{p}$ and $\hat{p}_{\mathsf{ext}}$ are fixed before the verifier samples η. Conditioned on $\mathsf{Good}_{\mathsf{PCS}}$, the corresponding PCS evaluation checks imply

$$v_\eta = \hat{p}(\eta^s), \quad \hat{p}_{\mathsf{ext}}(\eta) = v_\eta H_s(\eta).$$

Combining the two equalities gives

$$D_6(\eta) = \hat{p}_{\mathsf{ext}}(\eta) - \hat{p}(\eta^s)H_s(\eta) = 0.$$

Further, since $\deg(\hat{p}) < m$, $\deg(H_s) = s-1$, and $n = ms$, we have

$$\deg\big(\hat{p}(X^s)H_s(X)\big) \le s(m-1) + (s-1) = sm - 1 = n - 1.$$

Moreover, $\deg(\hat{p}_{\mathsf{ext}}) < n$, and hence $\deg(D_6) \le n-1$. Therefore, if $D_6 \not\equiv 0$, the Schwartz–Zippel lemma gives

$$\Pr_{\eta \leftarrow \mathbb{F}}\big[D_6(\eta) = 0 \mid D_6 \not\equiv 0\big] \le \frac{n-1}{|\mathbb{F}|}.$$

Consequently, except with probability at most $(n-1)/|\mathbb{F}|$, acceptance implies

$$\hat{p}_{\mathsf{ext}}(X) = \hat{p}(X^s)H_s(X) = \hat{p}(X^s)\sum_{j=0}^{s-1} X^j.$$

Next, writing $\hat{p}(X) = \sum_{j=0}^{m-1} p_j X^j$, the above identity gives

$$\hat{p}_{\mathsf{ext}}(X) = \left(\sum_{j=0}^{m-1} p_j X^{sj}\right)\left(\sum_{t=0}^{s-1} X^t\right) = \sum_{j=0}^{m-1}\sum_{t=0}^{s-1} p_j X^{sj+t}.$$

Hence the coefficient vector of $\hat{p}_{\mathsf{ext}}$ is obtained by repeating every coefficient of $\hat{p}$ exactly s times. Under the univariate-to-multilinear isomorphism φ^{-1}, it follows that $\widetilde{p}_{\mathsf{ext}}$ is independent of its first ρ variables, and

$$\widetilde{p}_{\mathsf{ext}}(\mathbf{r}, \mathbf{y}) = \widetilde{p}(\mathbf{y}), \qquad \forall\, \mathbf{r} \in \mathbb{F}^\rho,\ \mathbf{y} \in B^\kappa.$$

Recall that $\widetilde{p}_{\mathsf{ext}}(\mathbf{r}, \mathbf{Y}) = \widetilde{f}(\mathbf{r}, \mathbf{Y})$, which implies,

$$\widetilde{p}(\mathbf{Y}) = \widetilde{f}(\mathbf{r}, \mathbf{Y}).$$

Further, due to verifier-derived commitment C_f,

$$\widetilde{f}(\mathbf{x}) = \sum_{i=1}^{\ell} \gamma^{i-1}\widetilde{f}_i(\mathbf{x}) \implies \widetilde{p}(\mathbf{Y}) = \sum_{i=1}^{\ell} \gamma^{i-1}\widetilde{f}_i(\mathbf{r}, \mathbf{Y}) = \sum_{i=1}^{\ell} \gamma^{i-1}\widetilde{p}_i(\mathbf{Y}),$$

where $\widetilde{p}_i(\mathbf{Y})$ denotes $\widetilde{f}_i(\mathbf{r}, \mathbf{Y})$, $\forall i \in [\ell]$.

By the isomorphism φ from Section 3.1, the equivalent univariate polynomial identity can be stated as follows:

$$\hat{p}(X) = \sum_{i=1}^{\ell} \gamma^{i-1}\hat{p}_i(X) \implies \hat{p}[j-1] = \sum_{i=1}^{\ell} \gamma^{i-1}\hat{p}_i[j-1], \forall j \in [1, m].$$

Finally, recall that $\sum_{i=1}^{m} \hat{p}[i-1] \cdot \mu_i^{\mathbb{H}}(z) = \sum_{j=1}^{\ell} \gamma^{j-1} v_j$. This implies

$$\begin{aligned}
\sum_{i=1}^{m}\left(\sum_{j=1}^{\ell} \gamma^{j-1}\hat{p}_j[i-1]\right) \cdot \mu_i^{\mathbb{H}}(z) = \sum_{j=1}^{\ell} \gamma^{j-1} v_j \implies & \sum_{j=1}^{\ell} \gamma^{j-1}\left(\sum_{i=1}^{m} \hat{p}_j[i-1] \cdot \mu_i^{\mathbb{H}}(z)\right) = \sum_{j=1}^{\ell} \gamma^{j-1} v_j, \\
\implies & \sum_{j=1}^{\ell} \gamma^{j-1}(\mathsf{ifft}(\hat{p}_j))(z) = \sum_{j=1}^{\ell} \gamma^{j-1} v_j, \\
\implies & \sum_{j=1}^{\ell} \gamma^{j-1}\hat{q}_j(z) = \sum_{j=1}^{\ell} \gamma^{j-1} v_j,
\end{aligned}$$

Finally, we use the random batching challenge γ to lift the authenticated batched evaluation to the individual claims $\hat{q}_i(z) = v_i$. More concretely, we argue that, except with small probability over the choice of γ, this implies $\hat{q}_i(z) = v_i$ for every $i \in [\ell]$. Define the univariate polynomial

$$D_\gamma(X) = \sum_{i=1}^{\ell} X^{i-1}\big(\hat{q}_i(z) - v_i\big).$$

Observe that the coefficients of D_γ are fixed before the verifier samples γ: the polynomials $\hat{q}_i$ are determined by the extracted witness polynomials and the Phase-0 challenges, while the values v_i and the challenge z are fixed before γ is sampled. Moreover, $\deg(D_\gamma) \le \ell - 1$. So, if there exists an $i \in [\ell]$ such that $\widehat{q}_i(z) \neq v_i$, then D_γ is a nonzero polynomial. The relation above states precisely that $D_\gamma(\gamma) = 0$. Therefore, by the Schwartz-Zippel lemma,

$$\Pr_{\gamma \leftarrow \mathbb{F}}[D_\gamma(\gamma) = 0 \mid D_\gamma \not\equiv 0] \le \frac{\ell - 1}{|\mathbb{F}|}.$$

Consequently, except with probability at most $(\ell - 1)/|\mathbb{F}|$ over the random choice of γ, we have

$$\hat{q}_i(z) = v_i, \forall i \in [\ell],$$

which is exactly the definition of the event E. At this point, we conclude that

$$\Pr[\neg \mathsf{E} | \mathsf{Good}_{\mathsf{PCS}}] < \frac{2}{|\mathbb{F}|} + \frac{2m-2}{|\mathbb{F}|} + \frac{1}{|\mathbb{F}|} + \frac{m-1}{|\mathbb{F}|} + \frac{\kappa}{|\mathbb{F}|} + \frac{n-1}{|\mathbb{F}|} + \frac{\ell-1}{|\mathbb{F}|} = \mathsf{negl}(\lambda),$$

where the probability is over random choices of $(\gamma, r', r, \epsilon, \delta, \mathbf{y}, \eta)$. This concludes the proof of Lemma 4.4.

□

Finally, observe that, from Lemmas 4.3 and 4.4, we get the following: conditioned on the event $\mathsf{Good}_{\mathsf{PCS}}$, the set of extracted witness polynomials $\{\widetilde{f}_i\}_{i\in[\ell]}$ satisfy the sumcheck relation in Equation 1, except with negligible probability. Further, as described above, assuming the (knowledge) soundness of uPCS and mPCS, the event $\neg\mathsf{Good}_{\mathsf{PCS}}$ occurs with only negligible probability. In the AGM, under the q-DLOG assumption (Definition 2.7), uPCS = KZG [KZG10] and mPCS = Samaritan [GPS25] satisfy knowledge soundness. This concludes the proof of knowledge soundness of our sumcheck protocol in Figure 1 as summarized in Theorem 4.1.

4.4.3 Detailed Phase-Wise Efficiency Analysis

Finally, we present a detailed, phase-wide efficiency analysis of the sumcheck argument in Figure 1. Throughout, we instantiate uPCS = KZG [KZG10] and mPCS = Samaritan [GPS25].

Prover-to-Verifier Communication. We first analyze the prover-to-verifier communication of the protocol, which corresponds to the proof size after applying the Fiat-Shamir transform. Verifier challenges are therefore not included in the communication counts below. The following phase-wise analysis yields an overall communication complexity of $O(\ell + d \log\log n)$.

- **Phase-0**: In this phase, $\mathcal{P}$ sends to $\mathcal{V}$ the multilinear commitments $\{C_i\}_{i\in[\ell]}$, which contribute $\ell\mathbb{G}_1$ elements. $\mathcal{P}$ also sends to $\mathcal{V}$ $(d+1)\mathbb{F}$ elements in each of the $\rho = \log\log n$ rounds of the classical multivariate sumcheck protocol. Thus, the total prover-to-verifier communication in Phase-0 is $O(\ell + d\log\log n)$.
- **Phase-1**: In this phase, $\mathcal{P}$ sends to $\mathcal{V}$: (i) uPCS commitments to $\hat{g}(X)$ and $\hat{h}(X)$, contributing $2\mathbb{G}_1$ elements, (ii) evaluations of all $\hat{q}_i(X)$ at $X = z$, contributing $\ell\mathbb{F}$ elements, and (iii) one uPCS evaluation proof, contributing $1\mathbb{G}_1$ element. Thus, the total prover-to-verifier communication in Phase-1 is $O(\ell)$.

Table 2: Comparison of proof sizes for vanilla and improved sumcheck over the BLS12-381 curve using arkworks-0.5.0.

Number of gates (n)	2^{20}	2^{22}	2^{24}	2^{26}	2^{28}	2^{30}
Vanilla sumcheck proof size	1.69KB	1.81KB	1.94KB	2.06KB	2.19KB	2.31KB
Improved sumcheck proof size	1.45KB	1.45KB	1.45KB	1.45KB	1.45KB	1.45KB

- **Phase-2**: In this phase, $\mathcal{P}$ sends to $\mathcal{V}$: (i) uPCS commitments to $\hat{p}(X)$ and $\hat{p}_{ext}(X)$, contributing $2\mathbb{G}_1$ elements, (ii) the evaluation v_η contributing $1\mathbb{F}$ element, (iii) two uPCS proofs, contributing $2\mathbb{G}_1$ elements, and (iv) one mPCS evaluation proof, contributing $(6\mathbb{G}_1 + \mathbb{F})$ elements. Thus, the total prover-to-verifier communication in Phase-2 is $O(1)$.
- **Phase-3**: In this phase, $\mathcal{P}$ sends to $\mathcal{V}$: (i) commitments to $\hat{A}_z(X)$ and $\hat{S}(X)$, contributing $2\mathbb{G}_1$ elements, (ii) v_{Ainv}, v_{pinv}, and v_{Sinv}, contributing $3\mathbb{F}$ elements and (iii) two uPCS proofs, contributing $2\mathbb{G}_1$ elements, and (iv) $2\mathbb{G}_1$ elements arising from the degree check that combines the degree-check polynomials involved in multilinear PCS instance (Phase 2) as well as $\hat{h}(X)$. Thus, the total prover-to-verifier communication in Phase-3 is $O(1)$.

Thus, Phases 1–3 contribute a total of $O(\ell)$ prover-to-verifier communication, while Phase-0 contributes $O(\ell + d\log\log n)$ prover-to-verifier communication. Overall, the prover-to-verifier communication is therefore $O(\ell + d\log\log n)$, as claimed in Theorem 4.1.

Concrete Proof Sizes. In Table 2, we report the concrete proof sizes obtained by applying the Fiat-Shamir transform to the protocol in Figure 1 and instantiating over BLS12-381, for the following multivariate sumcheck instance

$$\sum_{\mathbf{x}\in B_\mu} \big(a(\mathbf{x})\cdot b(\mathbf{x}) - c(\mathbf{x})\big) = v.$$

We choose the above instance for illustration since it is simple but, at the same time, involves a polynomial multiplication operation. We compare the proof sizes of our improved multivariate sumcheck argument with the proof sizes of the "vanilla" sumcheck argument from Section 3.1 for $n = 2^\mu$ varying in the range $n = 2^{20}$ to $n = 2^{30}$. The proof sizes are reported over the BLS12-381 curve using arkworks-0.5.0. Notably, at $n = 2^{30}$, the concrete proof size for our improved sumcheck argument is $\approx 1.6\times$ smaller than the vanilla sumcheck argument.

Prover Cost. We now provide phase-wise prover cost analysis below and justify the overall prover complexity of $O(n)$. Throughout, let $c = \mathcal{C}(\widetilde{G}, d, \ell)$ be a constant dependent on the multivariate form $\widetilde{G}$ and the values of d and ℓ.

- **Phase-0**: $\mathcal{P}$ computes commitments to ℓ μ-variate multilinear polynomials using [GPS25], which incurs $O(\ell \cdot n)\mathbb{G}_1$ operations. Further, in each round of classical multivariate sumcheck protocol, $\mathcal{P}$ needs to compute the univariate round polynomial, and evaluate it at $(d+1)$ distinct points. The corresponding cost is upper bounded by the cost of the first round, which requires $O(d \cdot n)\mathbb{F}$ operations.
- **Phase-1**: Here, $\mathcal{P}$ first performs ℓ ifft operations to derive $\hat{q}_i(X)$'s from $(m-1)$-degree $\hat{p}_i(X)$'s, incurring $O(\ell m\log m)\mathbb{F}$ operations. Then it computes $\hat{G}(X)$ from $\{\hat{q}_i(X)\}_{i\in[\ell]}$, requiring $c\cdot m\log m)\mathbb{F}$ operations. Deriving $\hat{g}(X)$ and $\hat{h}(X)$ from $\hat{G}(X)$ additionally incurs $O(m\log m)\mathbb{F}$ operations. Further, computing commitments to the univariate polynomials $\hat{g}(X)$ and $\hat{h}(X)$ incurs $O(\ell\cdot m)\mathbb{G}_1$ operations. Evaluating $\{\hat{q}_i(X)\}_{i\in[\ell]}$ at $X = z$ takes $O(\ell\cdot m\log m)\mathbb{F}$ operations. $\mathcal{P}$ also computes one uPCS evaluation proof, incurring $O(\ell\cdot m)\mathbb{F}$ and $O(\ell\cdot m)\mathbb{G}_1$ operations. Plugging in $m = n/\log n$, the total prover cost for Phase-1 turns out to be $O(c\cdot n)\mathbb{F}$ and $O(\ell\cdot n/\log n)\mathbb{G}_1$ operations.
- **Phase-2**: In this phase, $\mathcal{P}$ computes $\hat{p}(X)$ by linearly combining $(m-1)$-degree polynomials $\{\hat{p}_i(X)\})_{i\in[\ell]}$, incurring $O(\ell\cdot m)\mathbb{F}$ operations. $\mathcal{P}$ also computes $(n-1)$-degree polynomial $\hat{p}_{ext}$ from $\hat{p}$, which requires $O(n)\mathbb{F}$ operations. Evaluating $\hat{p}(X)$ at $X = \eta^s$ requires $O(m\log m)\mathbb{F}$ operations. The uPCS evaluation

proof for $\hat{p}(X)$ incurs $O(m)\mathbb{F}$ and $O(m)\mathbb{G}_1$ cost, and the uPCS evaluation proof for $\hat{p}_{ext}$ incurs $O(n)\mathbb{F}$ and $O(n)\mathbb{G}_1$ cost. One mPCS evaluation proof computation requires $O(n)\mathbb{F}$ and $O(n)\mathbb{G}_1$ operations. Thus, the total prover cost for Phase-2 is $O(c \cdot n)\mathbb{F}$ and $O(n)\mathbb{G}_1$ operations.

- **Phase-3**: In this phase, the computation of the $(m-1)$-degree polynomial $\hat{A}_z(X)$ requires $O(m \log m)\mathbb{F}$ operations, while the computation of $\hat{S}(X)$ requires $O(m \log m)\mathbb{F}$ operations by Lemmas 4.1 and 4.2. Evaluating $\hat{A}_z(X)$, $\hat{S}(X)$, $\hat{p}(X)$ at $X = 1/r$ also require $O(m \log m)\mathbb{F}$ operations. Each of the two uPCS evaluation proofs for $\hat{b}(X)$ and $\hat{c}(X)$ incur $O(m)\mathbb{F}$ and $O(m)\mathbb{G}_1$ operations. Finally, the computation of v_{Binv} is $O(\log m)\mathbb{F}$. Thus, the total prover cost for Phase-3 is $O(n)\mathbb{F}$ and $O(n/\log n)\mathbb{G}_1$ operations.

Putting everything together, the total prover cost across all phases turns out to be $O(c \cdot n)$ $\mathbb{F}$-operations and $O(n + \ell n/\log n)$ $\mathbb{G}_1$ operations, as claimed in Theorem 4.1.

Verifier Cost. We now provide phase-wise verifier cost analysis below and justify the overall verifier complexity of $O(\log n)$. Throughout, let $c = \mathcal{C}(\widetilde{G}, d, \ell)$ be a constant dependent on the multivariate form $\widetilde{G}$ and the values of d and ℓ.

- **Phase-0:** Each round of classical multivariate sumcheck requires $\mathcal{V}$ to perform $O(d)\mathbb{F}$ operations. Thus, the total verifier work in Phase-0 is $O(d \cdot \log\log n)\mathbb{F}$ operations.
- **Phase-1:** $\mathcal{V}$ computes $\hat{G}(z)$ from $\{v_i\}_{i\in[\ell]}$ requiring $O(c)\mathbb{F}$ operations. $\mathcal{V}$ also computes C' from C_g and C_h using $O(1)\mathbb{G}_1$ operations. Thus, the total verifier work in Phase-1 is $O(c)\mathbb{F}$ operations and $O(1)\mathbb{G}_1$ operations.
- **Phase-2:** $\mathcal{V}$'s computation of C_f and C'' require $O(\ell)\mathbb{G}_1$ and $O(1)\mathbb{G}_1$ operations, respectively.
- **Phase-3:** $\mathcal{V}$'s computation of v' requires $O(\ell)\mathbb{F}$ operations, computation of v_{Binv} and v_{B} requires $O(\log m)\mathbb{F}$ operations, and computation of C''' from C_p, C_A and C_S incurs $O(1)\mathbb{G}_1$ cost. Thus, the total verifier work in Phase-3 is $O(\ell + \log m)\mathbb{F}$ operations and $O(1)\mathbb{G}_1$ operations.
- Finally, the verification checks in Lines 28-34 of Figure 1 comprise a constant number of univariate PCS evaluations, one multilinear PCS evaluation, and one degree check. Together, these contribute $O(\log n)\mathbb{F}$ and $O(1)\mathbb{G}_1$ operations.

Putting everything together, the total verifier cost across all phases turns out to be $O(d \cdot \log\log n + c + \ell + \log n)\mathbb{F}$ and $O(\ell)\mathbb{G}_1$ operations, as claimed in Theorem 4.1. For fixed d, ℓ, and $\widetilde{G}$, this simplifies to $O(\log n)$ verifier complexity.

Summary. Combining the above phase-wise bounds, the improved sumcheck argument in Figure 1 has $O(\ell + d\log\log n)$ prover-to-verifier communication, $O(n)$ prover complexity, and $O(d\log\log n + c + \ell + \log n)$ verifier complexity, where $c = C(\widetilde{G}, d, \ell)$. In the typical setting where d, ℓ, and the multivariate form $\widetilde{G}$ are fixed, these simplify to $O(\log\log n)$ communication, $O(n)$ prover complexity, and $O(\log n)$ verifier complexity, as claimed in Theorem 4.1.

5 HybridSpartan

5.1 Background: MicroSpartan

We recall the MicroSpartan multilinear PIOP [ZSCZ25] for R1CS satisfiability, which is a variant of the earlier Spartan [Set20] PIOP for the same. Let $m, n \in \mathbb{N}$ and let $A, B, C \in \mathbb{F}^{m\times n}$ be sparse matrices with $O(n)$ non-zero entries. 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. 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 $i = n_A + 1, \ldots, n_A + n_B$: $\mathbf{val}_B[\,i\,] = \mathsf{val}_i$.
- For $i = n_A + n_B + 1, \ldots, N$: $\mathbf{val}_C[\,i\,] = \mathsf{val}_i$.

For all of 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 the above vectors. Let $\mathbf{z}$ be R1CS witness satisfying $A\mathbf{z} \circ B\mathbf{z} = C\mathbf{z}$. Henceforth, with suitable zero-padding, we view A, B, C as $N \times N$ matrices and $\mathbf{z} \in \mathbb{F}^N$ satisfying $A\mathbf{z} \circ B\mathbf{z} = C\mathbf{z}$. The prover $\mathcal{P}$ and verifier $\mathcal{V}$ proceed as follows:

1. $\mathcal{P}$ computes $\mathbf{a} = A\mathbf{z}$, $\mathbf{b} = B\mathbf{z}$ and $\mathbf{c} = C\mathbf{z}$. Let $\widetilde{z}, \widetilde{a}, \widetilde{b}$ and $\widetilde{c}$ denote MLEs of vectors $\mathbf{z}, \mathbf{a}, \mathbf{b}$ and $\mathbf{c}$ respectively.
2. $\mathcal{P}$ sends polynomial oracles $[\![\widetilde{z}]\!], [\![\widetilde{a}]\!], [\![\widetilde{b}]\!], [\![\widetilde{c}]\!]$.
3. $\mathcal{V}$ sends $\boldsymbol{\tau} \leftarrow \mathbb{F}^{\log N}$.
4. $\mathcal{P}$ computes vectors $\mathbf{L}_r[\,i\,] = \mathbf{eq}_\tau[\,\mathbf{row}[\,i\,]\,]$ and $\mathbf{L}_c[\,i\,] = \mathbf{z}[\,\mathbf{col}[\,i\,]\,]$. Here $\mathbf{eq}_\tau$ denotes the vector with i^{th} entry as $\widetilde{eq}(\langle\, i \,\rangle, \boldsymbol{\tau})$ for $i \in [N]$.
5. $\mathcal{P}$ sends polynomial oracles $[\![\widetilde{L}_r]\!]$, $[\![\widetilde{L}_c]\!]$.

Next, the following checks are performed on the polynomial oracles:

- *R1CS relation holds*: This amounts to checking that $\widetilde{a}(\mathbf{x}) \cdot \widetilde{b}(\mathbf{x}) - \widetilde{c}(\mathbf{x})$ is 0 for all $\mathbf{x} \in B_{\log N}$. With overwhelming probability, this reduces to the following sumcheck instance:

$$\sum_{\mathbf{x} \in B_{\log N}} \widetilde{eq}(\mathbf{x}, \boldsymbol{\tau}) \left(\widetilde{a}(\mathbf{x}) \cdot \widetilde{b}(\mathbf{x}) - \widetilde{c}(\mathbf{x}) \right) = 0 \tag{14}$$

- *Correctness of $\widetilde{a}$, $\widetilde{b}$, $\widetilde{c}$.* To check $\mathbf{a} = A\mathbf{z}$, $\mathbf{b} = B\mathbf{z}$ and $\mathbf{c} = C\mathbf{z}$, $\mathcal{P}$ and $\mathcal{V}$ define following sumcheck instances:

$$\begin{aligned}
\sum_{\mathbf{x} \in B_{\log N}} \left(\widetilde{L}_r(\mathbf{x}) \cdot \widetilde{val}_A(\mathbf{x}) \cdot \widetilde{L}_c(\mathbf{x}) - \widetilde{eq}(\mathbf{x}, \boldsymbol{\tau}) \cdot \widetilde{a}(\mathbf{x}) \right) = 0 \\
\sum_{\mathbf{x} \in B_{\log N}} \left(\widetilde{L}_r(\mathbf{x}) \cdot \widetilde{val}_B(\mathbf{x}) \cdot \widetilde{L}_c(\mathbf{x}) - \widetilde{eq}(\mathbf{x}, \boldsymbol{\tau}) \cdot \widetilde{b}(\mathbf{x}) \right) = 0 \\
\sum_{\mathbf{x} \in B_{\log N}} \left(\widetilde{L}_r(\mathbf{x}) \cdot \widetilde{val}_C(\mathbf{x}) \cdot \widetilde{L}_c(\mathbf{x}) - \widetilde{eq}(\mathbf{x}, \boldsymbol{\tau}) \cdot \widetilde{c}(\mathbf{x}) \right) = 0
\end{aligned} \tag{15}$$

- *Correctness of Lookups*: Finally, to check that vectors $\mathbf{L}_r$ and $\mathbf{L}_c$ correspond to correct lookups on vectors $\mathbf{eq}_\tau$ and $\mathbf{z}$ respectively, we follow the "logarithmic derivative" approach of [Hab22]. The prover sends oracles to multiplicity polynomials $\widetilde{m}_r$, $\widetilde{m}_c$ and shows the equality of the rational functions:

$$\begin{aligned}
\sum_{\mathbf{x} \in B_{\log N}} \frac{\widetilde{m}_r(\mathbf{x})}{X + \widetilde{id}(\mathbf{x}) \cdot Y + \widetilde{eq}(\mathbf{x}, \boldsymbol{\tau})} = \sum_{\mathbf{x} \in B_{\log N}} \frac{1}{X + \widetilde{row}(\mathbf{x}) \cdot Y + \widetilde{L}_r(\mathbf{x})} \\
\sum_{\mathbf{x} \in B_{\log N}} \frac{\widetilde{m}_c(\mathbf{x})}{X + \widetilde{id}(\mathbf{x}) \cdot Y + \widetilde{z}(\mathbf{x})} = \sum_{\mathbf{x} \in B_{\log N}} \frac{1}{X + \widetilde{col}(\mathbf{x}) \cdot Y + \widetilde{L}_c(\mathbf{x})}
\end{aligned} \tag{16}$$

The verifier probabilistically checks the above at $X = \alpha$, $Y = \beta$ for $\alpha, \beta \leftarrow \mathbb{F}$. At this point verifier sends $\gamma \leftarrow \mathbb{F}$, and prover sends $\widetilde{g}(\mathbf{x})$ which interpolates the rational function:

$$\widetilde{g}(\mathbf{x}) = \chi_1(\mathbf{x}) + \gamma \cdot \chi_2(\mathbf{x}) \text{ where:}$$
$$\chi_1(\mathbf{x}) = \frac{\widetilde{m}_r(\mathbf{x})}{\alpha + \beta \cdot \widetilde{id}(\mathbf{x}) + \widetilde{eq}(\mathbf{x}, \boldsymbol{\tau})} - \frac{1}{\alpha + \beta \cdot \widetilde{row}(\mathbf{x}) + \widetilde{L}_r(\mathbf{x})}$$
$$\chi_2(\mathbf{x}) = \frac{\widetilde{m}_c(\mathbf{x})}{\alpha + \beta \cdot \widetilde{id}(\mathbf{x}) + \widetilde{z}(\mathbf{x})} - \frac{1}{\alpha + \beta \cdot \widetilde{col}(\mathbf{x}) + \widetilde{L}_c(\mathbf{x})}$$

The verifier then sends $\boldsymbol{\tau}' \leftarrow \mathbb{F}^{\log N}$ and defines the following sumcheck instances:

$$\sum_{\mathbf{x} \in B_{\log N}} \widetilde{eq}(\mathbf{x}, \boldsymbol{\tau}') \cdot (\widetilde{D}(\mathbf{x})\widetilde{g}(\mathbf{x}) - \widetilde{N}(\mathbf{x})) = 0, \qquad \sum_{\mathbf{x} \in B_{\log N}} \widetilde{g}(\mathbf{x}) = 0 \tag{17}$$

where $\widetilde{N}(\mathbf{x})$ and $\widetilde{D}(\mathbf{x})$ are such that $\widetilde{\chi}_1(\mathbf{x}) + \gamma \cdot \widetilde{\chi}_2(\mathbf{x}) = \widetilde{N}(\mathbf{x})/\widetilde{D}(\mathbf{x})$.

Using a random challenge, all the sumcheck instances in Equations (14), (15) and (17) can be aggregated into a single sumcheck instance of the form

$$\bar{G}\big(\widetilde{row}, \widetilde{col}, \widetilde{val_A}, \widetilde{val_B}, \widetilde{val_C}, \widetilde{z}, \widetilde{a}, \widetilde{b}, \widetilde{c}, \widetilde{L}_r, \widetilde{L}_c, \widetilde{m}_r, \widetilde{m}_c, \widetilde{g}, \widetilde{id}, \widetilde{eq}(\cdot, \boldsymbol{\tau}), \widetilde{eq}(\cdot, \boldsymbol{\tau}')\big) = 0 \tag{18}$$

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

5.2 HybridSpartan via Improved Sumcheck

We compile the aforementioned PIOP into a SNARK as follows: (i) we plug in our improved sumcheck argument from Theorem 4.1 for the sumcheck instance 18 (which incurs $O(\log \log n)$ proof size and $O(n)$ prover work), (ii) we realize the multilinear polynomial oracles using Samaritan [GPS25] (which also relies on the univariate KZG PCS [KZG10], but achieves $O(1)$ proof size). We call the resulting (Fiat-Shamir transformed) SNARK HybridSpartan. Concretely, we state the following theorem.

Theorem 5.1 (HybridSpartan)**.** *Compiling the MicroSpartan multilinear PIOP from Section 5.1 using our sumcheck argument from Figure 1 and* Samaritan *[GPS25] yields a succinct argument of knowledge for the satisfiability of $O(n)$ R1CS constraints that satisfies completeness and knowledge soundness in the AGM under the q-DLOG assumption (Definition 2.7), with (i) $O(\log \log n)$ communication, (ii) $O(n)$ prover complexity, and (iii) $O(\log n)$ verifier complexity. Further, applying the Fiat–Shamir transform [FS87] yields the corresponding* HybridSpartan *SNARK which is secure in combined ROM + AGM setting, and has $O(\log \log n)$ proof size.*

5.3 Concrete Proof Size of HybridSpartan

In this subsection, we present a detailed analysis of the concrete proof size of HybridSpartan. Since the MicroSpartan PIOP boils down to an instance of multivariate sumcheck protocol, with 17 multilinear polynomials, as given in Equation 18, and the maximum degree of a variable in the multivariate sumcheck instance is 6, following is the elaborated proof size analysis of HybridSpartan:

- HybridSpartan prover sends commitments to the following multilinear polynomials: $\widetilde{z}, \widetilde{a}, \widetilde{b}, \widetilde{c}, \widetilde{L}_r, \widetilde{L}_c, \widetilde{m}_r, \widetilde{m}_c, \widetilde{g}$. The other ones are either committed during a preprocessing phase (e.g., $\widetilde{row}, \widetilde{col}, \widetilde{val_A}, \widetilde{val_B}, \widetilde{val_C}, \widetilde{id}$) or their commits are not computed, since those multilinear polynomials (e.g., $\widetilde{eq}(\cdot, \boldsymbol{\tau}), \widetilde{eq}(\cdot, \boldsymbol{\tau}')$) can be efficiently evaluated in verifier end. So, HybridSpartan proof size is $9\mathbb{G}_1 + |\pi_{\mathsf{sumcheck}}|$.
- The improved sumcheck prover contributes $6 \log \log n \mathbb{F}$ in initial sumcheck phase, $3\mathbb{G}_1 + 17\mathbb{F}$ in Phase 1, $4\mathbb{G}_1 + 1\mathbb{F} + |\pi_{\mathsf{mlpcs}}|$ in Phase 2, $4\mathbb{G}_1 + 3\mathbb{F}$ in Phase 3, and $2\mathbb{G}_1$ for degree check (Section 4).

- When we instantiate the improved sumcheck with Samaritan PCS [GPS25], we get $|\pi_{\mathsf{mlpcs}}| = (6\mathbb{G}_1 + \mathbb{F})$.

Summing up, the proof size is $6 \log\log n\mathbb{F} + 28\mathbb{G}_1 + 22\mathbb{F}$. Using the optimization from [CBBZ23, Section 3.1], we further reduce the proof size to $\log\log n(1\mathbb{G}_1 + 1\mathbb{F}) + 28\mathbb{G}_1 + 22\mathbb{F}$. For $n = 2^{30}$, HybridSpartan's proof size is:

- 1.88KB over BN254 curve, with $|\mathbb{G}_1| = 32$ bytes, $|\mathbb{F}| = 32$ bytes.
- 2.39KB over BLS12-381 curve, with $|\mathbb{G}_1| = 48$ bytes, $|\mathbb{F}| = 32$ bytes.

In particular, for $n = 2^{30}$, HybridSpartan's proof size over BN254 is $\approx 3.6\times$ smaller than the proof size of Microspartan [ZSCZ25] over BN254.

6 HybridPlonk

6.1 Background: HyperPlonk

HyperPlonk [CBBZ23] is a multilinear PIOP which is an adaptation of Plonk [GWC19], a popular univariate PIOP. HyperPlonk leverages sumcheck protocol to achieve efficient prover compared to Plonk. We describe the variant of HyperPlonk for gates with 3 wires, and 5 selectors. Let $\mu \in \mathbb{N}$ and let $n = 2^\mu$ be the number of gates. We specify the HyperPlonk circuit $\mathcal{C}$ as below:

- Vectors $\mathbf{q}_M, \mathbf{q}_L, \mathbf{q}_R, \mathbf{q}_O$ and $\mathbf{q}_C$ in $\mathbb{F}^n$ denote the selectors.
- Permutation $\sigma : [3n] \to [3n]$ denotes the wiring constraints.
- The vector $\mathbf{z} = (z_1, \ldots, z_{3n})$ denotes the wires of the circuit. Without loss of generality, we assume $z_1, \ldots, z_k$ denote the k public input wires.

We say that $\mathbf{z} \in \mathbb{F}^{3n}$ satisfies $(\mathcal{C}, \mathbf{x})$ for $\mathbf{x} \in \mathbb{F}^k$ if it satisfies:

1. Input Constraints: $z_i = x_i$ for $i \in [k]$.
2. Gate Constraints: $\mathbf{q}_{M,i} \cdot z_i \cdot z_{i+n} + \mathbf{q}_{L,i} \cdot z_i + \mathbf{q}_{R,i} \cdot z_{i+n} + \mathbf{q}_{O,i} \cdot z_{i+2n} + \mathbf{q}_{C,i} = 0$ for all $i \in [n]$.
3. Wiring Constraints: $z_{\sigma(i)} = z_i$ for $i \in [3n]$.

The HyperPlonk indexer outputs the following multilinear polynomials corresponding to the circuit $\mathcal{C}$:

- For each $X \in \{M, L, R, O, C\}$, the multilinear polynomial $\widetilde{q}_X$ satisfying $\widetilde{q}_X(\langle i \rangle) = \mathbf{q}_{X,i}$ for $i \in [n]$.
- Multilinear polynomials $\widetilde{\sigma}_j$ satisfying $\widetilde{\sigma}_j(\langle i \rangle) = \sigma((j-1)n + i)$ for $j \in [3]$.
- Multilinear polynomials $\widetilde{\mathsf{id}}_j$, $j \in [3]$ for the identity permutation satisfying $\widetilde{\mathsf{id}}_j(\langle i \rangle) = i + (j-1)n$.

We note that the polynomials for identity permutation can be computed by the verifier in $O(\log n)$ $\mathbb{F}$-operations, so indexer need not output them.

6.1.1 HyperPlonk PIOP

We now describe the online phase of the HyperPlonk PIOP [CBBZ23] between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$. To show knowledge of $\mathbf{z} \in \mathbb{F}^{3n}$ satisfying $(\mathcal{C}, \mathbf{x})$ the prover $\mathcal{P}$ encodes the witness as multilinear polynomials $\widetilde{w}_1, \widetilde{w}_2$ and $\widetilde{w}_3$ satisfying $\widetilde{w}_1(\langle i \rangle) = z_i$, $\widetilde{w}_2(\langle i \rangle) = z_{i+n}$, $\widetilde{w}_3(\langle i \rangle) = z_{i+2n}$. As a technical point, $\mathcal{P}$ sends z_i as 0 for $i \in [k]$ to accommodate public inputs. It then sends oracles $[\![\widetilde{w}_1]\!]$, $[\![\widetilde{w}_2]\!]$ and $[\![\widetilde{w}_3]\!]$. The verifier $\mathcal{V}$

computes the multilinear polynomial $\widetilde{I}$ for the public inputs satisfying $\widetilde{I}(\langle i \rangle) = x_i$ for $i \in [k]$ and 0 elsewhere. It adjusts the oracle $[\![\widetilde{w}_1]\!]$ sent by $\mathcal{V}$ to $[\![\widetilde{w}_1]\!] + [\![\widetilde{I}]\!]$. Henceforth, we will use $[\![\widetilde{w}_1]\!]$ to denote this "adjusted" oracle. The PIOP proceeds as follows:

- $\mathcal{P}$ sends the witness oracles $[\![\widetilde{w}_i]\!]$, $i \in [3]$ as above.
- $\mathcal{V}$ sends challenges $\beta, \gamma \leftarrow \mathbb{F}$.
- On receiving the challenges, $\mathcal{P}$ computes $\mu + 1$ variate oracle $\widetilde{v}$ satisfying the following for all $\mathbf{x} \in B_\mu$.

$$\widetilde{v}(0, \mathbf{x}) = \prod_{i=1}^{3} \left(\frac{\widetilde{w}_i(\mathbf{x}) + \beta \cdot \widetilde{\mathsf{id}}_i(\mathbf{x}) + \gamma}{\widetilde{w}_i(\mathbf{x}) + \beta \cdot \widetilde{\sigma}_i(\mathbf{x}) + \gamma} \right), \quad \widetilde{v}(1, \mathbf{x}) = \widetilde{v}(\mathbf{x}, 0) \cdot \widetilde{v}(\mathbf{x}, 1).$$

- $\mathcal{P}$ computes μ-variate polynomials $\widetilde{v}_0$ and $\widetilde{v}_1$where $\widetilde{v}_b(\mathbf{X}) = \widetilde{v}(b, \mathbf{X})$ for $b \in \{0, 1\}$. $\mathcal{P}$ also computes μ-variate polynomials $\widetilde{u}_0$ and $\widetilde{u}_1$ where $\widetilde{u}_b(\mathbf{X}) = \widetilde{v}(\mathbf{X}, b)$ for $b \in \{0, 1\}$.
- $\mathcal{V}$ sends batching challenge $\xi \leftarrow \mathbb{F}$ and a zero-check challenge $\tau \leftarrow \mathbb{F}^\mu$.
- $\mathcal{P}$ and $\mathcal{V}$ execute the sumcheck PIOP:

$$\sum_{\mathbf{x} \in B_\mu} \widetilde{eq}_\mu(\mathbf{x}, \tau) \cdot \widetilde{G}(\mathbf{x}) = 0,$$

where $\widetilde{G}$ is defined as:

$$\begin{aligned} \widetilde{G}(\mathbf{x}) = \widetilde{q}_M(\mathbf{x})\widetilde{w}_1(\mathbf{x})\widetilde{w}_2(\mathbf{x}) + \widetilde{q}_L(\mathbf{x})\widetilde{w}_1(\mathbf{x}) + \widetilde{q}_R(\mathbf{x})\widetilde{w}_2(\mathbf{x}) + \\ \widetilde{q}_O(\mathbf{x})\widetilde{w}_3(\mathbf{x}) + \widetilde{q}_C(\mathbf{x}) + \xi \cdot \widetilde{v}_0(\mathbf{x}) \prod_{i=1}^{3} (\widetilde{w}_i(\mathbf{x}) + \beta\widetilde{\sigma}_i(\mathbf{x}) + \gamma) - \\ \xi \cdot \prod_{i=1}^{3} (\widetilde{w}_i(\mathbf{x}) + \beta\widetilde{\mathsf{id}}_i(\mathbf{x}) + \gamma) + \xi^2 \cdot (\widetilde{v}_1(\mathbf{x}) - \widetilde{u}_0(\mathbf{x}) \cdot \widetilde{u}_1(\mathbf{x})) \,. \end{aligned} \tag{19}$$

As noted in [CBBZ23], the oracles $[\![\widetilde{u}_b]\!]$ for $b \in \{0, 1\}$ can be simulated using the oracles $[\![\widetilde{v}_0]\!]$ and $[\![\widetilde{v}_1]\!]$. We will however require the prover to explicitly commit to oracles $[\![\widetilde{u}_b]\!]$ instead of simulating them to reduce the number of multilinear oracle queries.

6.2 HybridPlonk via Improved Sumcheck

In this subsection, we show how to compile an adaptation of the HyperPlonk multilinear PIOP [CBBZ23] with our improved sumcheck argument and Samaritan, along with some additional univariate checks (described below). The resulting (Fiat-Shamir transformed) SNARK, which we call HybridPlonk, is captured by the following theorem.

Theorem 6.1 (HybridPlonk)**.** *Compiling an adaptation of the HyperPlonk PIOP using our sumcheck argument from Figure 1 and* Samaritan *[GPS25] yields a succinct argument of knowledge for the satisfiability of $O(n)$ Plonkish constraints that satisfies completeness and knowledge soundness in the AGM under the q-DLOG assumption (Definition 2.7), with (i) $O(\log \log n)$ communication, (ii) $O(n)$ prover complexity, and (iii) $O(\log n)$ verifier complexity. Further, applying the Fiat–Shamir transform [FS87] yields the corresponding* HybridPlonk *SNARK which is secure in combined ROM + AGM setting, and has $O(\log \log n)$ proof size.*

We now present a detailed description of HybridPlonk. As described in Section 6.1, the HyperPlonk PIOP consists of: (i) one sumcheck instance represented by $\sum_{\mathbf{x} \in B_\mu} \widetilde{eq}_\mu(\mathbf{x}, \tau) \cdot \widetilde{G}(\mathbf{x}) = 0$, and (ii) a check for

correct construction of $\widetilde{u}_b$ and $\widetilde{v}_b$ for $b \in \{0,1\}$. The sumcheck instance is a function of the following set of multilinear polynomials,

$$\big(\widetilde{eq}(\cdot,\tau), \widetilde{\mathsf{id}}_1, \widetilde{\mathsf{id}}_2, \widetilde{\mathsf{id}}_3, \widetilde{q}_M, \widetilde{q}_L, \widetilde{q}_R, \widetilde{q}_O, \widetilde{q}_C, \widetilde{\sigma}_1, \widetilde{\sigma}_2, \widetilde{\sigma}_3, \widetilde{w}_1, \widetilde{w}_2, \widetilde{w}_3, \widetilde{v}_0, \widetilde{v}_1, \widetilde{u}_0, \widetilde{u}_1\big),$$

which we prove by directly invoking the improved sumcheck protocol from Section 4. Note that the sumcheck instance has 19 multilinear polynomials, and the combined multivariate polynomial has degree ≤ 5.

Next, we elaborate on how to prove correct construction of $\widetilde{u}_0$, $\widetilde{u}_1$, $\widetilde{v}_0$, $\widetilde{v}_1$, by formulating a suitable relation among them and then prove it using one univariate PCS query. Note that $\widetilde{u}_0$ and $\widetilde{u}_1$, if constructed correctly must satisfy,

$$\widetilde{u}_b(X_1,\ldots,X_\mu) = (1-X_1)\widetilde{v}_0(X_2,\ldots,X_\mu,b) + X_1\widetilde{v}_1(X_2,\ldots,X_\mu,b). \tag{20}$$

To prove the above, we define $(\mu+1)$-variate polynomial:

$$\widetilde{v}(X_1,\ldots,X_{\mu+1}) = (1-X_1)\widetilde{v}_0(X_2,\ldots,X_{\mu+1}) + X_1\widetilde{v}_1(X_2,\ldots,X_{\mu+1}). \tag{21}$$

Then we have $\widetilde{v}_b(\mathbf{X}) = \widetilde{v}(b,\mathbf{X})$ and $\widetilde{u}_b(\mathbf{X}) = \widetilde{v}(\mathbf{X},b)$ for $b \in \{0,1\}$. Interpreting the preceding relations combinatorially, the $2n$ coefficients of $\widetilde{v}$ contain the n coefficients of $\widetilde{v}_0$ at *even* positions and the n coefficients of $\widetilde{v}_1$ at *odd* positions. In terms of their corresponding univariate polynomials, this is equivalent to $\hat{v}(X) = \hat{v}_0(X^2) + X \cdot \hat{v}_1(X^2)$.

We observe that since $\widetilde{u}_b$ is determined by restricting the most significant bit, coefficients of $\widetilde{u}_0$ constitute the first n coefficients of $\widetilde{v}$, while coefficients of $\widetilde{u}_1$ constitute the latter n coefficients of $\widetilde{v}$. In terms of univariate polynomials, this is equivalent to $\hat{v}(X) = \hat{u}_0(X) + X^n \cdot \hat{u}_1(X)$. Equating, the correctness of polynomials $\widetilde{u}_b$, $b \in \{0,1\}$ is checked using the univariate identity:

$$\hat{u}_0(X) + X^n \cdot \hat{u}_1(X) = \hat{v}_0(X^2) + X \cdot \hat{v}_1(X^2). \tag{22}$$

This identity is verified using one univariate PCS query at a verifier-chosen random point, which contributes $(1\mathbb{G}_1 + 1\mathbb{F})$ elements.

6.3 Concrete Proof Size of HybridPlonk

In this subsection, we present a detailed analysis of the concrete proof size of HybridPlonk. Note that the HybridPlonk prover sends commitments to seven multilinear polynomials: $\widetilde{w}_1$, $\widetilde{w}_2$, $\widetilde{w}_3$, $\widetilde{u}_0$, $\widetilde{u}_1$, $\widetilde{v}_0$, and $\widetilde{v}_1$. The remaining polynomials are known publicly; hence, the prover need not commit to them. Overall, HybridPlonk's proof size is $7\mathbb{G}_1 + |\pi_{\mathsf{sumcheck}}| + (1\mathbb{G}_1 + 1\mathbb{F})$, where the cost of $(1\mathbb{G}_1 + 1\mathbb{F})$ is incurred by the identity check involving $\widetilde{u}_b$ and $\widetilde{v}_b$ for $b \in \{0,1\}$. Following the same analysis of $|\pi_{\mathsf{sumcheck}}|$ from Section 5.3, we conclude that proof size of HybridPlonk is $\log\log n(1\mathbb{G}_1 + 1\mathbb{F}) + 27\mathbb{G}_1 + 25\mathbb{F}$. For $n = 2^{30}$, HybridPlonk's proof size is:

- 2.44KB over BLS12-381 curve, with $|\mathbb{G}_1| = 48$ bytes, $|\mathbb{F}| = 32$ bytes.
- 1.94KB over BN254 curve, with $|\mathbb{G}_1| = 32$ bytes, $|\mathbb{F}| = 32$ bytes.

In particular, for $n = 2^{30}$, HybridPlonk's proof size over BLS12-381 is $\approx 2.4\times$ smaller than the proof size of HyperPlonk [CBBZ23] compiled with Samaritan PCS [GPS25] over the BLS12-381 curve. We record the above polynomial identity to be checked by the verifier.

7 Implementation and Evaluation

In this section, we evaluate our implementations of HybridSpartan and HybridPlonk, and compare them with three state-of-the-art prover-efficient SNARKs, namely HyperPlonk [CBBZ23] compiled with Samaritan

Table 3: Comparison of proof sizes for various pairing-based SNARKs over BLS12-381. Plonk [GWC19] is not prover-efficient and is only included for reference.

Number of gates (n)	2^{20}	2^{22}	2^{24}	2^{26}	2^{28}	2^{30}
Plonk [GWC19]	0.62 KB	0.62 KB	0.62 KB	0.62 KB	0.62 KB	0.62 KB
Spartan [Set20]	234.36 KB	394.58 KB	703.1KB	>1 MB	>1 MB	>1 MB
HyperPlonk [CBBZ23] + Samaritan [GPS25]	4.22 KB	4.54 KB	4.86 KB	5.18KB	5.5KB	5.82KB
HybridSpartan (this work)	2.39 KB	2.39 KB	2.39 KB	2.39 KB	2.39 KB	2.39 KB
HybridPlonk (this work)	2.44 KB	2.44 KB	2.44 KB	2.44 KB	2.44 KB	2.44 KB

Table 4: Comparison of proof sizes for various pairing-based prover-efficient SNARKs over BN254.

Number of gates (n)	2^{20}	2^{22}	2^{24}	2^{26}	2^{28}	2^{30}
Spartan [Set20]	176.6 KB	284.69 KB	489.6KB	>0.5 MB	>1 MB	>1 MB
HyperPlonk [CBBZ23] + Samaritan [GPS25]	3.97 KB	4.28 KB	4.59 KB	4.91KB	5.22KB	5.53KB
MicroSpartan [ZSCZ25]	4.96 KB	5.34 KB	5.73 KB	6.11 KB	6.5 KB	6.88 KB
HybridSpartan (this work)	1.88 KB	1.88 KB	1.88 KB	1.88 KB	1.88 KB	1.88 KB
HybridPlonk (this work)	1.94 KB	1.94 KB	1.94 KB	1.94 KB	1.94 KB	1.94 KB

[GPS25] (abbreviated as HyperPlonk + Samaritan from here on), Spartan [Set20] and MicroSpartan [ZSCZ25]. In particular, we point out that HyperPlonk + Samaritan and MicroSpartan are the two prover-efficient SNARKs with the smallest proof sizes currently. Our artifacts are available at: `https://github.com/SayaniSinha97/HybridSNARK`.

Implementation Details. We implement all of HybridSpartan, HybridPlonk, and Samaritan [GPS25] in Rust using the `arkworks-0.5.0` cryptographic libraries, and benchmark our constructions over both the BLS12-381 and BN254 curves. Over BLS12-381, we have $|\mathbb{G}_1| = 48$ bytes and $|\mathbb{F}| = 32$ bytes, while over BN254, $|\mathbb{G}_1| = |\mathbb{F}| = 32$ bytes. For comparison, we use the publicly available implementation of Spartan from `https://github.com/arkworks-rs/spartan/tree/e0aff4c` based on `arkworks-0.5.0`, and the public MicroSpartan implementation from the Nova repository (available at `https://github.com/microsoft/Nova/tree/148a2d4`), which is optimized using `halo2curves-0.9.0` over BN254. We benchmark HyperPlonk compiled with Samaritan, HybridSpartan, and HybridPlonk over both curves; MicroSpartan is evaluated only over BN254, as its public implementation does not support a flexible choice of curves. This enables same-curve comparisons between our constructions and HyperPlonk + Samaritan over both curves, as well as a direct BN254 comparison between our constructions and MicroSpartan.

Platform. All benchmarks reported in this section were run on an Intel(R) Xeon(R) Silver 4214R CPU with 2.40GHz of clock frequency, 48 cores, and 128GB RAM, running Ubuntu 22.04. To highlight protocol-level comparison, all benchmarks are reported for single-threaded implementation. We constrain all benchmarks to use a single CPU core. We leave comparison studies with parallelized implementations to future work.

Proof Size. Tables 3 and 4 compare the proof sizes of the evaluated SNARKs over BLS12-381 and BN254, respectively, as the number of gates/constraints varies from $n = 2^{20}$ to $n = 2^{30}$. For reference, Table 3 also includes Plonk [GWC19], which is not prover-efficient but achieves a very small constant proof size.

Observe that, across both curves, the proof sizes of HybridSpartan and HybridPlonk remain essentially independent of n over the evaluated range, reflecting their $O(\log \log n)$ asymptotic dependence. In contrast, the proof sizes of the prover-efficient baselines grow with n. At $n = 2^{30}$, HybridSpartan and HybridPlonk have proof sizes of 2.39 KB and 2.44 KB, respectively, over BLS12-381, compared with 5.82 KB for HyperPlonk + Samaritan. Over BN254, their proof sizes are 1.88 KB and 1.94 KB, respectively, compared with 5.53 KB for HyperPlonk + Samaritan and 6.88 KB for MicroSpartan. In particular, HybridSpartan is approximately $3.6\times$ smaller than MicroSpartan over BN254, while HybridPlonk is approximately $2.4\times$ smaller than HyperPlonk + Samaritan over BLS12-381.

These results demonstrate that the asymptotic reduction from $O(\log n)$ to $O(\log \log n)$ proof size al-

Table 5: Comparison of prover times for various pairing-based prover-efficient SNARKs over BLS12-381.

Number of gates (n)	2^{18}	2^{20}	2^{22}	2^{24}	2^{26}
Spartan [Set20]	63.94s	220.73s	801.72s	3023.85s	−−
HyperPlonk [CBBZ23] + Samaritan[GPS25]	39.34s	144.01s	534.21s	2230.89s	9035.19s
HybridSpartan (this work)	49.83s	178.75s	682.44s	2850.92s	11403.00s
HybridPlonk (this work)	45.61s	165.47s	618.92s	2535.32s	10341.00s

Table 6: Comparison of prover times for various pairing-based prover-efficient SNARKs over BN254.

Number of gates (n)	2^{18}	2^{20}	2^{22}	2^{24}	2^{26}
Spartan [Set20]	38.31s	132.45s	508.63s	1850.01s	−−
HyperPlonk [CBBZ23] + Samaritan[GPS25]	26.93s	100.66s	376.96s	1622.19s	6813.20s
MicroSpartan [ZSCZ25]	93s	337.99s	1320.59s	5776.75s	−−
HybridSpartan (this work)	33.29s	114.36s	446.75s	2021.60s	10108.00s
HybridPlonk (this work)	28.36s	106.19s	396.57s	1632.30s	6529.00s

ready yields substantial concrete savings at instance sizes of practical interest. In particular, the proofs of HybridSpartan and HybridPlonk are at least two orders of magnitude smaller than those of Spartan at the largest instance sizes considered, while also substantially improving over the most compact prover-efficient baselines.

Prover Time. Tables 5 and 6 report the single-threaded prover times over BLS12-381 and BN254, respectively, for instance sizes ranging from $n = 2^{18}$ to $n = 2^{26}$. Across both curves, HybridSpartan and HybridPlonk exhibit the expected linear scaling with n.

Over BLS12-381, the most relevant comparison is between HyperPlonk + Samaritan and HybridPlonk, since both use the same underlying curve and PCS implementation. HybridPlonk incurs a modest prover-time overhead from replacing vanilla sumcheck with our improved sumcheck; concretely, its prover is approximately 14% slower than HyperPlonk + Samaritan across the evaluated range. HybridSpartan exhibits comparable scaling, while Spartan does not scale to the largest instance size considered.

Over BN254, the comparison additionally includes MicroSpartan. HybridSpartan is approximately 2.8-2.9× faster than MicroSpartan across the instance sizes for which both systems complete, while simultaneously producing substantially smaller proofs. HybridPlonk remains only moderately slower than HyperPlonk + Samaritan, consistent with the corresponding BLS12-381 measurements. Spartan and MicroSpartan do not scale to $n = 2^{26}$.

Comparison with MicroSpartan. To better understand the prover-time gap between HybridSpartan and MicroSpartan in Table 6, we examine the dominant multi-scalar multiplication (MSM) costs in the two implementations. Although MicroSpartan and HybridSpartan are implemented using `halo2curves` and `arkworks`, respectively, we find that the cost of an individual size-n MSM is comparable across the two libraries in our benchmark setting. The main difference instead lies in the number and structure of the MSMs performed by the prover.

The MicroSpartan prover performs 13 online MSMs of size n, four of which arise from the HyperKZG evaluation proof; among these, 11 involve arbitrary scalar vectors. In comparison, HybridSpartan performs 10 size-n MSMs, of which one arises from the Samaritan evaluation proof. A black-box invocation of Samaritan would require an additional size-n MSM for its degree check; however, our implementation batches this check with the degree check already required by HybridSpartan, avoiding this additional cost. Moreover, four of HybridSpartan's 10 MSMs involve scalar vectors with small entries, leaving only 6 size-n MSMs with arbitrary scalars, compared with 11 for MicroSpartan.

Since size-n MSMs constitute a dominant component of prover time, the smaller number of arbitrary-scalar MSMs provides a plausible explanation for the lower prover time of HybridSpartan observed in Table 6. We emphasize, however, that the two systems use different implementation libraries, and hence the measured

speedup should be interpreted as an end-to-end implementation comparison rather than as a purely protocol-level speedup.

Verifier Time. Spartan has $O(\log^2 n)$ verifier complexity, whereas all of HyperPlonk + Samaritan, MicroSpartan, HybridSpartan, and HybridPlonk achieve $O(\log n)$ verification. At $n = 2^{20}$, their respective verification times are approximately 850 ms, 11 ms, 8 ms, 18 ms, and 16 ms; here, MicroSpartan and HybridSpartan are benchmarked over BN254, while the remaining schemes use BLS12-381. For the latter four schemes, verification times remain essentially unchanged as n grows; this is because the pairing checks dominate the overall verification cost.

Compared with MicroSpartan, HybridSpartan incurs an approximately 2.25× concrete verification overhead, primarily due to the additional PCS-based consistency checks required by our improved sumcheck argument. Nevertheless, both schemes retain $O(\log n)$ asymptotic verifier complexity, while HybridSpartan improves the proof-size dependence from $O(\log n)$ to $O(\log \log n)$. Thus, our construction trades a moderate increase in concrete verification cost for substantially smaller proofs.

Summary. Overall, HybridSpartan and HybridPlonk substantially reduce proof sizes relative to the most compact prover-efficient baselines for R1CS and Plonkish constraints, respectively. These savings come with comparable prover time for HybridPlonk and significantly lower prover time for HybridSpartan in our implementations, but with additional concrete verification overhead. Our results therefore highlight a proof-size/verification-time trade-off: the improved $O(\log \log n)$ proof-size dependence yields substantial concrete communication and storage savings while retaining $O(\log n)$ asymptotic verifier complexity. This trade-off is potentially attractive in applications where proof transmission and/or proof storage are significant components of deployment cost.

References

[ac22] arkworks contributors. `arkworks` zksnark ecosystem, 2022.

[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.

[ARZR25] Noor Athamnah, Noga Ron-Zewi, and Ron D. Rothblum. Linear prover IOPs in log star rounds. Cryptology ePrint Archive, Paper 2025/1269, 2025.

[BBHR18] 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.

[BCCT12] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 326–349, 2012.

[BCF+25a] Anubhav Baweja, Alessandro Chiesa, Elisabetta Fedele, Giacomo Fenzi, Pratyush Mishra, Tushar Mopuri, and Andrew Zitek-Estrada. Time-space trade-offs for sumcheck. Cryptology ePrint Archive, Paper 2025/1473, 2025.

[BCF+25b] Martijn Brehm, Binyi Chen, Ben Fisch, Nicolas Resch, Ron D. Rothblum, and Hadas Zeilberger. Blaze: Fast SNARKs from interleaved RAA codes. In Serge Fehr and Pierre-Alain Fouque, editors, *EUROCRYPT 2025, Part IV*, volume 15604 of *LNCS*, pages 123–152. Springer, Cham, May 2025.

[BCG+13] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. SNARKs for C: Verifying program executions succinctly and in zero knowledge. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 90–108. Springer, Berlin, Heidelberg, August 2013.

[BCG+14] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from Bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474. IEEE Computer Society Press, May 2014.

[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.

[BCI+13] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. Succinct non-interactive arguments via linear interactive proofs. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 315–333. Springer, Berlin, Heidelberg, March 2013.

[BCJ+24] Foteini Baldimtsi, Konstantinos Kryptos Chalkias, Yan Ji, Jonas Lindstrøm, Deepak Maram, Ben Riva, Arnab Roy, Mahdi Sedaghat, and Joy Wang. zkLogin: Privacy-preserving blockchain authentication with existing credentials. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *ACM CCS 2024*, pages 3182–3196. ACM Press, October 2024.

[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.

[BCTV14] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Succinct non-interactive zero knowledge for a von neumann architecture. In Kevin Fu and Jaeyeon Jung, editors, *USENIX Security 2014*, pages 781–796. USENIX Association, August 2014.

[BFS20] Benedikt Bünz, Ben Fisch, and Alan Szepieniec. Transparent SNARKs from DARK compilers. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 677–706. Springer, Cham, May 2020.

[BGM17] Sean Bowe, Ariel Gabizon, and Ian Miers. Scalable multi-party computation for zk-SNARK parameters in the random beacon model. Cryptology ePrint Archive, Report 2017/1050, 2017.

[BHKM26] Juraj Belohorec, Pavel Hubáček, Aleksi Kalsta, and Kristỳna Mašková. Chopin: Optimal pairing-based multilinear polynomial commitments from bivariate kzg. *Cryptology ePrint Archive*, 2026.

[BS23] Ward Beullens and Gregor Seiler. LaBRADOR: Compact proofs for R1CS from module-SIS. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part V*, volume 14085 of *LNCS*, pages 518–548. Springer, Cham, August 2023.

[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.

[CGG+23] Matteo Campanelli, Nicolas Gailly, Rosario Gennaro, Philipp Jovanovic, Mara Mihali, and Justin Thaler. Testudo: Linear time prover SNARKs with constant size proofs and square root size universal setup. In Abdelrahaman Aly and Mehdi Tibouchi, editors, *LATINCRYPT 2023*, volume 14168 of *LNCS*, pages 331–351. Springer, Cham, October 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.

[CMNW24] Valerio Cini, Giulio Malavolta, Ngoc Khanh Nguyen, and Hoeteck Wee. Polynomial commitments from lattices: Post-quantum security, fast verification and transparent setup. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part X*, volume 14929 of *LNCS*, pages 207–242. Springer, Cham, August 2024.

[DMS24] Michel Dellepere, Pratyush Mishra, and Alireza Shirzad. Garuda and pari: Faster and smaller snarks via equifficient polynomial commitments. *IACR Cryptol. ePrint Arch.*, page 1245, 2024.

[EG25] 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.

[FKL18] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Cham, August 2018.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Berlin, Heidelberg, August 1987.

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 626–645. Springer, Berlin, Heidelberg, May 2013.

[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.

[GPS25] Chaya Ganesh, Sikhar Patranabis, and Nitin Singh. Samaritan: Linear-time prover SNARK from new multilinear polynomial commitments. In Goichiro Hanaoka and Bo-Yin Yang, editors, *ASIACRYPT 2025, Part V*, volume 16249 of *LNCS*, pages 456–489. Springer, Singapore, December 2025.

[Gro10] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 321–340. Springer, Berlin, Heidelberg, December 2010.

[Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Berlin, Heidelberg, May 2016.

[Gru24] Angus Gruen. Some improvements for the PIOP for ZeroCheck. Cryptology ePrint Archive, Report 2024/108, 2024.

[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.

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments. In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 723–732, 1992.

[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.

[Lab17] Protocol Labs. Filecoin: A decentralized storage network. `https://filecoin.io/filecoin.pdf`, 2017.

[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.

[LFKN90] Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. In *31st FOCS*, pages 2–10. IEEE Computer Society Press, October 1990.

[Lip12] Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 169–189. Springer, Berlin, Heidelberg, March 2012.

[Lip13] Helger Lipmaa. Succinct non-interactive zero knowledge arguments from span programs and linear error-correcting codes. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 41–60. Springer, Berlin, Heidelberg, December 2013.

[LKKO20] Seunghwa Lee, Hankyung Ko, Jihye Kim, and Hyunok Oh. vCNN: Verifiable convolutional neural network. Cryptology ePrint Archive, Report 2020/584, 2020.

[LMN25] Christophe Levrat, Tanguy Medevielle, and Jade Nardi. A divide-and-conquer sumcheck protocol. *IACR Communications in Cryptology*, 2(1), 2025.

[LVA+25] Hidde Lycklama, Alexander Viand, Nikolay Avramov, Nicolas Küchler, and Anwar Hithnawi. Artemis: Efficient commit-and-prove snarks for zkml, 2025.

[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.

[Mic94] Silvio Micali. Cs proofs. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 436–453. IEEE, 1994.

[NS24] Ngoc Khanh Nguyen and Gregor Seiler. Greyhound: Fast polynomial commitments from lattices. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part X*, volume 14929 of *LNCS*, pages 243–275. Springer, Cham, August 2024.

[PHGR13] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy*, pages 238–252. IEEE Computer Society Press, May 2013.

[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.

[rol21] An incomplete guide to rollups. `https://vitalik.eth.limo/general/2021/01/05/rollup.html`, 2021.

[SDP22] Nitin Singh, Pankaj Dayama, and Vinayaka Pandit. Zero knowledge proofs towards verifiable decentralized AI pipelines. In Ittay Eyal and Juan A. Garay, editors, *FC 2022*, volume 13411 of *LNCS*, pages 248–275. Springer, Cham, May 2022.

[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.

[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.

[XZS22] Tiancheng Xie, Yupeng Zhang, and Dawn Song. Orion: Zero knowledge proof with linear prover time. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part IV*, volume 13510 of *LNCS*, pages 299–328. Springer, Cham, August 2022.

[XZZ+19] Tiancheng Xie, Jiaheng Zhang, Yupeng Zhang, Charalampos Papamanthou, and Dawn Song. Libra: Succinct zero-knowledge proofs with optimal prover computation. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 733–764. Springer, Cham, August 2019.

[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.

[ZFZS20] Jiaheng Zhang, Zhiyong Fang, Yupeng Zhang, and Dawn Song. Zero knowledge proofs for decision tree predictions and accuracy. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 2039–2053. ACM Press, November 2020.

[ZSCZ25] Jiaxing Zhao, Srinath T. V. Setty, Weidong Cui, and Greg Zaverucha. Micronova: Folding-based arguments with efficient (on-chain) verification. In Marina Blanton, William Enck, and Cristina Nita-Rotaru, editors, *IEEE Symposium on Security and Privacy, SP 2025, San Francisco, CA, USA, May 12-15, 2025*, pages 1964–1982. IEEE, 2025.

[ZXZS20] Jiaheng Zhang, Tiancheng Xie, Yupeng Zhang, and Dawn Song. Transparent polynomial delegation and its applications to zero knowledge proof. In *2020 IEEE Symposium on Security and Privacy*, pages 859–876. IEEE Computer Society Press, May 2020.

A Ethical Considerations

In this paper, we proposed a new, highly compact, prover-efficient SNARK called HybridPlonk with sublogarithmic proof size and provable security in the ROM. Along the way, we reduced the communication complexity of the classical sumcheck protocol for multivariate polynomials from logarithmic to sublogarithmic, while retaining linear prover complexity. We contributed a performant implementation for HybridPlonk, built on top of the popular Arkworks [ac22] ecosystem, and empirically evaluated its performance. We analyze potential harms and benefits to stakeholders that are impacted broadly by the technology researched and developed in this paper using a consequentialist approach.

Our research enables succinct verification of large computations and improves over the state-of-the-art in the sense that our proposed scheme HybridPlonk is: (a) either substantially more prover-efficient than prior SNARKs with sublogarithmic proof size [Gro10, Lip12, BCCT12, BCI+13, GGPR13, PHGR13, BCG+13, Lip13, BCTV14, MBKM19, CHM+20, GWC19, DMS24], or (b) has $2.4-3.6\times$ smaller proof size than state-of-the-art prover-efficient SNARKs [WTs+18, XZZ+19, Set20, SL20, CBBZ23, BCHO22, XZS22, GLS+23].

The main group impacted by our research are potential users of SNARKs. Increased compactness (in terms of proof size) and/or increased prover efficiency broaden the availability of SNARKs as a technology, especially to economically marginalized groups, and potentially broaden the set of real-world applications where SNARKs can be deployed in an efficient and scalable manner. Overall, we believe that our proposal has the potential to enable increased adoption of SNARKs, thus benefiting society at large by enabling efficiently verifiable computation.