Fully Succinct Arguments over the Integers from First Principles*

Matteo Campanelli[1] and Mathias Hall-Andersen[2]

[1] Offchain Labs and University of Tartu
binarywhalesinternaryseas@gmail.com
[2] ZKSecurity
mathias@hall-andersen.dk

Abstract. We construct succinct arguments of knowledge for computations over the ring of integers $\mathbb{Z}$ ($\mathbb{Z}$NARKs). Our motivation stems both from the theoretical challenge of developing general techniques for arguments over the integers and their practical applications. For example, many natural applications can be viewed as special cases of proving over $\mathbb{Z}$—e.g. cryptographic primitives based on RSA groups or Ring-LWE; field emulation and field "switching"; arbitrary-precision arithmetic. Moreover, $\mathbb{Z}$NARKs can support arbitrary mixing and matching of these settings *within the same* computation, which is not easily possible with solutions tailored to a specific arithmetic.

Unlike prior works targeting $\mathbb{Z}$ or $\mathbb{Z}_{2^k}$, we avoid most of the complexities involved in extracting over these rings directly. Instead, we introduce a general, simpler theoretical framework for building succinct arguments over $\mathbb{Z}$, one which allows protocol designers to reuse existing SNARK techniques. This is possible thanks to our key technique—*fingerprinting*, a form of arithmetic hashing—for "bootstrapping" protocols over the integers from existing systems over prime fields (e.g., multilinear-flavored ones, such as Spartan). The resulting protocol can then be compiled into a cryptographic argument via a novel kind of polynomial commitment allowing queries to a multivariate integer polynomial modulo an arbitrary prime q. We show how to instantiate our framework and obtain a concrete scheme, Zaratan. This is the first construction in literature being *fully* succinct over integer computations, i.e., with short proofs and fast verification even when the witness consists of large integers.

* Published in PKC 2026 under the title *General Techniques for Building SNARKS Over the Integers* [CHA26].

Table of Contents

Fully Succinct Arguments over the Integers from First Principles . . . 1
Matteo Campanelli and Mathias Hall-Andersen
1 Introduction . . . 3
1.1 Technical Overview . . . 5
1.2 Related Work . . . 8
1.3 More on Motivation for Native Arguments over $\mathbb{Z}$ . . . 9
2 Preliminaries . . . 10
3 Relations over $\mathbb{Z}$ and their Fingerprint . . . 12
3.1 Fingerprinting Relations . . . 12
3.2 R1CS over $\mathbb{Z}$ and its Fingerprint . . . 13
3.3 Multilinear Polynomial Evaluation and its Fingerprint . . . 14
4 Idealized Protocols for Arguments over $\mathbb{Z}$ . . . 14
4.1 Algebraic Holographic Proofs with Modular Remainder Queries . . . 14
4.2 Weak Knowledge Sound mod-AHPs . . . 16
5 Integer Polynomial Commitments with Evaluation Opening over $\mathbb{Z}_q$ . . . 16
5.1 Model . . . 17
5.2 mod-PCs in Action: compiling mod-AHPs into SNARKs over $\mathbb{Z}$ . . . 18
6 From Multilinear Sumchecks to mod-AHPs . . . 19
6.1 Background on Spartan . . . 19
6.2 Spartan as a mod-AHP . . . 20
7 How to Build mod-PCs: Core Schemes and Compilers . . . 20
7.1 A Core Construction from a New Perspective on DARK . . . 20
7.2 A Compiler for Norm-Succinct Polynomial Commitments . . . 22
7.3 A Compiler Obtaining Prover-Efficiency for Sparse Polynomials . . . 23
8 Putting it All Together: Zaratan . . . 24
A Further Preliminaries . . . 29
A.1 The Sumcheck Protocol . . . 29
B Generic Group of Unknown Order (GGUO) Model . . . 29
C Wesołowski's Proof-of-Exponentiation . . . 29
D Our Core mod-PC Construction . . . 30
D.1 Evaluation Protocol in [BHR+21] . . . 32
D.2 Our mod-PC construction . . . 32
E Reverse Kronecker Substitution . . . 33
F Reducing the Requirements for Indexing Commitments in our Compiler . . . 36
G Missing Proofs . . . 36
G.1 Proof of Theorem 2 . . . 36
G.2 Proof of Theorem 3 . . . 36
G.3 Proof of Theorem 4 . . . 40
G.4 Proof of Theorem 9 . . . 41
G.5 Proof of Theorem 10 . . . 41
H Building mod-PCs for Sparse Polynomials and Delayed-Input Soundness . . . 41
H.1 Delayed-Input (Deterministic) Soundness . . . 42
H.2 A construction from SPARK [Set20] . . . 43
I Additional Definitions for mod-PCs . . . 43
I.1 Weak Evaluation binding . . . 43
I.2 (Strong) Knowledge Soundness over Adversarial Primes . . . 44
J Oracle Polynomials over $\mathbb{F}$ in mod-AHP . . . 44
J.1 Augmented model . . . 44
J.2 Prime-Agnostic Polynomial Commitments . . . 45
J.3 Extending the compiler . . . 46
K More on Motivation of Native SNARKs over $\mathbb{Z}$ . . . 47
K.1 Non-Zero Gates for Integers . . . 48
L On the Efficiency of Zinc [GWV+25] . . . 49

1 Introduction

Succinct arguments of knowledge allow an untrusted prover P to convince a verifier V that they possess a witness w for some statement x in a NP relation $(\mathsf{w}, \mathsf{x}) \in \mathcal{R}$, with verifier complexity poly-logarithmic in the (bit)length of the witness. To further allow the verifier to be poly-logarithmic in the size of the description of the relation (e.g. an arithmetic circuit), many such argument systems are so-called *preprocessing* succinct arguments of knowledge, where the description of the relation $\mathcal{R}$ is compressed cryptographically into a small commitment by an "indexer" algorithm. Throughout this work, unless stated otherwise, we will be referring to preprocessing succinct arguments of knowledge, simply as succinct arguments.

Succinct Arguments over Finite Fields. Most efficient succinct arguments in the literature prove satisfiability of relations over *finite fields*, e.g. satisfiability of arithmetic circuits over $\mathbb{F}_{p^e}$ for a prime p. The focus on finite fields is largely a side-effect of their rich algebraic structure which enables very efficient succinct arguments to be constructed, e.g. every non-zero element has an inverse, fields with smooth (additive/multiplicative) groups can easily be constructed and elliptic curves enable efficient homomorphic commitments, pairings and Fourier transforms [BSCKL21] for any (sufficiently large) finite field. In contrast, the focus of this work is to explore succinct arguments which directly prove computations over the integers, an infinite integral domain whose only invertible elements are ± 1, which often complicates extraction.

Practical Motivations for Computation over $\mathbb{Z}$. By computation over the integers we refer to arithmetic circuits with multiplication and addition gates (over $\mathbb{Z}$) where wires carry integers of arbitrary size[3]. In this context, proving a computation means proving knowledge of a wire assignment satisfying the circuit. Besides the obvious applications in proving computations over the integers, such circuits also allow very efficient emulation of integer rings $\mathbb{Z}/(n\mathbb{Z})$, including: k-bit arithmetic $\mathbb{Z}_{2^k}$, prime fields (of any size) $\mathbb{F}_p$ and RSA groups $\mathbb{Z}_{\hat{N}}$: by operating over the integers and allowing the prover to witness the reduction[4]. Because integer circuits also allow very simple range checks using Lagrange's four-square theorem, efficiently "switching" between different integer rings, e.g. embedding a $\mathbb{Z}_{2^k}$ element into $\mathbb{Z}_p$ is also possible. Applications include proving operations on RSA operations (e.g. signatures and accumulators [CL02,BCF+21,CFH+22]), ideal class group operations, homomorphic encryption [Bra12, BGV12, FV12], equivalence of discrete logarithm in different groups [BCF+21, OKMZ24]. Being able to prove computations over $\mathbb{Z}$ would even enable mixing-and-matching these settings within the *same* integer circuit, a feature that could be for example useful in the case of aggregation of SNARKs relying on different arithmetic. Finally, circuits over $\mathbb{Z}$ can be used to compute over $\mathbb{Q}$ and algebraic number fields, by representing rational numbers as pairs of integers. We refer the reader to Section 1.3 for further details and applications.

Comparison with Emulation. Since existing succinct arguments already exist for NP-complete relations, it is possible to use existing succinct arguments for machine emulation [GPR21] over finite fields $\mathbb{F}_p$ to prove satisifiability of a circuit over the integers, by emulating the integer arithmetic over $\mathbb{F}_p$: processing the circuit gate-by-gate using subroutines for addition and multiplication on arbitrary elements of $\mathbb{Z}$. Such an approach incurs a substantial overhead both asymptotically and concretely, similarly to how emulating arbitrary precision arithmetic using a fixed word size machine incurs overhead outside the domain of succinct arguments. In practice, such techniques are used to emulate "foreign fields"[5] and operations in RSA groups, however the resulting circuits/machine execution trace is significantly larger than the original computation, e.g. the overhead for RSA signature verification is 5000× [KPS18] (see also discussion in Section 1.3).

Challenges for Arguments over $\mathbb{Z}$. The design of succinct arguments is far better understood over finite fields than over rings. Among the challenges in designing arguments over $\mathbb{Z}$ is that the invertible elements are just ± 1, this issue sometimes shows up in extraction where the extractor must invert a

[3] But of course, ultimately, bounded by a polynomial in the security parameter, for any malicious prover (and therefore the honest one) will have to run in $O(\mathsf{poly}(\lambda))$.

[4] Note that we do not need to ensure that the prover fully reduces the integer: at worst he picks a larger representative of the equivalence class.

[5] A field different from the "native field" of the argument over which circuit satisfiability is proved.

matrix of challenges for different transcripts. Smaller challenges include the lack of Lagrange interpolation and the lack of smooth subgroups for the integers. This makes arithmetizations relying on such structure impossible [GWC19] [BBHR18] [COS20] and makes GKR-inspired [GKR08] [CBBZ23] techniques a natural choice. It is known that the multilinear sum-check can be adapted to work in a large class of rings [Sor22]; however, our approach will bypass the such adaptations entirely by only requiring the *indexer* (the preprocessing) to be executed over the integers. Since the latter depends only on multilinear interpolation, we can do without adapt other components to the ring setting and instead reuse many techniques from the world of finite fields through some of the observations in this work.

Resolving an Open Question: Succinctness over $\mathbb{Z}$. Our goal in this work is to propose succinct arguments for integer computations, meaning that the verifier complexity should be poly-logarithmic in the size of the witness. For finite alphabets, like the finite fields, this is equivalent to being poly-logarithmic in the dimension N of the witness vector $\vec{w} \in \mathbb{F}^N$. However, for computations over the integers, the witness vector $\vec{w} \in \mathbb{Z}^N$ may have elements of *arbitrary size*, hence there is a distinction between being succinct in the dimension N and the bitlength of the witness $N \cdot \log_2 \|\vec{w}\|_\infty$. Using Bulletproofs-inspired [BBB+18] techniques, existing work by Towa and Vergnaud [TV20] constructs a succinct argument over $\mathbb{Z}$ with proof size that is logarithmic in N but linear in $\log_2 \|\vec{w}\|_\infty$, while having a verifier running linear in N. The work in [TV20] leaves succinctness in the bitlength of the witness as an open question. In this work, we will propose a succinct argument with poly-logarithmic verification time (and hence communication) in $N \cdot \log_2 \|\vec{w}\|_\infty$—the bitlength of the wire assignments in the circuit[6]. We believe that the techniques in this paper, namely the reverse Kronecker substitution (Section 7.2and Section E), can also be applied to this work of Towa and Vergnaud to achieve communication logarithmic in $N \cdot \log_2 \|\vec{w}\|_\infty$, but with linear verification time in N.

Our Contributions. In this paper we study SNARKs whose underlying computational model is defined over the ring of integers and which are fully-succinct ($\mathbb{Z}$NARKs). We identify a general paradigm through which to construct efficient $\mathbb{Z}$NARKs, introduce key abstractions, and show how they can be used to obtain concretely efficient SNARKs over the integers. Our core contributions are:

- **General Techniques.** We identify a simple information-theoretic technique—a form of arithmetic hashing—that, when applied over relations, lets us bootstrap arguments for $\mathbb{Z}$ from existing SNARKs over large prime fields. This central observation forms the basis for our other contributions.
- **A Methodology: Framework for Building Arguments over** $\mathbb{Z}$. We generalize to the integers a framework that mirrors an approach by-now established in literature: the one based on *i) idealized protocols* (AHP/PIOP, or PHP[7]) and *ii) polynomial commitments* [KZG10] to build SNARKs. We introduce respective analogues for these primitives on which we elaborate in the technical overview. We provide a compiler that constructs a knowledge-sound argument over the integers from our new building blocks (see also Fig. 1).
- **A Construction: the First Fully Succinct** $\mathbb{Z}$**NARK.** By instantiating our framework, we obtain Zaratan[8], the first native argument for R1CS satisfiability over $\mathbb{Z}$ with both proof size and verification time sublinear in the *total* witness size: both the dimension of the witness and the bit-length of its largest integer. Zaratan can be instantiated to obtain a *transparent setup* using ideal class groups. Its efficiency is summarized in Table 1.
- **A Case Study: Integer Spartan.** We showcase our framework through a prominent protocol in literature, Spartan [Set20]. Through our approach, we bootstrap the information-theoretic backbone of Zaratan with Spartan and prove its security *with minimal effort*. Our approach is plug-and-play: future works can do the same with different proof systems.

[6] We will occasionally refer to this property, sublinearity in $\log_2 \|\vec{w}\|_\infty$, as *norm-succinctness*, to distinguish it from "succinctness" in $|\vec{w}|$. While we choose this phrase because we believe it can intuitively carry the right meaning, we also stress that, if interpreted literally, is slightly imprecise: we are actually requiring sublinearity in the logarithm of the norm (not just the norm itself); see, e.g., Table 1.

[7] Algebraic Holographic Proofs [CHM+20], Polynomial Interactive Oracle Proofs [BFS20] and Polynomial Holographic IOPs [CFF+21].

[8] Zaratan (za·ra·tán) is a mythological sea turtle, allegedly of incredible size, appearing in the work of Jorge Luis Borges [BGTT57] and in Arabic folklore.

Scheme	Proof Size	Prover	Verifier
Our mod-PC (Section 7.1+7.2)	$O_\lambda\left(\max\left\{\nu \log N, \nu^2\right\}\right)$	$O_\lambda(mN \log N)$	$O_\lambda\left(\max\left\{\nu \log N, \nu^2\right\}\right)$
$\mathbb{Z}$aratan (Section 8)	$O_\lambda\big(\max\big\{\nu \log^2 N, \nu^2 \log N\big\}\big)$	$O_\lambda(mN \log N)$	$O_\lambda\big(\max\big\{\nu \log^2 N, \nu^2 \log N\big\}\big)$
Zinc [GWV+25]	$\omega\left(\sqrt{2^\nu \cdot N}\right)$	N/A	$\omega\left(\sqrt{2^\nu \cdot N}\right)$

Table 1. Efficiency summary for our constructions and comparison with the subsequent work Zinc (CRYPTO 2025); see also Section 1.2 and Section L.
Notation: For mod-PC, the quantities refer to the case of multilinear polynomials. The witness $\vec{w}$ is assumed to be a vector of integers of N elements (for mod-PC, it is the number of coefficients of a multilinear polynomial); $m := \log_2 \|\vec{w}\|_\infty$ is the size in bits of the norm of the witness vector (resp. coefficients vector, for mod-PC); $\nu := \log m$. Notice that mN is essentially the size of the witness/polynomial (and the prover is quasi-linear in it). For simplicity, above, we use λ as both a computational and statistical parameter. $O_\lambda(f)$ is a shortcut for $O(p(\lambda) \cdot f)$ where $p(\cdot)$ is a fixed (and small) polynomial. In addition to the above the verifier's time requires the time to read the public input.

– **More Techniques for Polynomial Commitments in Our Setting.** In Section 7 we propose:
  - *A new Perspective on the DARK approach.* We show how to augment the techniques from DARK— [BFS20] and [BHR+21]—previously used for commitments *over finite fields*, to construct *integer* polynomial commitments of the kind we introduced (where an integer polynomial can be opened modulo an arbitrary prime after sending the commitment).
  - *Compilers to the Fully-Succinct and Sparse Polynomial Settings.* We propose a general approach to transform any polynomial commitments for integer polynomials and opening modulo a prime into one with that is fully-succinct. We also revisit known techniques to obtain sparse polynomial commitments and show how to apply them in our setting.

1.1 Technical Overview

Our Approach in a Nutshell. Our approach can be summarized as follows: we use an integer commitment to commit to the wire assignment of an arithmetic circuit, then view the technique of *reducing an arithmetic circuit modulo a random prime*[9] as mapping a relation (circuit) over the integers to a randomized relation over a prime field. Finally, we use an existing SNARK blueprint to prove satisfiability of the reduced relation. We will now expand a bit more upon the many subtle technical details required to take this idea to a provable construction.

Committing to the Witness. In order to commit to the wire assignment (witness) and later be able to prove satisfiability of the *same* assignment modulo a prime, we need a polynomial commitment scheme which enables us to efficiently commit to multilinear integer polynomials, and, subsequently prove evaluations of the polynomial at *arbitrary points* modulo an *arbitrary prime*, i.e. commit to $f(X_1, \ldots, X_n) \in \mathbb{Z}[X_1, \ldots, X_n]$ and open $\bar{f}(x_i, \ldots, x_i) \bmod q$ for $x_i, \ldots, x_i \in \mathbb{F}_q$ and any sufficiently large prime $q \in \mathbb{P}$. We call such a scheme a *polynomial commitment with modular remainder openings* and provide the first such construction based on groups of unknown order using DARK-like techniques.

Fingerprinting. In this paper, we consider the satisfiability of R1CS relations over the integers, defined by three matrices $A, B, C \in \mathbb{Z}^{N \times r}$. A witness $\mathsf{w} \in \mathbb{Z}^N$ satisfies the relation, iff. $(A \cdot \mathsf{w}) \circ (B \cdot \mathsf{w}) = (C \cdot \mathsf{w})$ and $\mathsf{w}_1 = 1$, where $\circ$ is the component-wise (Hadamard) product. After having the prover commit to the wire assignment w, we will sample a prime $q \leftarrow\!\!\$\ \mathbb{P}_\lambda$ from a set $\mathbb{P}_\lambda$ of primes. Then we ask the prover to demonstrate (using a succinct proof) that $\mathsf{w} \bmod q$ satisfies $(A_q \cdot \mathsf{w} \bmod q) \circ (B_q \cdot \mathsf{w} \bmod q) = (C_q \cdot \mathsf{w} \bmod q)$ where $A_q, B_q, C_q \in \mathbb{F}_q^{N \times r}$ are the matrices obtained by reducing the entries of A, B, C modulo q. This is a (low depth) example of arithmetic fingerprinting, where satisfiability of an arithmetic circuit C over the integers for input $\mathsf{w} \in \mathbb{Z}^N$ can be verified by computing the circuit modulo a random number

[9] Similar techniques have been used in complexity theory and randomized algorithms (in settings unrelated to ours); see e.g., [AB06, §7.2.3].

[AB06, §7.2.3]. Our observation is that by: 1) limiting the depth of the circuit to a constant, thereby allowing the prime to be roughly λ bits large, and 2) using a commit-and-prove proof system to avoid sending w to the verifier and computing the circuit modulo q. We can "scale down" this classical technique and make the *verifier poly-logarithmic* in the size of the circuit and the witness.

Indexing Over The Integers. Because we need a *preprocessing* SNARK to prove satisfiability of the *reduced relation* in the last step, the protocol sketch above runs into a problem: we want to prove satisfiability of the R1CS relation A_q, B_q, C_q for the challenge q, but the verifier requires *the index of this reduced relation* to verify the proof. Naively, this would require having the verifier reduce each matrix modulo q, then running the indexer for the R1CS relation (A_q, B_q, C_q): meaning that the *verifier's running time would be linear* in the size of the relation. We circumvent this issue by looking at SNARKs where *the indexer is run over the integers*. Notably *multilinear interpolation over the Boolean hypercube* can be done over the integers, by combining this with our polynomial commitment with modular remainder openings, we can compute a *universal index* which works for any q. This means the verifier can check satisfiability of the R1CS relation modulo (A_q, B_q, C_q) for *any sufficiently large prime* q using *the same index*. We explore this, in detail, for Spartan [Set20] and the Spark compiler.

Proving Knowledge Soundness. A natural approach to constructing an extractor for the scheme outlined is: 1) Run the prover until the end. 2) Extract a witness modulo q from the SNARK. 3) Repeatedly rewind to the point where we sample q and extract the witness modulo many different primes $q_1, q_2, \ldots, q_k$. 4) Finally, *recover the integer witness using the Chinese Remainder Theorem*. However, if we attempt this with the scheme as outlined above we encounter an issue: the malicious prover is a PPT Turing machine, however, *we cannot bound the size of the witness* recovered using CRT by the prover's running time. Consider the following polynomial commitment scheme:

1. The (non-hiding) commitment com to $f(\vec{X}) \in \mathbb{Z}[\vec{X}]$ is a Turing machine.
   Which, when fed $\vec{x} \in \mathbb{F}_q^\mu$ and q, produces the result $f(\vec{x}) \bmod q$.
2. The opening of com at $\vec{x} \in \mathbb{F}_q^\mu$ and q to $y \in \mathbb{F}_q$, is an efficiently checkable mathematical proof that:
   (a) com is a Turing machine evaluating an integer polynomial of degree 1.
   (b) When evaluating $f(\vec{x})$, the result is $y = f(\vec{x}) \bmod q$.

Although this polynomial commitment scheme is (perfectly) binding, we will be *unable to extract a witness*. Consider $\mathsf{w} \in \mathbb{Z}^N$ st. $\forall q \in \mathbb{P}_\lambda.(A_q\mathsf{w}) \circ (B_q\mathsf{w}) = (C_q\mathsf{w}) \bmod q$ yet $A\mathsf{w} \circ B\mathsf{w} \neq C\mathsf{w}$. Observe that, e.g. the relation $\mathsf{w} \cdot 2 = 1$ is not satisfiable over the integers, but is satisfiable over $\mathbb{Z}/(\prod_{q \in \mathbb{P}_\lambda} q)\mathbb{Z}$. The problem is that although w has exponential size, a cheating prover *can efficiently construct a Turing machine (commitment) which computes the reduced evaluation* of such a polynomial modulo any prime $q \in \mathbb{P}_\lambda$. Hence he can successfully prove that w satisfies the relation modulo any prime $q \in \mathbb{P}_\lambda$. Intuitively, the problem is that we cannot "force" the prover to write down w in e.g. base 2, which would allow us to bound the size of the witness by the running time of the prover. The construction above is contrived, however, in general we need to rule out the possibility of committing to (some) exponentially large integers in polynomial time. In order to achieve this, we use an idealized model, the *generic group of unknown order* model, to "count" the number of bits in w in terms of oracle queries, e.g. it is not possible to compute $[v] \cdot x$ in time less than $\log(v)$ within this model. Note that this issue is unique to the integer setting[10]: for relations over finite fields, the size of the witness is bounded by the size of the field.

Our General Framework: mod-AHP + mod-PC ⇒ SNARKs for $\mathbb{Z}$. We turn the approach outlined above into a general framework, which we explore now. Our starting point is the modular recipe used in the construction of recent SNARKs where the core construction is described as an *idealized protocol* with algebraic properties, or Algebraic Holographic Proof (AHP) [CHM+20]. This type of constructions assume a finite field $\mathbb{F}$ and their flow looks roughly as follows: a prover (P), on input a statement and a witness, sends some *oracle polynomials* in each round to the verifier (V), who responds with a random challenge; afterwards, during a *query stage*, V can query an oracle polynomial f with an evaluation point $\vec{x}$ to obtain $v = f(\vec{x})$. V can iterate this process for several different polynomials and evaluation points (all arithmetic being performed in $\mathbb{F}$). Finally, V outputs a

[10] When we don't enforce a fixed a priori bound on the norm of the witness.

decision bit indicating "accept" or "reject", based on the result of the evaluation queries. An AHP can be turned into an argument system by replacing the oracles and the query phase with a *polynomial commitment scheme* (PC) [KZG10]: the prover can commit to the oracle polynomials and later, upon receiving an evaluation point $\vec{x}$, can send an evaluation proof to convince the verifier that evaluation $v = f(\vec{x})$ is done correctly (again, both the polynomial and evaluation are over $\mathbb{F}$). For them to be combined effectively, both the AHP and the PC need to satisfy some extractability-flavored properties.

Recall that our key idea is to leverage fingerprinting, i.e., in some stage of the protocol, to sample a prime q and then continue the evaluation of the protocol "over $\mathbb{F}_q$". The counterpart for AHP we introduce is called a AHP over $\mathbb{Z}$ with modular remainder queries (or mod-AHP, Definition 10) and it is thus called because it works this way: P and V interact with P sending oracle polynomials over $\mathbb{Z}$; at the end of this stage a prime q is sampled; the verifier can now request an evaluation point $\vec{x}$ for a polynomial f but will be constrained to obtain only $f(\vec{x})$ mod q. That is, the stage where oracles are sent is "more expressive", while the query stage is still constrained to work over a finite field. We say that a mod-AHP is *(full) knowledge-sound*[11] if, intuitively, we are able to extract a witness *over the integers* from a prover who is able to successfully convince the verifier.

At this point the reader can probably already imagine a polynomial commitment notion that would be a good match for mod-AHPs: it should be able to commit to polynomials over the integers, *yet* it will not need to support full-fledged integer evaluations. This type of PC, which we dub mod-PC (Definition 14), in fact just needs to support evaluation queries modulo q, for a prime q *unknown* at commitment time. The type of extractability property for mod-PC that we need, however, requires us to be able to extract a *polynomial over* $\mathbb{Z}$ (not just over $\mathbb{F}_q$) from an adversary providing valid proofs. With these two notions under our belt we are able to provide an abstract compiler from a knowledge-sound mod-AHP and a secure mod-PC to arguments for non-deterministic relations over $\mathbb{Z}$ (see Fig. 1). We stress that, while we used univariate polynomials for our examples above, all our primitives are defined over multivariate polynomials.

When applied together, mod-AHP and mod-PC, yield a final argument that is *fully succinct* as long as the mod-PC is succinct in m (see Remark 4). Keeping this requirement in mind, we now discuss how to instantiate our building blocks.

From DARK to a mod-PC Construction. We use techniques from the DARK compiler [BFS20] to construct your mod-PCs. The original construction of Bünz et al. [BFS20] is not directly applicable to our setting, because the binding notion is too weak: it only allows extracting a rational function with small integer denominator. Instead we rely on a protocol by Block et al. [BHR$^+$21] which allows extraction of an integer polynomial. We combine this protocol with a generic approach allowing us to reduce the norm of the coefficients of the integer polynomial at the cost of introducing an additional logarithmic number of variables (see Section E). The result is a mod-PC with linear commitment/opening time for the prover in the size of the polynomial and polylogarithmic verification time. The commitment is a single group element from a group of unknown order. The resulting construction requires solving the problem of how to obtain norm-succinctness of such schemes, which required solving additional technical challenges; our resulting techniques can be generally applied to other constructions in order to obtain norm-succinctness and are of independent interest. (In order to keep this general technical overview brief, we simply refer the reader to the more detailed one in Section 7.1).

Towards a "plug-and-play" Framework In order to populate our framework through existing constructions, we look for the weakest possible requirement on them. This is not just for theoretical interest; it will in fact make it significantly easier for us to prove these properties hold for prior constructions. Our compiler requires what is arguably the most natural notion of soundness in this setting, i.e., what we earlier called "full" knowledge soundness (Full KSND)—where we should be able to extract an *integer* witness; however, it turns out we can actually do with a weaker property: a form of KSND where we require to extract only the "fingerprint" (modulo q) of a potential witness. With a very idealized example: consider $\mathsf{w}^\star = 42^{150}$, the integer witness satisfying the equation $\mathsf{w}^{100} - 42^{15000} = 0$; after sampling a prime, say[12] 13, we only need to extract the witness fingerprint $\mathsf{w}_q^\star \in \{0, \ldots, 12\}$ s.t. the "fingerprinting variant" of the original relation is satisfied, i.e. $\mathsf{w}_q^\star \equiv 42^{150} \pmod{13}$.

[11] One intuition on why "full": it allows us to extract the "full" integer witness. Later, we will be able to weaken this property and show that a mod-AHP not "fully" extractable can still yield an argument over $\mathbb{Z}$ (with a few extra requirements).

[12] This is just an example—we sample primes of λ bits in our constructions.

We precisely formalize these properties and show that, for the case of R1CS over the integers, this type of weaker form of extractability can be lifted to its full counterpart. In fact, our results are not limited to R1CS: we provide a general set of definition and properties expressing when this type of lifting is possible.

Spartan as a mod-AHP We are able to show that the argument for R1CS Spartan [Set20], at its core, is a mod-AHP with the aforementioned weak extraction property. The original Spartan works over a finite field and, at a high level, works by: having the prover send an oracle polynomial to a multilinear extension of the witness vector $\tilde{w}$ (a polynomial encoding of the vector) and then run two sumcheck protocols [GKR08] for appropriately crafted equations; after the last round, the verifier queries $\tilde{w}$ on a random point and performs some consistency checks. We are able to observe that the first oracle message $\tilde{w}$ does not require a finite field to be defined. We modify Spartan to sample the (prime) order of the field *throughout the interaction*. We can then argue that the core proof of knowledge soundness of [Set20] can be leveraged for the case of $\tilde{w}$ as a polynomial over the integers. Our general approach, which we describe in Section 6.1, should be applicable with minimal modifications to other multivariate-flavored schemes, such as HyperPlonk [CBBZ23].

mod-PC for Sparse Polynomials In the presentation above we deliberately omitted that the verifier, at the end, needs to also query polynomials $\tilde{A}, \tilde{B}, \tilde{C}$ encoding the R1CS matrices. What we sketched so far does give us a version of Spartan over $\mathbb{Z}$, but gives us an efficient verifier only for the case where the computation is "highly regular" and thus $\tilde{A}, \tilde{B}, \tilde{C}$ can be evaluated very efficiently (this is the case for example of data-parallel circuits [CGG+23]). Our goal for our final protocol, Zaratan, is to support an efficient verifier for arbitrary computations. In order to do this we need to solve an additional challenge: obtaining an efficient mod-PC for *sparse* polynomials[13]. Our approach works by showing that the SPARK compiler in [Set20] (which lifts a polynomial commitment for *dense* polynomials into one for *sparse* ones) can be recast as a mod-AHP for deterministic computations with specific properties. In this part of our work we are able to reuse some of the abstractions we used to define "weak" knowledge soundness. Although we do not cast them explicitly under this light, our techniques have at their heart a recipe to construct general *succinct functional commitments* with "fingerprinting properties" (a natural generalization of our mod-PC notion) and hence we believe them to be of independent interest (see Section 7.3).

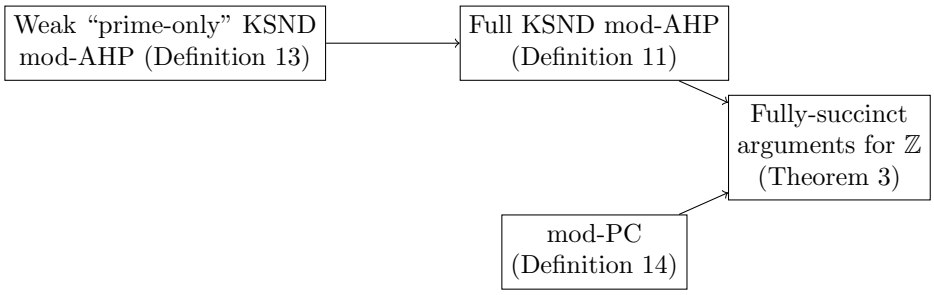


Fig. 1. Relationships among some of our core abstractions.

1.2 Related Work

Poly. Comm from Groups of Unknown Order. Several works, starting with Bünz et al. [BFS20] [BFS19] and Block [BHR+21] have constructed polynomial commitments over prime fields from groups of unknown order, by lifting the evaluation to the integers or rational functions of bounded norm. Related techniques have been leveraged in the works Dew [AGL+23] and Behemoth [SB23]. All these works obtain polynomial commitments over finite fields, not the integers.

[13] The reason we need this is that it will be used to commit to (and prove evaluations of) to $\tilde{A}, \tilde{B}, \tilde{C}$. These are of quadratic size but they have a sparse representation in that only have a linear number of non-zero elements.

Arguments for Rings. The work in [GNS23] constructs a family of SNARKs for computations over finite rings, called Rinocchio. Rinocchio relies on new "linear-only" assumptions in rings which have not received scrutiny; it also requires a trusted setup *specific to the circuit being proved*; to the best of our knowledge Rinocchio is not succinct in the size of the elements of the underlying ring. We have already mentioned the work of Towa and Vergnaud [TV20] who constructed a Bulletproofs-inspired argument [BBB+18] for Diophantine equations over the integers. Their construction has a non-succinct verifier but, differently from our scheme, achieves zero-knowledge. We leave as future work how to add zero-knowledge into our approach. In [Sor22] Soria-Vazquez constructs interactive proofs for *deterministic computations* expressed over infinite and non-commutative rings. Our work differs in several respects. Our focus is on *non-deterministic computations* over the ring of integers. While the rings considered in [Sor22] are more general, the constructions proposed in it do not apply to the non-deterministic case.

Other Works. The sampling of random primes in proofs of exponentiation in [Wes19] can be seen as a form of fingerprinting for a very specific relation.

Subsequent Work. The work in [GWV+25] leverages in part our methodology to offer a construction, Zinc, with different tradeoffs. From the technical standpoint, they also use fingerprinting modulo a random prime, but their construction eventually works over the rationals. They design an appropriate polynomial commitment scheme for the rational setting by building on the techniques from Brakedown [GLS+23]. Being hash-based, their scheme has smaller constants than the one arising in our current polynomial commitment instantiation (based on groups of unknown order). On the other hand, their final construction has exponentially worse asymptotic behavior for proof size and verification time compared to Zaratan[14]. The fact that [GWV+25] leverages hash-based building blocks increases our confidence that future work could find more concretely efficient instantiations for mod-PCs compared to ours.

1.3 More on Motivation for Native Arguments over $\mathbb{Z}$

Having[15] native arguments over $\mathbb{Z}$ is important from an engineering perspective, simplifying circuits, improving developer experience and reducing the probability of bugs. From a practical perspective we believe there are several scenarios where native arguments over $\mathbb{Z}$ may offer interesting tradeoffs (see also Section K where we discuss more settings, including range checks with $O(1)$ constraints and computations on rational numbers). We remark that some of these settings have not been pointed out in prior literature.

Checking RSA signatures The case of RSA signatures is mentioned here, not only because of its practical value, but also because it is instructive to see some of the concrete costs of emulation over fields. A valid RSA signature [RSA78] $\sigma \in \mathbb{Z}$ should verify the following equation:

$$\sigma^e \equiv \mathsf{m} \pmod{\hat{N}}$$

where $e, \mathsf{m}, \hat{N} \in \mathbb{Z}$ are respectively a fixed constant (typically $e \in \{3, 2^{16}+1\}$), the message (in practice, the *hash* of the message) and the RSA modulus (the public key, whose typical size is of the order of thousands of bits). Expressed as an integer relation this can be expressed as showing knowledge of $\sigma, q \in \mathbb{Z}$ such that

$$\sigma^e - q\hat{N} - \mathsf{m} = 0$$

It is easy to convince oneself that, for the case $e = 2^{16}+1$, this can be expressed in 16 R1CS constraints over the integers (we assume that $\hat{N}$ and m are public inputs, but a different choice would be of little consequence). When expressing the same computation over a standard field choice (254-bit prime) and through state-of-the-art tools (that is, highly optimized to obtain small circuits on this type of computations) the resulting emulation-based R1CS has $\approx 90K$ constraints, a

[14] See Table 1; proving time is not reported in [GWV+25] and thus the table has no entry on it. We also remark that it is plausible that the hash-based techniques in [GWV+25] offer concrete speedups compared to our current instantiation (based on unknown-order groups) at least in settings with relatively small integer sizes (few hundreds of bits).

[15] We expand on the points in this section in Section K.

5000× blowup (see Table III in [KPS18]). While it would not be a fully apples-to-apples comparison to just compare the two numbers, *this does provide reasonable evidence of lost opportunities for optimizations*: for example, in the information-theoretic backbone of our protocol, ℤaratan, the prover effectively runs linearly in N, the number of constraints in the R1CS, i.e., 16 (!)[16]

Arbitrary-precision arithmetic Our work directly enables rational number arithmetic (see Section K). With this tool under our belt, we can perform arbitrary-precision arithmetic. This can be of interest when proving computations in the realms of engineering and natural science which may require high levels of precision (which makes *full* succinctness ideal in this setting).

Machine learning SNARKs over $\mathbb{Z}$ also offer a natural avenue to machine learning settings: quantization to integers is, in fact, used in commercially available chips specialized for machine learning [SY]. Further evidence for this being a viable approach comes from recent works showing how integer computations can be sufficient to obtain reasonable precision in ML [LS24].

Heterogeneous Proof Aggregation In the setting of proof aggregation, given cryptographic proofs $\pi_1, \pi_2, \ldots, \pi_\ell$ for respective statements $x_1, x_2, \ldots, x_\ell$, one produces a single *aggregate proof* π^* for the same statements. The final π^* is expected to have size sublinear in ℓ. This approach is useful to reduce communication complexity whenever we expect the initial proofs π_i-s to be computed independently by different parties. Aggregation has been deployed in real-world settings [Lab,Lay] and has recently received attention from the research community [GMN22,ABST23,GPPS24,YZRM24]. It is natural to think of settings where the proofs to aggregate come from different schemes (akin to how TLS supports, e.g., both RSA and ECDSA signatures). The schemes underlying these proofs may be *heterogeneous*, that is they could rely on very different algebraic structures (e.g., distinct elliptic curves with very different properties). In this case the aggregate proving algorithm would need to handle different arithmetic for the same computations, which makes ℤNARKs a natural candidate (a single integer circuit could internally express different algebraic structures at the same time). In this scenario, the aggregation prover—which certifies the computation of the verifiers for the constituent proofs—would need to emulate arithmetic in a range of different integer rings (e.g., different prime fields). A ℤNARK is a natural candidate for this aggregation step since integer circuits can emulate any integer ring with essentially no overhead.

Future Work There are a few interesting questions stemming from our work. Obvious future directions include exploring zero-knowledge and instantiations of mod-AHPs besides Spartan. The latter provides a blazing fast information-theoretic backbone for SNARK over $\mathbb{Z}$. Our mod-PC, however, has relatively large constants due to using unknown-order groups and the extra sub-protocols from Block et al. [BHR+21]. Could one prove its soundness without relying on [BHR+21]? What are alternative constructions of mod-PCs based on light-weight tools (e.g., hashing) with better asymptotics than those in [GWV+25]?

2 Preliminaries

Notation We write $f \in \mathbb{Z}_{\leq D}[X_1, \ldots, X_\mu]$ to denote that f is a polynomial over the integers in μ variables $X_1, \ldots, X_\mu$ such that the individual degree of each variable X_i is at most D, usually $D = 1$. For a positive integer n we write $[n]$ to denote the set $\{1, \ldots, n\}$. We define the following notations related to vectors. We denote by $\circ$ the Hadamard product. We write $\vec{0}$ to denote a vector with entries equal to the additive identity in a ring that will be made obvious from the context. Given a vector of integers $\vec{u} \in \mathbb{Z}^n$ and a prime q, we denote by $[\![\vec{v}]\!]_q$ the vector $\vec{v} \in \mathbb{F}_q$ such that for all $i \in [n]$ $v_i = u_i \bmod q$. For a matrix M and vector $\vec{v}$ we denote by $M \cdot \vec{v}$ the matrix-vector multiplication operation. We denote by M_i the i-th row of a matrix M. We denote by $\|\vec{v}\|_\infty = \max_i\{v_i\}$ the infinity norm of a vector of integers $\vec{v} \in \mathbb{Z}^n$. We sometimes abuse notation and, given a polynomial f we denote by $\|f\|_\infty$ the infinity norm of its vector of coefficients. Denote by $\mathcal{H}_n$ the n-dimensional Boolean hypercube over the integers $\mathcal{H}_n = \{0,1\}^n \subseteq \mathbb{Z}^n$.

[16] This includes all the "non-oracle" interactions of the prover. In the final protocol, these will be accounted for by requiring a polynomial commitment and opening (in the flavor of Section 5.1); there the prover's running time will also depend on the size of the integer coefficients. This suggests, in any event, that improving the efficiency of mod-PCs is a reasonable avenue to more efficient proofs for integer computations, as an alternative to field emulation.

Prime Sampling We denote by $\mathbb{P}_\lambda$ the set of λ-bit primes, i.e. $\mathbb{P}_\lambda \subseteq [2^{\lambda-1}, 2^\lambda)$. We use random prime sampling as done in previous works, e.g. [Wes19, BFS20, BCF+21]. By the prime number theorem it is easy to show that $|\mathbb{P}_\lambda| = \Theta\left(\frac{2^\lambda}{\lambda}\right)$.

Theorem 1 (Number of Primes [Ros41]). *Let $\pi(x)$ denote the prime-counting function, i.e. the number of primes less than or equal to x. Then:*

$$\frac{x}{\ln(x)+2} < \pi(x) < \frac{x}{\ln(x)-4}, \quad \text{for } x \geq 55.$$

Indexed Relations An indexed relation $\mathcal{R}$ is a set of triples $(\mathsf{i}, \mathsf{x}, \mathsf{w})$ where i is the index, x is the instance, and w is the witness. Intuitively the index describes the computation we are checking through the relation. For instance, for the case of circuits, the index will describe the circuit itself. We say that a relation is *deterministic* if it is a set of pairs index–instance rather than triples (or equivalently if w is always $\perp$). For any indexed relation we define a function $|\cdot|$ which associates to each index its size (a natural number). Given a size bound $\mathsf{n} \in \mathbb{N}$, we denote by $\mathcal{R}_\mathsf{n}$, the restriction of $\mathcal{R}$ to triples $(\mathsf{i}, \mathsf{x}, \mathsf{w})$ with $|\mathsf{i}| \leq \mathsf{n}$.

Multilinear Extensions We observe that the usual definition of multilinear extension (MLE) extends directly to the case of rings. Below we define MLE for integer-valued functions on the boolean hypercube. We refer the reader to [Tha23] for a broader discussion of multilinear extensions.

Definition 1. *Let $f : \mathcal{H}_\mu \to \mathbb{Z}$ be a function. The multilinear extension of f (which we denote by $\mathsf{MLE}(\mathsf{f})$ or $\tilde{f}$) is the unique polynomial $\tilde{f}(\vec{X}) \in \mathbb{Z}_1[X_1, \ldots, X_\mu]$ such that for all $\vec{x} \in \mathcal{H}_\mu . \tilde{f}(\vec{x}) = f(\vec{x})$. This polynomial can be constructed as*

$$\tilde{f}(\vec{X}) = \sum_{\vec{b} \in \mathcal{H}_\mu} \chi_{\vec{b}}\left(\vec{X}\right) \cdot f(\vec{b})$$

where $\chi_{\vec{b}}(\vec{s}) = \prod_{i=1}^{\mu} (s_i \cdot b_i + (1-s_i)\cdot(1-b_i)) \in \mathbb{Z}_1[X_1, \ldots, X_\mu]$ is the multilinear polynomial that equals 1 if and only if $\vec{b} = \vec{s}$, and 0 otherwise.

We often consider a vector $\vec{v}$ of size n as the function $f(i) := v_i$ with domain $\{0,1\}^{\log n}$ and we abuse notation writing $\mathsf{MLE}(\tilde{\mathsf{v}})$ to denote $\mathsf{MLE}(\mathsf{f})$. We will use this result whose proof is immediate from the construction of $\mathsf{MLE}(\mathsf{f})$. It essentially states that, for a function f, the MLE of f evaluated modulo q "matches" the evaluation modulo q of $\mathsf{MLE}(\mathsf{f})$.

Lemma 1. *Let $f : \mathcal{H}_\mu \to \mathbb{Z}$ be a function and let $q \in \mathbb{N}_{>0}$. Define $f_q : \mathcal{H}_\mu \to \mathbb{Z}$ as $f_q(\vec{\vec{x}}) := f(\vec{x}) \mod q$. Then for each $\vec{x} \in \mathbb{Z}^\mu$ we have that*

$$\mathsf{MLE}(\mathsf{f_q})(\tilde{\mathsf{x}}) \equiv \mathsf{MLE}(\mathsf{f})(\tilde{\mathsf{x}}) \mod \mathsf{q}$$

Dense and Sparse MLEs We recall some of the observations on sparse/dense representations of MLEs from [Set20]. A multilinear polynomial $f : \mathbb{Z}^\mu \to \mathbb{Z}$ can be represented uniquely by the list of evaluations of f over the boolean hypercube $\mathcal{H}_\mu$. We denote this representation as the *dense* representation of f, or $\mathsf{DenseRepr}(f)$. It is easy to show that if $f(\vec{x})$ is zero on any point $\vec{x} \in \mathcal{H}_\mu$, this does not need to be included in $\mathsf{DenseRepr}(f)$.

We say that a multilinear polynomial is *sparse* if the size of its dense representation is $o(2^\mu)$. Else we say it is *dense*. An example of sparse MLEs that is relevant for us is from R1CS matrices (Definition 5): the MLE of each matrix A, B, C has $2^{2\mu}$ coefficients where $N := 2^\mu$ is the size of the R1CS; however, its dense representation is of size $O(2^\mu)$.

Succinct Arguments with Universal SRS

Definition 2 (Preprocessing Argument with Universal SRS [CHM+20]). *A Preprocessing Argument with Universal SRS is a tuple $\mathsf{ARG} = (\mathcal{S}, \mathcal{I}, \mathcal{P}, \mathcal{V})$ of four algorithms. $\mathcal{S}$ is a probabilistic polynomial-time setup algorithm that given a bound $\mathsf{n} \in \mathbb{N}$ samples a (potentially structured) reference string srs supporting indices of size up to n. The indexer algorithm $\mathcal{I}$ is deterministic and, given as input the srs produces a proving index key and a verifier index key, used respectively by $\mathcal{P}$ and $\mathcal{V}$. The latter two are PPT interactive algorithms.*

Completeness *For all size bounds* $\lambda, \mathsf{n} \in \mathbb{N}$, $(\mathsf{i}, \mathsf{x}, \mathsf{w}) \in \mathcal{R}_{\mathsf{n}}$

$$\Pr\left[\langle \mathcal{P}(\mathsf{ipk}, \mathsf{x}, \mathsf{w}), \mathcal{V}(\mathsf{ivk}, \mathsf{x})\rangle = 1 \ : \ \begin{array}{c} \mathsf{srs} \leftarrow \mathcal{S}(1^\lambda, 1^{\mathsf{n}}) \\ (\mathsf{ipk}, \mathsf{ivk}) \leftarrow \mathcal{I}(\mathsf{srs}, \mathsf{i}) \end{array}\right] = 1$$

Knowledge Soundness For every $\lambda, \mathsf{n} \in \mathbb{N}$ and efficient adversary $\tilde{\mathcal{P}} = \left(\tilde{\mathcal{P}}_1, \tilde{\mathcal{P}}_2\right)$ there exists a (possibly non-uniform) efficient extractor Ext such that

$$\Pr\left[\begin{array}{c} (\mathsf{i}, \mathsf{x}, \mathsf{w}) \notin \mathcal{R}_{\mathsf{n}} \wedge \\ \left\langle \tilde{\mathcal{P}}_2(\mathsf{st}), \mathcal{V}(\mathsf{ivk}, \mathsf{x})\right\rangle = 1 \end{array} : \begin{array}{c} \mathsf{srs} \leftarrow \mathcal{S}(1^\lambda, 1^{\mathsf{n}}) \\ (\mathsf{i}, \mathsf{x}, \mathsf{st}) \leftarrow \tilde{\mathcal{P}}_1(\mathsf{srs}) \\ \mathsf{w} \leftarrow \mathsf{Ext}(\mathsf{srs}) \\ (\mathsf{ipk}, \mathsf{ivk}) \leftarrow \mathcal{I}(\mathsf{srs}, \mathsf{i}) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

Above the extractor takes in input the same random tape as the malicious prover.

Plain Interactive Protocols In some of our definitions (e.g., Definition 10) we will require a public-coin interactive sub-protocol. With the exception of constraining one of the parties to sending only random challenges, we will only need the syntactic properties of this interaction and the resulting view. We call a *plain interactive protocol* an interaction between two parties $\mathsf{P}_{\mathrm{ip}}, \mathsf{V}_{\mathrm{ip}}$ such that they both take in input a security parameter $\lambda \in \mathbb{N}$; the party P_{ip} also takes as input an arbitrary string aux. The security parameter is passed implicitly to both parties as a unary string. We denote by $\mathsf{tr} \leftarrow \mathsf{transcript}_\lambda\left(\langle \mathsf{P}_{\mathrm{ip}}(\mathsf{aux}), \mathsf{V}_{\mathrm{ip}}\rangle\right)$ the result of the interaction between the two parties. We can parse tr as a pair $(\vec{m}, \vec{\rho})$ such that $(m_1, \rho_1, \ldots, m_r, \rho_r)$ is the transcript of the interaction throughout the r rounds of the protocol. Each m_i (resp. ρ_i) is a message (resp. random challenge) sent by P_{ip} (resp. V_{ip}). Each message is assumed to be of size $\mathsf{poly}(\lambda)$.

3 Relations over $\mathbb{Z}$ and their Fingerprint

In this section we provide a set of definitions that will later allow us to capture when a standard interactive argument for prime finite fields can be lifted to an argument over the integers. For that, we need to build a vocabulary for what it means to map a relation to its "associated fingerprinting" over a prime q.

3.1 Fingerprinting Relations

Definition 3 (Associated fingerprinting relation). *Let $\mathcal{R}$ be an indexed relation over the integers. An associated fingerprinting relation for $\mathcal{R}$ is a mapping $[\![\cdot]\!]$ parametrized by a prime, such that for all primes q and positive integers n associates to $\mathcal{R}_n$ an efficiently computable relation $[\![\mathcal{R}_n]\!]_q$. This associated relation takes as input triples of the form $(\mathsf{i}, \vec{x}_q, \vec{w}_q)$ where i is in the same domain as the indices for $\mathcal{R}_n$ and $\vec{x}_q, \vec{w}_q \in \mathbb{F}_q$. We require that a fingerprinting relation is "admissible" if it preserves valid statement–witness pairs, that is: for all indices i, integer vectors $\vec{x}$ and $\vec{w}$*

$$(\mathsf{i}, \vec{x}, \vec{w}) \in \mathcal{R}_n \implies (\mathsf{i}, [\![\vec{x}]\!]_q, [\![\vec{w}]\!]_q) \in [\![\mathcal{R}_n]\!]_q$$

At times we will require that relations satisfy an additional property that will be key for some of our "lifting" results. This property intuitively states that a fingerprinting relation provides a reasonable probabilistic test for checking whether something is in the relation. This will be true for example for the case of R1CS structures and a natural associated fingerprinting relation for them.

Definition 4 (Good Membership Test). *An associated fingerprinting relation for $\mathcal{R}$ is said to provide a good membership test (for $\mathcal{R}$) if for all $n, \lambda \in \mathbb{N}$, for all input triples $(\mathsf{i}, \vec{x}, \vec{w}) \notin \mathcal{R}_n$ the following holds:*

$$\Pr_{q \leftarrow \mathbb{S}\mathbb{P}_\lambda}\left[(\mathsf{i}, [\![\vec{x}]\!]_q, [\![\vec{w}]\!]_q) \in [\![\mathcal{R}_n]\!]_q\right] \leq \mathsf{negl}(\lambda)$$

3.2 R1CS over $\mathbb{Z}$ and its Fingerprint

We provide a general definition of Rank-1 Constraint Satisfiability (R1CS) over arbitrary commutative rings.

Definition 5 (Rank-1 Constraint Satisfiability). *Let $\mathbb{A}$ be a commutative ring. An R1CS triple over $\mathbb{A}$ (or $\mathbb{A}$-R1CS) of size N consists of three matrices $A, B, C \in \mathbb{A}^{N\times N}$ each having at most $O(N)$ non-zero entries. A pair statement–witness for an R1CS triple consists of vectors $(\vec{x}, \vec{w})$ with elements in $\mathbb{A}$ such that $|\vec{x}| + |\vec{w}| = N$. We say that $(\vec{x}, \vec{w})$ satisfies the R1CS if*

$$(A \cdot \vec{z}) \circ (B \cdot \vec{z}) - C \cdot \vec{z} = \vec{0} \tag{\dagger}$$

where $\vec{z} = (\vec{x}, \vec{w}) \in \mathbb{A}^N$.

In this paper, when we consider an $\mathbb{F}$-R1CS over some field $\mathbb{F}$ we always implicitly assume that $\mathbb{F}$ is of prime order.

Definition 6 (R1CS Relation over the Integers). *We denote by $\mathcal{R}^{R1}$ the relation parametrized by a bound function $b(\cdot)$ (polynomial in λ) and whose syntax is as follows: on input an index i (describing an R1CS triple A, B, C over $\mathbb{Z}$ of size N), a statement x and a witness w returns 1 if and only if: (i) Eq. (†) is satisfied; and (ii) x and w are such that $\|\mathsf{x}||\mathsf{w}\|_\infty \leq 2^{b(N)}$. The size of the index is given by $N \cdot \log_2(\|A||B||C\|_\infty)$.*

From now on we refer to an R1CS structure as "R1CS" for short. We write $\mathbb{F}$-R1CS to refer to an R1CS over some prime-order field $\mathbb{F}$. We write $\mathbb{Z}$-R1CS to refer to an R1CS in the sense of Definition 5.

Definition 7 (Fingerprinting for R1CS). *We define the associated fingerprinting relation for R1CS as the one that checks the R1CS constraint equations over $\mathbb{F}_q$ for a prime q, i.e. if i encodes R1CS matrices A, B, C then*

$$(\mathsf{i}, [\![\vec{x}]\!]_q, [\![\vec{w}]\!]_q) \in [\![\mathcal{R}_n^{R1CS}]\!]_q \iff \text{for all } i\ \langle A_i, \vec{z}\rangle \circ \langle B_i, \vec{z}\rangle - \langle C_i, \vec{z}\rangle \equiv 0 \mod q$$

where $\vec{z} := (\vec{x}||\vec{w})$.

It is easy to check that the fingerprinting relation above is admissible. In Lemma 3 we also show it is a good membership test, which follows from this technical lemma:

Lemma 2 (Probabilistic Vanishing of Integer Polynomials). *Let $f(\vec{X}) \in \mathbb{Z}[\vec{X}]$ be a polynomial of total degree d with less than 2^ℓ non-zero coefficients. Denote by n the smallest integer such that $\|f\|_\infty \leq 2^n$. Fix $\vec{x} \in \mathbb{Z}^\mu$ and denote by m the smallest integer such that $\|\vec{x}\|_\infty \leq 2^m$. Then $f(\vec{x}) \neq 0$ implies:*

$$\Pr_{q \leftarrow\$ \mathbb{P}_\lambda}[f(\vec{x}) \equiv 0 \bmod q] \leq \frac{\lambda \cdot (\ell + m \cdot d + n)}{2^{\lambda-1}}$$

Which is negligible for any ℓ, m, d, n polynomial in λ.

Proof. The evaluation $y = f(\vec{x})$ has a norm $\|y\|_\infty \leq 2^{\ell+m\cdot d+n}$: it is a summation of at most 2^ℓ non-zero terms, each of norm at most $2^{m\cdot d+n}$. Now consider the set $\mathcal{P}(y)$ of distinct prime factors of y and observe that $y \equiv 0 \mod q \iff q \in \mathcal{P}(y)$. Therefore, by Theorem 1, the probability that $y \equiv 0 \mod q$ for a uniformly sampled prime $q \in \mathbb{P}_\lambda$ and any λ with $2^\lambda > 55$ is:

$$\begin{aligned}\Pr_{q \leftarrow\$ \mathbb{P}_\lambda}[f(\vec{x}) \equiv 0 \bmod q] &= \frac{|\mathcal{P}(y)|}{|\mathbb{P}_\lambda|} \leq \frac{\log_2(y)}{|\mathbb{P}_\lambda|} \leq \frac{\ell + m \cdot d + n}{|\mathbb{P}_\lambda|} = \frac{\ell + m \cdot d + n}{\pi(2^\lambda) - \pi(2^{\lambda-1})} \\ &\leq \frac{\ell + m \cdot d + n}{\frac{2^\lambda}{\ln(2^\lambda)+2} - \frac{2^{\lambda-1}}{\ln(2^{\lambda-1})-4}} \leq \frac{\ell + m \cdot d + n}{\frac{2^\lambda - 2^{\lambda-1}}{\lambda}} = \frac{\lambda \cdot (\ell + m \cdot d + n)}{2^{\lambda-1}}\end{aligned}$$

□

Lemma 3. *The associated fingerprinting relation for integer R1CS in Definition 7 provides a good membership test.*

Proof. Consider $(\mathsf{i}, \vec{x}, \vec{w}) \notin \mathcal{R}_n$. If the statements are such that the bound b in Definition 6 then we are done. Otherwise let us proceed as follows and let us bound the probability that:

$$\text{for all } j,\ \langle A_j, \vec{z}\rangle \circ \langle B_j, \vec{z}\rangle - \langle C_j, \vec{z}\rangle \equiv 0 \mod q \tag{1}$$

for a randomly sampled prime q of λ bits and $\vec{z} := (1, \vec{x}, \vec{w})$. Since $(\mathsf{i}, \vec{x}, \vec{w}) \notin \mathcal{R}_n$ there must exist index j^* such that

$$\langle \vec{a}, \vec{z}\rangle \circ \langle \vec{b}, \vec{z}\rangle - \langle \vec{c}, \vec{z}\rangle \neq 0$$

where the operations in the last equality are over the integers and $\vec{a} := A_{j^*}, \vec{b} := B_{j^*}, \vec{c} := C_{j^*}$. Let $y := \langle \vec{a}, \vec{z}\rangle \circ \langle \vec{b}, \vec{z}\rangle - \langle \vec{c}, \vec{z}\rangle$. The probability that Eq. (1) holds is bounded from above by the probability that q divides y. We can bound this probability by a quantity negligible in λ through a straightforward invocation of Lemma 2 seeing y as the evaluation of a low-degree polynomial in $\vec{z}$ (notice that for that we use the norm bound requirements from Definition 6 and the bounded coefficients of the R1CS index). □

3.3 Multilinear Polynomial Evaluation and its Fingerprint

We also define another indexed relation that will be useful for our results in Section H—polynomial evaluation—as well as its straightforward associated fingerprinting. For simplicity we define it only for the multilinear case.

Definition 8. *The deterministic relation $\mathcal{R}^{poly}$ takes as input an index describing a multilinear polynomial f in μ variables and a statement—a pair $(\vec{x} \in \mathbb{Z}^\mu, y \in \mathbb{Z})$. It returns 1 iff $f(\vec{x}) = y$. The size of the index describing a multilinear polynomial f is 2^μ, the maximum number of its non-zero coefficients.*

Definition 9 (Fingerprinting for Polynomial Evaluation). *We define the associated fingerprinting relation for polynomial evaluation as the (deterministic) relation that checks the polynomial evaluation over $\mathbb{F}_q$ for a prime q, i.e.,*

$$(\mathsf{i} := f, [\![(\vec{x}||y)]\!]_q, \perp) \in [\![\mathcal{R}^{poly}]\!]_q \iff f(\vec{x}) \equiv y \mod q$$

4 Idealized Protocols for Arguments over $\mathbb{Z}$

4.1 Algebraic Holographic Proofs with Modular Remainder Queries

An AHP over $\mathbb{Z}$ with modular remainder queries (or mod-AHP) is like a standard AHP-like protocol [CHM+20, CFF+21, BFS20] with the following core differences: the oracles, both in the indexing and online stage, are (multivariate) polynomials over the integers (rather than over a finite field); at a prespecified round the verifier samples a random prime q and the interaction continues as a standard interactive proof; after the interaction, the verifier can receive evaluations of the oracle (integer) polynomials modulo q.

Definition 10 (AHP over $\mathbb{Z}$ with modular remainder queries (mod-AHP)). *An Algebraic Holographic Proof (AHP) over $\mathbb{Z}$ with modular remainder queries (or mod-AHP) for an indexed relation $\mathcal{R}$ is given by the following tuple:*

$$\mathsf{modAHP} = (\mathsf{k}, \mathsf{k}', \mathsf{v}, \mathsf{s}, \mathsf{d}, \mathcal{I}, \mathsf{P}, \mathsf{V})$$

where $\mathsf{k}, \mathsf{k}', \mathsf{v}, \mathsf{s}, \mathsf{d} : \{0,1\}^* \to \mathbb{N}$ are polynomial-time computable functions; $\mathcal{I}, \mathsf{P}, \mathsf{V}$ are the indexer, prover, and verifier algorithms; k is the number of oracle polynomial rounds; k' is the number rounds of "plain interaction" (see below); v is the number of variables in the multivariate oracle polynomials[17]; s is the number of polynomials in each round; d specifies degree bounds (in each variable) on these polynomials. The protocol proceeds as follows (and continues on next page):

[17] We assume for simplicity that the number of variables is the same for all the polynomials provided at the same round and that s depends only on the round index.

Indexing phase *The indexer $\mathcal{I}$ receives as input a security parameter 1^λ and the index i for $\mathcal{R}$, and outputs $\mathsf{s}(0)$ polynomials $p_{0,1}, \ldots, p_{0,\mathsf{s}(0)} \in \mathbb{Z}[\vec{X}]$ of degrees at most $\mathsf{d}(\lambda, |\mathsf{i}|, 0, 1), \ldots, \mathsf{d}(\lambda, |\mathsf{i}|, 0, \mathsf{s}(0))$ respectively; $|\vec{X}| = \mathsf{v}(\lambda, |\mathsf{i}|, 0)$. This phase does not depend on the public input or witness and simply consists of encoding the given index i. We require that each $p_{0,j}$ is such that $\|p_{0,j}\|_\infty$ is bounded w.r.t to b as in Definition 6.*
Online phase *The prover P receives $(1^\lambda, \mathsf{i}, \mathsf{x}, \mathsf{w})$, for an instance x and witness w such that $(\mathsf{i}, \mathsf{x}, \mathsf{w}) \in \mathcal{R}$. The verifier V receives 1^λ, x and oracle access to the polynomials output by $\mathcal{I}(1^\lambda, \mathsf{i})$[a]. The prover P and the verifier V interact over a number of rounds as follows:*

- ***Integer Oracle Polynomials Phase:*** *In the i-th round, $i \in \{1, \ldots, \mathsf{k}(\lambda, |\mathsf{i}|)\}$, the verifier V sends messages $\vec{\rho}_i \in \{0,1\}^{\mathsf{poly}(\lambda)}$ to the prover P; the prover P responds with $\mathsf{s}(i)$ oracle polynomials $p_{i,1}, \ldots, p_{i,\mathsf{s}(i)} \in \mathbb{Z}[\vec{X}]$ where each is respectively of degree at most $\mathsf{d}(\lambda, |\mathsf{i}|, i, 1), \ldots, \mathsf{d}(\lambda, |\mathsf{i}|, i, \mathsf{s}(i))$, $|\vec{X}| = \mathsf{v}(\lambda, |\mathsf{i}|, i)$.*
- ***Prime Sampling Phase:*** *After k rounds, the verifier samples a prime $q \leftarrow\!\!\$\ \mathbb{P}_\lambda$ and sends it to P.*
- ***Plain Interaction Phase:*** *The prover and verifier engage in a plain interactive protocol (see Section 2) for k' rounds:*

$$\mathsf{tr}_{rst} := (\vec{m}_{rst}, \vec{\rho}_{rst}) \leftarrow \mathsf{transcript}_\lambda\big(\langle \mathsf{P}_{rst}(\vec{\rho}_1, \ldots, \vec{\rho}_\mathsf{k}, q), \mathsf{V}_{rst}\rangle\big)$$

(Recall that by convention $\vec{m}_{rst}$ and $\vec{\rho}_{rst}$ denote the concatenation of respectively all the messages and challenges sent during the interaction)
Query phase *Let $\mathbf{p} = (p_{i,j})_{i \in \{0,1,\ldots,\mathsf{k}\}, j \in [\mathsf{s}(i)]}$ be a vector consisting of all the polynomials sent by the indexer $\mathcal{I}$ and prover P. The verifier V executes a subroutine Q_V that receives $(1^\lambda, \mathsf{x}; \vec{\rho}_1, \ldots, \vec{\rho}_\mathsf{k}, \mathsf{tr}_{rst}, q)$ and outputs a query set Q consisting of tuples $((i,j), z)$ that are interpreted as "query $p_{i,j}$ at $z \in \mathbb{F}_q^{\mathsf{v}(|\mathsf{i}|)}$", where q is the prime sampled earlier. We denote the vector consisting of the answers to these queries as $\mathbf{p}(Q)$.*
Decision phase *The verifier outputs* accept *or* reject *based on the answers received to the queries and its randomness. That is, V executes a subroutine D_V which outputs a decision bit on input $(1^\lambda, \mathsf{x}, \mathbf{p}(Q); \vec{\rho}_1, \ldots, \vec{\rho}_\mathsf{k}, \mathsf{tr}_{rst}, q)$.*

[a] The specific type of oracle access is defined in **Query phase**.

The function d determines what kind of provers are considered for the completeness and soundness properties of the proof system[18]*. A (potentially malicious) prover $\tilde{\mathsf{P}}$ is considered admissible for* modAHP *if, in an interaction with the verifier V, it holds with overwhelming probability that for every round $i \in [\mathsf{k}]$ and oracle index $j \in [\mathsf{s}(i)]$, variable index $t \in [\mathsf{v}(\lambda, |\mathsf{i}|, i)]$ we have $\deg(p_{i,j}, X_t) \leq \mathsf{d}(\lambda, |\mathsf{i}|, i, j)$. We also require that each $p_{i,j}$ is s.t. that $\|p_{i,j}\|_\infty$ is bounded w.r.t to b as in Definition 6. The honest prover P is required to be admissible under this definition. A mod-AHP should always satisfy completeness as defined below.*

Completeness *A mod-AHP is complete if for any $\lambda \in \mathbb{N}, (\mathsf{i}, \mathsf{x}, \mathsf{w}) \in \mathcal{R}$, the output returned by $\mathsf{V}^{\mathcal{I}(1^\lambda, \mathsf{i})}(1^\lambda, \mathsf{x})$ interacting with the honest $\mathsf{P}(1^\lambda, \mathsf{i}, \mathsf{x}, \mathsf{w})$ is 1.*

The following notion states that we can extract an integer witness interacting with a successful AHP prover. This is the type of knowledge soundness we would like to require from any "secure" AHP over the integers. We dub it "full" to stress the difference with the weaker "fingerprint-only" definition in Definition 13. We will later show (Theorem 8) that weaker knowledge soundness can at times be immediately lifted to obtain full knowledge soundness.

Definition 11 (Full Knowledge Soundness). *We say that a mod-AHP has (full) knowledge error ϵ if there exists a probabilistic polynomial-time extractor* Ext *such that for any admissible prover P^*, for every $\lambda \in \mathbb{N}$, (i, x), and auxiliary input* aux*:*

$$\Pr\left[\begin{array}{c}(\mathsf{i}, \mathsf{x}, \mathsf{w}) \notin \mathcal{R}_\mathsf{n} \wedge \\ \langle \mathsf{P}^*(1^\lambda, \mathsf{i}, \mathsf{x}, \mathsf{aux}), \mathsf{V}^{\mathcal{I}(1^\lambda, \mathsf{i})}(1^\lambda, \mathsf{x})\rangle = 1\end{array} \;:\; \mathsf{w} \leftarrow \mathsf{Ext}^{\mathsf{P}^*}(1^\lambda, \mathsf{i}, \mathsf{x}, \mathsf{aux})\right] \leq \epsilon$$

Here the notation $\mathsf{Ext}^{\mathsf{P}^*}$ means that the extractor Ext *has black-box access to each of the next-message functions that define the interactive algorithm P^*. (In particular, the extractor can "rewind" the prover.)*

[18] We will call a mod-AHP *multilinear* if d is the constant function equal to 1.

4.2 Weak Knowledge Sound mod-AHPs

In this section we prove basic properties of mod-AHPs so to make our framework more "plug-and-play". We define a "weaker"—and easier to prove—notion of extractability for AHP over $\mathbb{Z}$ with modular remainder queries and we later show that this notion implies the stronger one in Definition 10. This result is interesting when combined with our results in Section 5.2. Together with the result of Theorem 3 this section shows that, in order to obtain a succinct interactive argument for $\mathcal{Z}$ all we need is to to show an AHP with the weaker property in this section. This is advantageous because this is easier to prove directly for a given protocol (and will allow the very simple proof of the statement of Theorem 4).

For notational convenience, in Definition 13 we restrict the mod-AHP interaction to be "simple", i.e., to send a single oracle polynomial.

Definition 12 (Simple mod-AHP). *We say that a mod-AHP is simple (or, "it has a simple prover") if there is a single round in which the prover sends oracle polynomials, that is if* $\mathsf{k}(\lambda, N) = 1$ *(see Definition 10).*

Remark 1 (Notation for simple mod-AHPs). Let $\mathsf{tr} \leftarrow \mathsf{transcript}\big(\langle \mathsf{P}(1^\lambda, \mathsf{i}, \mathsf{x}, \mathsf{aux}), \mathsf{V}^{\mathcal{I}(1^\lambda,\mathsf{i})}(1^\lambda, \mathsf{x})\rangle\big)$, be the interaction transcript. We can parse tr as $\Big(g(\vec{X}), q, \mathsf{tr}_{\mathrm{rst}})\Big)$ where $g(\vec{X})$ is the oracle sent during the interaction and q is the sampled prime. A simple prover P can always be split into a pair of (stateful) algorithms $(\mathsf{P}_{\mathrm{orcl}}, \mathsf{P}_{\mathrm{rst}})$

This weaker notion informally states that we can extract in a straight-line manner a "witness modulo a prime" (in the sense of Definition 7) when receiving as input the oracle polynomials from the prover. Notice that this is a weaker notion because we are not requiring to extract an integer witness for the original integer relation, rather for its restriction. Below we use the terminology "decoding" for this since we are not extracting a proper witness for $\mathcal{R}$. We require this weak extractor/decoder to be "partly" straight-line in that we give it as input the oracle polynomial from the prover. Intuitively, this is necessary to make sure that the extractor is always dealing with the same integer witness. Also notice, and this is crucial, that the oracle polynomial committed by the prover is still over the integers, not over the field $\mathbb{F}_q$.

Definition 13 (Weak ("fingerprint-only") Knowledge-Soundness). *Consider a mod-AHP with a simple prover (Definition 10 and Definition 12) for an indexed relation* $\mathcal{R}$*. Let* $[\![\cdot]\!]$ *be an associated fingerprinting relation for* $\mathcal{R}$ *(Definition 3). We say the mod-AHP has weak knowledge error* ϵ *over* $\mathcal{R}$ *and* $[\![\cdot]\!]$ *if there exists an efficient deterministic decoding algorithm* Dec *such that for any admissible prover* P^**, for every* $\lambda, n \in \mathbb{N}$*, index* i*, statement* x*, and auxiliary input* aux*:*

$$\Pr\left[\begin{array}{c}(\mathsf{i}, [\![\mathsf{x}]\!]_q, [\![\mathsf{w}]\!]_q) \notin [\![\mathcal{R}_n]\!]_q \wedge \\ \langle \mathsf{P}^*(1^\lambda, \mathsf{i}, \mathsf{x}, \mathsf{aux}), \mathsf{V}^{\mathcal{I}(1^\lambda,\mathsf{i})}(1^\lambda, \mathsf{x})\rangle = 1\end{array} : \mathsf{w} \leftarrow \mathsf{Dec}(1^\lambda, \mathsf{i}, g^*(\vec{X}))\right] \leq \epsilon$$

$\Big(g^*(\vec{X}), q, \dots\Big) \leftarrow \mathsf{transcript}\Big(\langle \mathsf{P}^*(1^\lambda, \mathsf{i}, \mathsf{x}, \mathsf{aux}), \mathsf{V}^{\mathcal{I}(1^\lambda,\mathsf{i})}(1^\lambda, \mathsf{x})\rangle\Big)$ *is as by Remark 1.*

The following theorem states that weak knowledge soundness can be lifted to obtain full knowledge soundness if its associated fingerprinting relation provides a good membership test (Definition 4). By applying Lemma 3 we can interpret this as a lifting theorem for weak knowledge sound mod-AHPs for R1CS.

Theorem 2. *Let* modAHP *be a mod-AHP with negligible weak knowledge soundness error over* $\mathcal{R}$ *and* $[\![\cdot]\!]$ *(Definition 13). If* $[\![\cdot]\!]$ *provides a good membership test (Definition 4) then* modAHP *has negligible full knowledge soundness error (Definition 11).*

5 Integer Polynomial Commitments with Evaluation Opening over $\mathbb{Z}_q$

Here we first define and then show how to apply a special form of polynomial commitment that can *commit* to polynomials with *integer* coefficients but that can efficiently prove evaluations modulo a random prime q. We will use such schemes to compile a mod-AHP into an argument for *integer computations*.

Remark 2 (Difference with field-agnostic commitments). A field-agnostic polynomial commitment is one that can work over arbitrary (sufficiently large) finite fields [GLS+23]. The commitments defined in this section are in general more powerful than field-agnostic ones because they can commit to a polynomial with arbitrary integer coefficients and later open its evaluation over any prime-order field; in contrast, field-agnostic commitments must receive a description of the field at commitment time already.

5.1 Model

Definition 14. *A polynomial commitment with modular remainder opening (or mod-PC) consists of a tuple* $(\mathsf{Setup}, \mathsf{Com}, \mathsf{Prv}, \mathsf{Vfy})$ *such that:*

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$*: on input a security parameter* λ*, it outputs public parameters.*
- $\mathsf{Com}(\mathsf{pp}, f) \to (c, \mathsf{opn})$*: on input public parameters, a multilinear polynomial over the integers* $f \in \mathbb{Z}[\vec{X}]$*, it outputs a commitment* c *and an additional opening string* opn *(used as auxiliary input for opening).*
- $\mathsf{Prv}(\mathsf{pp}, q, c, \mathsf{opn}, \vec{x}) \to \pi$*: on input parameters* pp*, prime* q*, commitment* c*, opening* opn *and* $\vec{x} \in \mathbb{Z}^\mu$*, it outputs a proof* π *certifying the value* $y \equiv f(\vec{x}) \bmod q$*.*
- $\mathsf{Vfy}(\mathsf{pp}, q, c, \vec{x}, y, \pi, \mu) \to b$*: on input public parameters, prime* q*, commitment* c*, claimed value* $y \in \mathbb{Z}_q$*, proof* π*, and number of variables* μ*, it outputs a bit accepting or rejecting the proof.*

Correctness. For any $\mu, \lambda \in \mathbb{N}$, $f(\vec{X})$ of individual degree at most 1, prime $q < 2^\lambda$ and $\vec{x} \in \mathbb{Z}^\mu$, the following probability is overwhelming:

$$\Pr\left[\mathsf{Vfy}\left(\mathsf{pp}, q, c, y, \pi, \mu\right) = 1 \;:\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ (c, \mathsf{opn}) \leftarrow \mathsf{Com}(\mathsf{pp}, f) \\ \pi \leftarrow \mathsf{Prv}(\mathsf{pp}, q, c, \mathsf{opn}, \vec{x}) \\ y \coloneqq f(\vec{x}) \bmod q \end{array}\right]$$

Weak evaluation binding. The following property is the analogue for mod-PCs of weak evaluation binding for functional commitments (see, e.g., [CFT22, CFK24]). It intuitively states that for an honestly generated commitment (hence the relative "weakness" of the property), it should not be feasible to provide a convincing false proof. We define it formally in the appendix.

Knowledge soundness (with knowledge error ϵ*).* This notion follows its counterpart in [CHM+20].

Definition 15. *For any* $\lambda, \mu \in \mathbb{N}$ *and PPT* $\mathcal{A} = (\mathcal{A}_{com}, \mathcal{A}_{prv})$ *there exists a non-uniform polynomial time extractor* Ext *such that for any efficient query algorithm (with random tape independent from that of the adversary)* $\mathcal{Q}$ *auxiliary string* $\mathsf{aux} \in \{0,1\}^{\mathsf{poly}(\lambda)}$*, the following probability is at most* ϵ*:*

$$\Pr\left[\begin{array}{l} \Big(f(\vec{X}) \notin \mathbb{Z}_{\le 1}[X_1, \ldots, X_\mu] \;\vee \\ \quad \exists j \in [m].\ f(\vec{x}_j) \not\equiv y_j \bmod q\Big) \;\wedge \\ \bigwedge_j \mathsf{Vfy}\left(\mathsf{pp}, q, c, \vec{x}_j, y_j, \pi_j, \mu\right) = 1 \end{array} \;:\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ ((c, \mu), \mathsf{st}) \leftarrow \mathcal{A}_{com}(\mathsf{pp}, \mathsf{aux}) \\ q \leftarrow\!\!\$\ \mathbb{P}_\lambda \\ \left((\vec{x}_j)_{j\in[m]}, \mathsf{aux}_{\mathcal{Q}}\right) \leftarrow \mathcal{Q}(\mathsf{pp}, \mathsf{aux}, q) \\ (y_j, \pi_j)_{j\in[m]} \leftarrow \mathcal{A}_{prv}(\mathsf{st}, q, (\vec{x}_j)_j, \mathsf{aux}_{\mathcal{Q}}) \\ f(\vec{X}) \leftarrow \mathsf{Ext}(\mathsf{pp}, \mathsf{aux}) \end{array}\right]$$

where above the extractor has access to the random tape of the adversary.

Remark 3 (Interactive Openings). All our definitions in this section involve a non-interactive opening stage. We point out they can be adapted straightforwardly for the interactive setting. We do not provide explicit formal variants of our definitions since we will use mod-PCs with weak evaluation binding and interactive opening only in Theorem 9.

Definition 16 (mod-PC for sparse polynomials). *We say that mod-PC is "for sparse (multilinear) polynomials" (Section 2) if the running time of the* Prv *stage is linear in* $\ell \cdot \log_2 \|f\|_\infty$ *where* ℓ *is the number of non-zero coefficients of the dense representation of* f*.*

5.2 mod-PCs in Action: compiling mod-AHPs into SNARKs over $\mathbb{Z}$

To build SNARKs from idealized protocols, we follow the approach "commit to oracles; prove query responses" [Kil92], that is also the blueprint beneath standard compilers in this space (see [CHM+20, CFF+21, ABC+22]). Concretely, this means that we use a polynomial commitment to commit to each of the oracle polynomials; at the end of the interaction the prover responds to the oracle queries and "certifies" them through the polynomial commitment opening. The key difference with prior works is that here we need to follow the more nuanced query semantics of mod-AHPs: all the queries will depend on (and occur after) a prime sampled during the interaction. For this, we rely on our special flavor of polynomial commitment. The compiler we just sketched is fully described in Fig. 2. The arguments from the next theorem can be heuristically made non-interactive through Fiat-Shamir.

Setup$(1^\lambda, N)$**:** $\mathsf{pp}_{\mathsf{PC}} \leftarrow \mathsf{PC}_{\mathrm{prj}}.\mathsf{Setup}(1^\lambda)$ Return $\mathsf{srs} = \left(1^\lambda, \mathsf{pp}_{\mathsf{PC}}\right)$

Indexer$(\mathsf{srs}, \mathsf{i})$**:** Run the AHP indexer on input the index i and the security parameter 1^λ to obtain $\mathsf{s}(0)$ polynomials $(g_{0,j})_{j=1}^{\mathsf{s}(0)}$ each in $\mathbb{Z}[\vec{X}]$. Compute, for $j = 1, \ldots, \mathsf{s}(0)$: $\left(c_{0,j}, \mathsf{opn}_{0,j}\right) \leftarrow \mathsf{PC}_{\mathrm{prj}}.\mathsf{Com}\left(\mathsf{pp}_{\mathsf{PC}}, g_{0,j}\right)$. Output $(\mathsf{ipk}, \mathsf{ivk})$ where $\mathsf{ipk} = \left(\mathsf{i}, \left(\mathsf{opn}_{0,j}, c_{0,j}\right)_{j=1}^{\mathsf{s}(0)}\right)$, $\mathsf{ivk} = \left(c_{0,j}\right)_{j=1}^{\mathsf{s}(0)}$

Interaction: This consists of several rounds of interaction between the prover and the verifier of the underlying AHP, i.e., $\mathsf{P}(\mathsf{i}, \mathsf{x}, \mathsf{w})$ and $\mathsf{V}(\mathsf{i}, \mathsf{x})$:

- Integer Polynomial Oracle phase: For each "oracle" round $i \in [\mathsf{k}]$:
  1. $\mathcal{V}$ receives random challenges $\vec{\rho}_i$ from V and forwards them to $\mathcal{P}$.
  2. $\mathcal{P}$ forwards $\vec{\rho}_i$ to P which replies with polynomials $(g_{i,j})_{j=1}^{\mathsf{s}(i)}$.
  3. $\mathcal{P}$ compute commitments
  $$\left(c_{i,j}, \mathsf{opn}_{i,j}\right) = \mathsf{PC}_{\mathrm{prj}}.\mathsf{Com}\left(\mathsf{pp}_{\mathsf{PC}}, g_{i,j}\right), \quad j = 1, \ldots, \mathsf{s}(i)$$
  and sends $\left(c_{i,j}\right)_{j=1,\ldots,\mathsf{s}(i)}$ to $\mathcal{V}$.
- Prime sampling: After k rounds, $\mathcal{V}$ obtains a prime q and sends it to $\mathcal{P}$.
- Plain-Interaction phase: here $\mathcal{P}, \mathcal{V}$ follow exactly the mod-AHP protocol:
  $$\mathsf{tr}_{\mathrm{rst}} := (\vec{m}_{\mathrm{rst}}, \vec{\rho}_{\mathrm{rst}}) \leftarrow \mathsf{transcript}_\lambda\left(\langle \mathsf{P}_{\mathrm{rst}}(\vec{\rho}_1, \ldots, \vec{\rho}_{\mathsf{k}}, q), \mathsf{V}_{\mathrm{rst}}\rangle\right)$$
- Query Phase:
  1. $\mathcal{P}$ uses the query algorithm of V to compute the query set $Q := Q_{\mathsf{V}}\left(1^\lambda, \mathsf{x}; \vec{\rho}_1, \ldots, \vec{\rho}_{\mathsf{k}}, \mathsf{tr}_{\mathrm{rst}}, q\right)$. Recall from Definition 10 that each entry in Q is of the form $((i,j), z)$.
  2. $\mathcal{P}$ computes and sends $\mathbf{v} := \mathbf{g}(Q)$ and $\boldsymbol{\pi}$ where
  $$\mathbf{g}(Q) := \left(g_{i,j}(z) \bmod q : ((i,j), z) \in Q\right)$$
  $$\boldsymbol{\pi} := \left(\mathsf{PC}_{\mathrm{prj}}.\mathsf{Prv}\left(\mathsf{pp}_{\mathsf{PC}}, q, c_{i,j}, \mathsf{opn}_{i,j}, z\right) : ((i,j), z) \in Q\right)$$
- Decision Phase: $\mathcal{V}$ accepts if and only if all the following conditions hold:
  - the decision algorithm of V accepts the answers, that is $\mathsf{D}_{\mathsf{V}}\left(1^\lambda, \mathsf{x}, \mathbf{v}, \vec{\rho}_1, \ldots, , \vec{\rho}_{\mathsf{k}}, \mathsf{tr}_{\mathrm{rst}}, q\right) = 1$.
  - the evaluation proofs verify, i.e. $\mathsf{PC}_{\mathrm{prj}}.\mathsf{Vfy}\left(\mathsf{pp}_{\mathsf{PC}}, q, c_{i,j}, v, \pi\right) = 1$ for each $((i,j), z)$ in Q and corresponding value v in $\mathbf{v}$ and proof π in $\boldsymbol{\pi}$.

Fig. 2. Compiler from multilinear mod-AHPs to succinct arguments over $\mathbb{Z}$. Calligraphic letters, $\mathcal{P}$ and $\mathcal{V}$, denote the prover and verifier of the final interactive argument. (The setup algorithm of preprocessing requires a size parameter N, but our compiler can ignore it: the setup for mod-PCs can work for polynomials of any size (polynomial in λ) regardless of how N affects their size)

Theorem 3. *Let* modAHP *be a multilinear AHP over $\mathbb{Z}$ with modular remainder queries (Definition 10) for $\mathcal{R}$, let* PC_{prj} *be a mod-PC (Definition 14) satisfying weak-evaluation binding and with negligible knowledge error, then the construction in Fig. 2 is an interactive argument with preprocessing for $\mathcal{R}$.*

Remark 4 (When is the result of the compilation above a fully succinct argument?). Recall that our goal is to obtain a fully succinct argument, i.e., succinct in both $N = |\mathsf{w}|$ and $m = \log\|\mathsf{w}\|_\infty$ where w is the witness. Succinctness in N is inherited directly from the AHP+PC approach where a few polynomial evaluations (with adequately succinct proofs) "guarantee knowledge" of a witness of size N. The case of succinctness in m is a little different. We are able to achieve it since all polynomial

evaluations in the mod-AHP approach are modulo q, a prime of λ bits (λ being a security parameter). We can conclude that as long as we are able to keep the commitment and the size/verification of the evaluation proofs in the modPC succinct in m, the final proof and verifier will be as well.

One may also use two different types of commitments when compiling, an extractable one for the online oracles and a "weaker" one for indexing polynomials (indexing is run honestly). We discuss this further in Section F.

6 From Multilinear Sumchecks to mod-AHPs

The point of this section is two-fold. On the one hand, it will provide a concrete mod-AHP on which to instantiate our final construction. At the same time, however, it will showcase how our methodology can port existing SNARK technique into the world of integer computations (and do that in a modular way). We will use Spartan [Set20] as a case-study for information theoretic protocols that are also mod-AHPs. While we will discuss only Spartan in detail, *our observations here can be applied to any protocol that at its essence relies on multilinear sumchecks* (see also Remark 5). Besides Spartan, such protocols include all the "non-deterministic" variants of GKR [GKR08] such as those in [ZGK$^+$18, CFQ19, CBBZ23, CGG$^+$23, STW24, CFG25]. We leave it as a future work how to formalize these other works as mod-AHPs.

Remark 5 (Why are multilinear techniques special?). Multivariate techniques suit our fingerprinting approach well since multilinear interpolation works over integers, allowing the indexer and prover to send well-defined integer-coefficient polynomial oracles. In contrast, univariate interpolation faces a fundamental hurdle, as Lagrange interpolation requires division and, as a consequence, fields.

6.1 Background on Spartan

In this section we review Spartan [Set20], a transparent SNARK for R1CS. The first step in Spartan is to encode the R1CS matrices A, B, C, and the vector $\vec{z} = \vec{x}||\vec{w}$ via their multilinear polynomial extensions. Let $\mu = \log N$. Consider the matrix A, which corresponds to the unique multilinear polynomial in 2μ variables, $\tilde{A}$ such that $\tilde{A}(i_1, \ldots, i_\mu, j_1, \ldots, j_\mu) = A(i, j)$, where $(i_1, \ldots, i_\mu)$ is the binary expansion of i, and $(j_1, \ldots, j_\mu)$ is the binary expansion of j. The polynomials $\tilde{B}$ and $\tilde{C}$ are defined similarly, as is the polynomial $\tilde{Z}$ where $Z(i_1, \ldots, i_\mu) = z(i)$. The satisfiability condition then translates to the following polynomial $F(t_1, \ldots, t_\mu)$ being zero at all points of the boolean hypercube:

$$F(\vec{t}) = \left(\sum_{\vec{u} \in \{0,1\}^\mu} \tilde{A}(\vec{t}, \vec{u}) \tilde{Z}(\vec{u}) \right) \cdot \left(\sum_{\vec{u} \in \{0,1\}^\mu} \tilde{B}(\vec{t}, \vec{u}) \tilde{Z}(\vec{u}) \right) - \sum_{\vec{u} \in \{0,1\}^\mu} \tilde{C}(\vec{t}, \vec{u}) \tilde{Z}(\vec{u})$$

Now consider the MLE of $F(\cdot)$, i.e., $Q(\vec{s}) = \sum_{\vec{t} \in \{0,1\}^\mu} F(\vec{t}) \cdot \chi_{\vec{t}}(\vec{s})$ where χ is as in Definition 1. Since $F(\vec{t})$ vanishes on the boolean hypercube, by the Schwartz-Zippel lemma, $Q(\vec{s})$ is identically the zero polynomial[19]. This condition can be verified by evaluating $Q(\vec{s})$ at a random point. Spartan provides an efficient way to check this evaluation. Specifically, to verify the original R1CS, Spartan performs the following over a field $\mathbb{F}$:

1. Prove that $Q(\vec{r}) = 0$ for a random point $\vec{r} \in \mathbb{F}^\mu$. Thanks to the definition of $Q(\cdot)$, this can be done using a sumcheck protocol (Section A.1).
2. This sumcheck reduces to proving that $\sigma = F(\vec{\rho})$ for a random $\vec{\rho} \in \mathbb{F}^\mu$. Due to the structure of F, this is reduced to proving the value of three summations:

$$\sum_{\vec{u} \in \{0,1\}^\mu} \tilde{A}(\vec{\rho}, \vec{u}) \tilde{Z}(\vec{u}), \qquad \sum_{\vec{u} \in \{0,1\}^\mu} \tilde{B}(\vec{\rho}, \vec{u}) \tilde{Z}(\vec{u}), \qquad \sum_{\vec{u} \in \{0,1\}^\mu} \tilde{C}(\vec{\rho}, \vec{u}) \tilde{Z}(\vec{u})$$

   These can also be proven using a sumcheck protocol each; in Spartan, these three sumchecks are aggregated into one.

[19] **NB**: we can make this observation in our setting only if we have already sampled a prime at that point, but this will be the case.

3. Finally, the sumchecks reduce to proving the values of the multilinear extensions at random points, i.e., $\tilde{A}(\vec{r_x}, \vec{r_y})$, $\tilde{B}(\vec{r_x}, \vec{r_y})$, $\tilde{C}(\vec{r_x}, \vec{r_y})$, and $\tilde{Z}(\vec{r_y})$.

The final step is achieved through the use of polynomial commitments. The prover commits to the polynomials $\tilde{A}$, $\tilde{B}$, and $\tilde{C}$ and $\tilde{Z}$. In the next subsection we will explicitly formalize these as oracle polynomials in a mod-AHP.

6.2 Spartan as a mod-AHP

The protocol we just described requires a field only for the sumchecks, but not for the oracle polynomial encoding the witness (or for the indexing polynomials). As a result, we can sample a prime after sending this polynomial. We are able to prove that our variant satisfies weak knowledge soundness by relying on the security of Spartan and by simple properties of MLEs (namely Lemma 1).

We describe the resulting AHP over $\mathbb{Z}$ with modular remainder queries from Spartan in Fig. 3. Compared to the presentation above we made a few changes, some to remain close to the original treatment in [Set20] (e.g., $\tilde{w}$ is morally what we described as $\tilde{Z}$ before). $\mathcal{G}_{io,\tau}$ is a polynomial related to F as described above, $\hat{A}, \hat{B}, \hat{C}$ are related to $\tilde{A}, \tilde{B}, \tilde{C}$ we described above, M_{r_x} is a polynomial intuitively used for batching the sumchecks on the partial evaluations on the indexing polynomials. The syntax $e \leftarrow \langle \mathcal{P}_{SC}(\dots), \mathcal{V}_{SC}(\dots) \rangle (\dots)$ refers to the invocation of a sumcheck returning e as final challenge. We refer the details to [Set20] for details. To maintain visual similarity with [Set20] we do not use the notation $\vec{v}$ for vectors in most of the figure.

Theorem 4. *The protocol in Fig. 3 is a multilinear mod-AHP with negl. weak knowledge soundness over $\mathcal{R}^{R1CS}$ and $[\![\cdot]\!]$ as from Def. 7. For an R1CS of size N it has $O(\log N)$ rounds and a prover linear in the total witness size, or $O_\lambda(mN)$.*

7 How to Build mod-PCs: Core Schemes and Compilers

This section is concerned with approaches to building mod-PCs. We first provide a core construction from unknown-order groups and then two general compilers that transform *any* mod-PC into one with some extra property. The first compiler transforms a mod-PC into one that is succinct in the bit-length of the coefficients of the polynomial, at the price of increasing the number of variables by a logarithmic additive factor. The second compiler lifts a mod-PC into one that is efficient for *sparse* integer polynomials.

7.1 A Core Construction from a New Perspective on DARK

Our construction of mod-PC derives from techniques originally proposed by Bünz et al. [BFS20] which construct polynomial commitments for polynomials $f(\vec{X}) \in \mathbb{F}_p[\vec{X}]$ from groups of unknown order. The resulting commitment scheme is in [BFS20] is for *finite fields*. However, their techniques already have a foot in $\mathbb{Z}$: at their heart, their scheme works by lifting polynomials $f \in \mathbb{F}_p[\vec{X}]$ over the field to polynomials $f \in \mathbb{Z}[\vec{X}]$ over the integers (with bounded coefficients), then committing to the integer polynomials. Because *honest* DARK commitments are commitments to integer polynomials it seems a natural starting point for our application (the dishonest case requires more care, as we discuss below).

Preliminaries on DARK Commitments. To commit to a multi-linear integer polynomial $f(\vec{X}) \in \mathbb{Z}[\vec{X}]$, it is evaluated a point $(\mathsf{q}_1, \dots, \mathsf{q}_\mu) \in \mathbb{Z}^\mu$ where the q_i's are sufficiently large compared to the norm of the coefficients of $f(\vec{X})$ so that the evaluation uniquely determines the polynomial: letting $\mathsf{q}_i = \mathsf{q}^{2^i}$ for a sufficiently large odd value $\mathsf{q} \in \mathbb{Z}$, has the effect of embedding the coefficients of the polynomial as "q'nary" digits of small norm, in a single *very large* integer. The polynomial commitment is simply an integer commitment [FO97] [DF02] to this evaluation: $\mathsf{Com}(\mathsf{pp}, f(\vec{X})) = [f(\mathsf{q}_1, \dots, \mathsf{q}_\mu)] \cdot G \in \mathbb{G}_{\hat{N}}$ in a group of unknown order $\mathbb{G}_{\hat{N}}$. Also, this commitment is (bounded) linearly homomorphic: $\mathsf{Com}(\mathsf{pp}, f_0(\vec{X})) + [v] \cdot \mathsf{Com}(\mathsf{pp}, f_1(\vec{X}) = \mathsf{Com}(\mathsf{pp}, f_0(\vec{X}) + v \cdot f_1(\vec{X}))$ assuming the q_i's are large enough relative to the coefficients on the polynomial $f_0(\vec{X}) + v \cdot f_1(\vec{X}) \in \mathbb{Z}[\vec{X}]$.

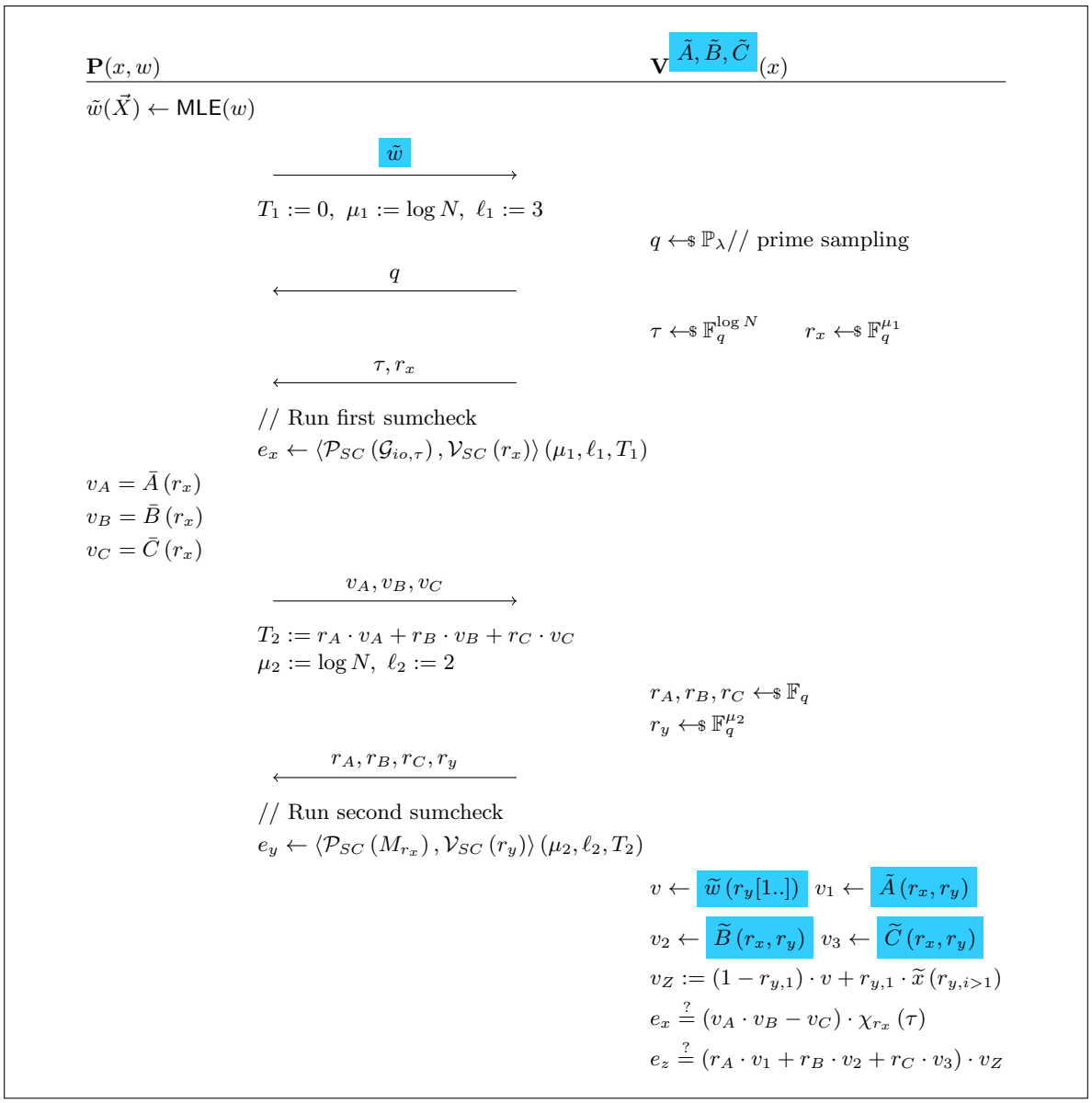


Fig. 3. Spartan as as an AHP over $\mathbb{Z}$ with modular remainder queries. The parameter N denotes the (known) size of the witness. In cyan we denote the integer oracle polynomials and queries to them. Notice that $\tilde{w}, \tilde{A}, \tilde{B}, \tilde{C}$ are all oracle polynomials over the integers. All the final evaluation queries to the oracles are implicitly modulo q (and so are the final checks). Above $r_{y,1}$ is the first element of the vector r_y and $r_{y,i>1} := (r_{y,2}, \ldots, r_{y,\mu_2})$. The figure above simply consists of the online stage; indexing simply returns $\tilde{A}, \tilde{B}, \tilde{C}$.

Towards a mod-PC and Challenges for Showing Soundness. The original DARK construction is for committing to finite fields. In order to obtain a mod-PC—which commits to *integer* polynomials but evaluates them over a *field* provided at a *later stage*—a first idea we explore is a natural adaptation of the recipe in DARK: observe, as we did above, that their commitment stage is syntactically compatible with integer polynomials; then, at proving time on input a prime q[20], run the interactive protocol for proving opening from DARK but *with arithmetic modulo the fresh prime* q.

While the approach above satisfies completeness, it is not sound. The proof of security for DARK (described in [BFS19]) in fact relies on a *weak* notion of binding, which while sufficient for constructing polynomial commitments over large prime fields, is not enough for our application. We now explore this in more detail. The DARK extractor recovers a rational function from the set:

$$\left\{ f(\vec{X})/N \;\middle|\; f(\vec{X}) \in \mathbb{Z}[\vec{X}] \wedge N \in \mathbb{Z} \wedge \|f\|_\infty \leq \beta_N \wedge \|N\|_\infty \leq \beta_D \right\}$$

Where the denominator N is an integer of small norm. In the construction of polynomial commitments over fields, this suffices, since we may view a commitment to the rational polynomial as a commitment to the following polynomial over the prime field: $f'(\vec{X}) \coloneqq f(\vec{X}) \cdot N^{-1} \mod q$. However an issue arises over the integers, because there exists relations satisfied by rational functions for *every prime* but which are not satisfied over the integers: consider for instance the relation $N \cdot w - 1 = 0$ with $N \neq 1$. This relation has no satisfying assignment $w \in \mathbb{Z}$ however letting $w = 1/N$ yields a satisfying assignment over *every prime field* $\mathbb{F}_q$ since $N \cdot (1/N) - 1 = N \cdot 1 \cdot N^{-1} - 1 = 0 \mod q$. Hence without a binding commitment to *integer polynomials* we cannot hope to extract an integer witness from our fingerprinting technique outlined earlier.

Alternative Extraction Strategies. We find a more viable starting point towards a provably extractable mod-PC in the work of Block et al. [BHR+21]. It proposes an alternative construction (close to DARK) which allows the extractor to recover an *integer polynomial with coefficients of bounded norm.* Their scheme circumvents the issue by using *matrices* with $\{0, 1\}$ entries as challenges (rather than integers) which are invertible over the integer ring with overwhelming probability. The drawback is that the complexity of the scheme is increased by a factor of λ. The only difference between the scheme of Block et al. [BHR+21] compared to the original by Bünz et al. is the evaluation proof—the commitment procedure remains the same. We formally describe the resulting construction and its analysis in Section D. This construction is not succinct in the sense explored in this paper.

7.2 A Compiler for Norm-Succinct Polynomial Commitments

In Section 7.1 we outlined the construction Block et. al [BHR+21] which also give rise to a mod-PC, unfortunately, the final round of the protocol has the prover sending a vector $Z \in \mathbb{Z}^\lambda$ with $\|Z\|_\infty \leq 2^m \cdot (2\lambda)^\mu$ where $m = \lceil \log_2 \|f\|_\infty \rceil$ is the bit-size of the largest coefficient in the original multilinear integer polynomial $f \in \mathbb{Z}[X_1, \ldots, X_\mu]$. As a result the verification depends *linearly* on m and as a consequence our verifier would not be poly-logarithmic in the size of the witness which is $O(m \cdot 2^\mu)$.

Reducing The Norm via Reverse Kronecker Substitution. We avoid this issue with a generic technique to reduce the norm of the coefficients of a polynomial f at the cost of increasing the number of variables in the polynomial: allowing the prover to commit to a polynomial with potentially (very) large coefficients, by instead committing to a polynomial of constant norm $\hat{m}$, but with more variables. In more detail, suppose the polynomial $f(\vec{X}) \in \mathbb{Z}[\vec{X}]$ has coefficients of norm $\|f\|_\infty \leq 2^m$, then we instead commit to a polynomial $g(\vec{X}, \vec{Z}) \in \mathbb{Z}[\vec{X}, \vec{Z}]$ with coefficients of norm $\|g\|_\infty \leq 2^{\hat{m}}$:

$$f(\vec{X}) = \sum_{\vec{i} \in \mathcal{H}_\mu} c_{\vec{i}} \cdot \vec{X}^{\vec{i}} \in \mathbb{Z}[\vec{X}] \qquad g(\vec{X}, \vec{Z}) = \sum_{\vec{i} \in \mathcal{H}_\mu} g_{\vec{i}}(\vec{Z}) \cdot \vec{X}^{\vec{i}} \quad \in \mathbb{Z}[\vec{X}, \vec{Z}]$$

Where:

$$g_{\vec{i}}(\vec{Z}) = \sum_{\vec{j} \in \mathcal{H}_{\hat{\mu}}} g_{\vec{i},\vec{j}} \cdot \vec{Z}^{\vec{j}} \in \mathbb{Z}[\vec{Z}] \qquad \forall \vec{j}.\ g_{\vec{i},\vec{j}} \in (-2^{\hat{m}}, 2^{\hat{m}})$$

[20] This has no connection to the odd integer q mentioned earlier. We distinguish them typographically as q (the *prime*) and q (the *radix* used to encode the polynomial when committing to it.

Picked such that $g_i(1, 2^{\hat{m}}, 2^{2\hat{m}}, \ldots, 2^{2^{(\hat{\mu}-1)}\hat{m}}) = c_i$. By constructing the polynomial this way, when we apply Kronecker substitution to the $\vec{Z}$ variables in $g(\vec{X}, \vec{Z})$ it recovers the original polynomial $f(\vec{X})$:

$$f(\vec{X}) = g(\vec{X}, (1, 2^{\hat{m}}, 2^{2\cdot\hat{m}}, \ldots, 2^{2^{\hat{\mu}-1}\cdot\hat{m}}))$$

Note that such a decomposition can be computed very efficiently, e.g. if $\hat{m} = 64$, then the coefficients of $g_{\vec{i}}(\vec{Z})$ are simply the (possibly negative) 64-bit limbs of the coefficients of $c_{\vec{i}}$. The next observation is that during evaluation, *after* the prime q is sampled, the verifier can *efficiently* perform the Kronecker substitution during evaluation:

$$\forall q. \begin{array}{l} f(\vec{x}) \equiv g(\vec{x}, (1, 2^{\hat{m}}, 2^{2\hat{m}}, \ldots, 2^{2^{\hat{\mu}-1}\hat{m}})) \bmod q \\ f(\vec{x}) \equiv g(\vec{x}, (1 \bmod q, 2^{\hat{m}} \bmod q, 2^{2\hat{m}} \bmod q, \ldots, 2^{2^{\hat{\mu}-1}\hat{m}} \bmod q)) \bmod q \end{array}$$

Since computing $2^{2^{\hat{\mu}-1}\cdot\hat{m}} \equiv \left(2^{\hat{m}}\right)^{2^{\hat{\mu}-1}} \bmod q$ requires only poly-logarithmic time in the original m using repeated squaring, where m is the bit-size of the largest coefficient in the original polynomial f.

Corollary 1 (Reverse Kronecker mod-PC). *There exists a mod-PC with weak evaluation binding secure in the GGUO where for any multilinear polynomial f in μ variables with coefficients bounded by $\|f\|_\infty \leq 2^m$, the prover runs in time $O(2^\mu \cdot m \cdot \mathsf{poly}(\lambda, \log(m)))$ while the verifier runs in time $O(\mathsf{poly}(\lambda, \log(m), \mu))$.*

We explain this technique in detail in Section E along with a more exact version of the complexity analysis outlined in Corollary 1.

7.3 A Compiler Obtaining Prover-Efficiency for Sparse Polynomials

In this section we discuss a general transformation from a mod-PC for *dense* polynomials into one where the prover can run efficiently for *sparse* polynomials (see Section 2 and Definition 16). This transformation is useful to obtain a mod-PC which we can apply to the indexing polynomial representing the *sparse* R1CS matrices in Fig. 3. Our starting point will be another compiler in literature to obtain *standard* (for finite fields) sparse polynomial commitments: the SPARK compiler in [Set20]. Our approach to porting the ideas from SPARK will consist of seeing it as a modAHP for a (deterministic) relation, that of polynomial evaluation. While the specific details of SPARK will not be crucial for all of the technical points in this section, but it can provide a slightly more concrete frame for our problem and make our discussion more self-contained.

Brushing up on SPARK: For concreteness we describe the high-level idea behind SPARK for our case of interest: the (sparse) multilinear encoding $\tilde{M}$ of a sparse $N \times N$ matrix M. In order to commit to $\tilde{M}$, the scheme applies the dense polynomial commitment over a constant number of dense polynomials that encode $\tilde{M}$ (and which intuitively encode the non-zero values in M in relation to their position and work as "advice" for later stages of the protocol). These dense polynomialscan be thought of as the *index* for the *deterministic* relation $\tilde{M}(\vec{x}) = y \pmod q$ (q being the prime order of the field, which is *fixed* and known *before hand* in SPARK) and committing to them can be thought of as the indexing stage of the corresponding argument After the public input—consisting of $(\vec{x}, y)$—becomes available, the prover commits to additional polynomials which will depend on the newly available input. The scheme then continues as an interactive protocol, at the end of which the verifier will query all the polynomials it received access to.

*Trying not to reinvent the wheel—*SPARK *as a mod-AHP?* Intuitively, if we could see the protocol above as a mod-AHP for the right relation our job would be then. In fact, we could just apply our compiler from Section 5.2 (instantiated with a dense polynomial commitment) and the resulting SNARK would be the polynomial commitment we were looking for: one for the relation $\tilde{M}(\vec{x}) = y \pmod q$ (this time, with q randomly sampled) and with a "sparse" efficiency profile.

This intuitive plan, however, cannot work directly for a somewhat essential reason[21], that is: in a mod-PC—the object we want to realize—the challenger in its soundness experiment *first* samples a prime q and *then* receives the input $(\vec{x}, y)$; **however**, in the soundness definition of mod-AHPs *this*

[21] There is actually one more reason why this cannot work immediately, but it is more superficial and very easy to fix: that, in addition to integer oracle polynomials, we now also have polynomials with field coefficients (sent after sampling the prime). It is straightforward to extend our framework in this sense (see Section J.1).

is impossible because the adversary can see the sampled prime *only after* it has provided an (alleged) oracle polynomial encoding to the integer witness—and therefore definitely *after* it has declared its public input.

When *looking at general relations*, this feature of the soundness of mod-AHPs is not optional and therefore, in this sense, is not some type of technical artifact we could do without through different definitions[22]. *Nonetheless*, in the more limited setting of *deterministic* relation with *integer* indices (e.g., the polynomial $\tilde{M}$) but with public inputs "modulo the freshly sampled prime" (e.g., the evaluation input $\vec{x}$ and output y), we can show that a different form of soundness can be achieved (and that SPARK satisfies it). We are thus able to extend our framework with the notion of *delayed-input deterministic soundness* and, as a consequence, to import the techniques in SPARK. More importantly, once done the formal work in the framework, showing the soundness of the resulting protocol is trivial. We thus have the following result[23][24]:

Corollary 2. *There exists a mod-PC for sparse polynomials with weak evaluation binding secure in the GGUO model (Section B)and with knowledge error ϵ. In particular, for a sparse multilinear polynomial g in 2μ variables and dense representation of size $O(2^\mu)$, the prover runs in time $O_\lambda(2^\mu \cdot \log_2 \|g\|_\infty)$.*

Remark 6 (A note on functional commitments "modulo q"). We believe that the "deterministic" version of our framework discussed in this section (and developed in Section H)can be of independent interest: it provides a language and basic results to bootstrap *functional commitments over integer inputs* with opening to any *"modular remainder" restriction of a function f*—that is, a generalization of our notion of mod-PC (where f is polynomial evaluation). In this section we used it specifically to bootstrap sparse mod-PCs from mod-PCs for dense polynomials. A potential object of interest could be for example subvector commitments [LM19] for $\mathbb{Z}^n$ where arbitrary subvectors can be opened in $\mathbb{Z}_q$ for a sampled q. Such a subvector commitment could in principle be useful to compile an IOP counterpart ([BCS16]) of our mod-AHPs). We leave this as an interesting open problem.

8 Putting it All Together: Zaratan

Our final construction Zaratan is the result of compiling the mod-AHP in Section 6 with the compiler from Section 5.2. We instantiate the polynomial commitment for compilation through our construction and compilers in Section 7, more specifically (see Fig. 3 for reference): we compile the witness oracle polynomial $\tilde{w}$ with the construction from Corollary 1; we apply the sparse polynomial commitment from Corollary 2 to compile the indexing polynomials $\tilde{A}$, $\tilde{B}$, $\tilde{C}$. Zaratan is secure in the generic group model for unknown-order groups (GGUO) and heuristically through Fiat-Shamir in the ROM. The cost of the final verifier is dominated by our polynomial commitment, in particular the sparse ones for indexing (the verifier does not pay this cost in the case of *uniform* computations where the R1CS has a "nice" encoding that the verifier can compute directly). The final efficiency is described in Table 1.

We stress that the prover has a linear dependency (not quasi-linear) w.r.t. $m := \log_2 \|\vec{w}\|_\infty$. This is because in our mod-PC from Section 7, we do not perform any operations over $\mathbb{Z}$ on the integer witness coefficients; the dominating operations using those coefficients are exponentiations in the group of unknown order, whose complexity is $O(m \cdot \mathsf{poly}(\lambda))$.

Acknowledgments We thank Mahak Pancholi for useful feedback on early drafts of our work and for useful discussions on compilers for idealized protocols. Thanks to Diego Aranha and Misha Volkhov for their thoughtful inputs. We thank the anonymous Eurocrypt reviewers for useful criticism and for having spotted a superlinear step in one of our subprotocols in an earlier version of our polynomial commitment construction (now removed and replaced by the approach based on reverse Kronecker substitution). This work was supported by the Estonian Research Council grant PRG2531.

[22] Otherwise, after we sampled the prime, the adversary could convince a verifier it knows non-existing integer witnesses (as in the equation $w^2 = -1$ over $\mathbb{Z}$) just by strategically providing the right oracle polynomial on the basis of the sampled prime.

[23] A formal treatment of this section is in Section H.

[24] The resulting verification time/proof size from SPARK has an additional logarithmic factor (in the number of coefficients of the polynomial) compared to the construction in Section 7.2, as shown in Table 1.This is not inherent and works following up SPARK [SL20] remove this factor; we leave as future work to bring these results to our settings.

References

AB06. S. Arora and B. Barak. Computational Complexity: A Modern Approach. Cambridge University Press, 2006. URL: https://theory.cs.princeton.edu/complexity/book.pdf.

ABC+22. Diego F. Aranha, Emil Madsen Bennedsen, Matteo Campanelli, Chaya Ganesh, Claudio Orlandi, and Akira Takahashi. ECLIPSE: Enhanced compiling method for Pedersen-committed zk-SNARK engines. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, PKC 2022: 25th International Conference on Theory and Practice of Public Key Cryptography, Part I, volume 13177 of Lecture Notes in Computer Science, pages 584–614. Springer, Cham, March 2022. doi:10.1007/978-3-030-97121-2_21.

ABST23. Miguel Ambrona, Marc Beunardeau, Anne-Laure Schmitt, and Raphaël R Toledo. aplonk: Aggregated p lon k from multi-polynomial commitment schemes. In International Workshop on Security, pages 195–213. Springer, 2023.

AGL+23. Arasu Arun, Chaya Ganesh, Satya V. Lokam, Tushar Mopuri, and Sriram Sridhar. Dew: A transparent constant-sized polynomial commitment scheme. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, PKC 2023: 26th International Conference on Theory and Practice of Public Key Cryptography, Part II, volume 13941 of Lecture Notes in Computer Science, pages 542–571. Springer, Cham, May 2023. doi:10.1007/978-3-031-31371-4_19.

BBB+18. Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In 2018 IEEE Symposium on Security and Privacy, pages 315–334. IEEE Computer Society Press, May 2018. doi:10.1109/SP.2018.00020.

BBHR18. Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Report 2018/046, 2018. URL: https://eprint.iacr.org/2018/046.

BCF+21. Daniel Benarroch, Matteo Campanelli, Dario Fiore, Kobi Gurkan, and Dimitris Kolonelos. Zero-knowledge proofs for set membership: Efficient, succinct, modular. In Nikita Borisov and Claudia Díaz, editors, FC 2021: 25th International Conference on Financial Cryptography and Data Security, Part I, volume 12674 of Lecture Notes in Computer Science, pages 393–414. Springer, Berlin, Heidelberg, March 2021. doi:10.1007/978-3-662-64322-8_19.

BCS16. Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In Martin Hirt and Adam D. Smith, editors, TCC 2016-B: 14th Theory of Cryptography Conference, Part II, volume 9986 of Lecture Notes in Computer Science, pages 31–60. Springer, Berlin, Heidelberg, October / November 2016. doi:10.1007/978-3-662-53644-5_2.

BFS19. Benedikt Bünz, Ben Fisch, and Alan Szepieniec. Transparent SNARKs from DARK compilers. Cryptology ePrint Archive, Report 2019/1229, 2019. URL: https://eprint.iacr.org/2019/1229.

BFS20. Benedikt Bünz, Ben Fisch, and Alan Szepieniec. Transparent SNARKs from DARK compilers. In Anne Canteaut and Yuval Ishai, editors, Advances in Cryptology – EUROCRYPT 2020, Part I, volume 12105 of Lecture Notes in Computer Science, pages 677–706. Springer, Cham, May 2020. doi:10.1007/978-3-030-45721-1_24.

BGTT57. Jorge Luis Borges, Margarita Guerrero, Francisco Toledo, and Francisco Toledo. Manual de zoología fantástica, volume 125. Fondo de cultura económica México, 1957.

BGV12. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, ITCS 2012: 3rd Innovations in Theoretical Computer Science, pages 309–325. Association for Computing Machinery, January 2012. doi:10.1145/2090236.2090262.

BHR+21. Alexander R. Block, Justin Holmgren, Alon Rosen, Ron D. Rothblum, and Pratik Soni. Time- and space-efficient arguments from groups of unknown order. In Tal Malkin and Chris Peikert, editors, Advances in Cryptology – CRYPTO 2021, Part IV, volume 12828 of Lecture Notes in Computer Science, pages 123–152, Virtual Event, August 2021. Springer, Cham. doi:10.1007/978-3-030-84259-8_5.

Bra12. Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In Reihaneh Safavi-Naini and Ran Canetti, editors, Advances in Cryptology – CRYPTO 2012, volume 7417 of Lecture Notes in Computer Science, pages 868–886. Springer, Berlin, Heidelberg, August 2012. doi:10.1007/978-3-642-32009-5_50.

BSCKL21. Eli Ben-Sasson, Dan Carmon, Swastik Kopparty, and David Levit. Elliptic curve fast fourier transform (ecfft) part i: Fast polynomial algorithms over all finite fields, 2021. URL: https://arxiv.org/abs/2107.08473, arXiv:2107.08473.

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, Advances in Cryptology – EUROCRYPT 2023, Part II, volume 14005 of Lecture Notes in Computer Science, pages 499–530. Springer, Cham, April 2023. doi:10.1007/978-3-031-30617-4_17.

CFF+21. Matteo Campanelli, Antonio Faonio, Dario Fiore, Anaïs Querol, and Hadrián Rodríguez. Lunar: A toolbox for more efficient universal and updatable zkSNARKs and commit-and-prove extensions. In Mehdi Tibouchi and Huaxiong Wang, editors, Advances in Cryptology – ASIACRYPT 2021, Part III, volume 13092 of Lecture Notes in Computer Science, pages 3–33. Springer, Cham, December 2021. doi:10.1007/978-3-030-92078-4_1.

CFG25. Matteo Campanelli, Dario Fiore, and Rosario Gennaro. Natively compatible super-efficient lookup arguments and how to apply them. Journal of Cryptology, 38(1):14, 2025.

CFH+22. Matteo Campanelli, Dario Fiore, Semin Han, Jihye Kim, Dimitris Kolonelos, and Hyunok Oh. Succinct zero-knowledge batch proofs for set accumulators. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, ACM CCS 2022: 29th Conference on Computer and Communications Security, pages 455–469. ACM Press, November 2022. doi:10.1145/3548606.3560677.

CFK24. Matteo Campanelli, Dario Fiore, and Hamidreza Khoshakhlagh. Witness encryption for succinct functional commitments and applications. In Qiang Tang and Vanessa Teague, editors, PKC 2024: 27th International Conference on Theory and Practice of Public Key Cryptography, Part II, volume 14602 of Lecture Notes in Computer Science, pages 132–167. Springer, Cham, April 2024. doi:10.1007/978-3-031-57722-2_5.

CFQ19. Matteo Campanelli, Dario Fiore, and Anaïs Querol. LegoSNARK: Modular design and composition of succinct zero-knowledge proofs. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, ACM CCS 2019: 26th Conference on Computer and Communications Security, pages 2075–2092. ACM Press, November 2019. doi:10.1145/3319535.3339820.

CFT22. Dario Catalano, Dario Fiore, and Ida Tucker. Additive-homomorphic functional commitments and applications to homomorphic signatures. In Shweta Agrawal and Dongdai Lin, editors, Advances in Cryptology – ASIACRYPT 2022, Part IV, volume 13794 of Lecture Notes in Computer Science, pages 159–188. Springer, Cham, December 2022. doi:10.1007/978-3-031-22972-5_6.

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, Progress in Cryptology - LATINCRYPT 2023: 8th International Conference on Cryptology and Information Security in Latin America, volume 14168 of Lecture Notes in Computer Science, pages 331–351, Quito, Ecuador, October 2023. Springer, Cham. doi:10.1007/978-3-031-44469-2_17.

CHA26. Matteo Campanelli and Mathias Hall-Andersen. General techniques for building snarks over the integers. In Public-Key Cryptography–PKC 2026: 29th IACR International Conference on Practice and Theory of Public-Key Cryptography, West Palm Beach, FL, USA, May 25–28, 2026, Proceedings, Part III, page 197. Springer Nature, 2026.

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, Advances in Cryptology – EUROCRYPT 2020, Part I, volume 12105 of Lecture Notes in Computer Science, pages 738–768. Springer, Cham, May 2020. doi:10.1007/978-3-030-45721-1_26.

CL02. Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In Moti Yung, editor, Advances in Cryptology – CRYPTO 2002, volume 2442 of Lecture Notes in Computer Science, pages 61–76. Springer, Berlin, Heidelberg, August 2002. doi:10.1007/3-540-45708-9_5.

COS20. Alessandro Chiesa, Dev Ojha, and Nicholas Spooner. Fractal: Post-quantum and transparent recursive proofs from holography. In Anne Canteaut and Yuval Ishai, editors, Advances in Cryptology – EUROCRYPT 2020, Part I, volume 12105 of Lecture Notes in Computer Science, pages 769–793. Springer, Cham, May 2020. doi:10.1007/978-3-030-45721-1_27.

DF02. Ivan Damgård and Eiichiro Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In Yuliang Zheng, editor, Advances in Cryptology – ASIACRYPT 2002, volume 2501 of Lecture Notes in Computer Science, pages 125–142. Springer, Berlin, Heidelberg, December 2002. doi:10.1007/3-540-36178-2_8.

DK02. Ivan Damgård and Maciej Koprowski. Generic lower bounds for root extraction and signature schemes in general groups. In Lars R. Knudsen, editor, Advances in Cryptology – EUROCRYPT 2002, volume 2332 of Lecture Notes in Computer Science, pages 256–271. Springer, Berlin, Heidelberg, April / May 2002. doi:10.1007/3-540-46035-7_17.

FO97. Eiichiro Fujisaki and Tatsuaki Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In Burton S. Kaliski, Jr., editor, Advances in Cryptology – CRYPTO'97, volume 1294 of Lecture Notes in Computer Science, pages 16–30. Springer, Berlin, Heidelberg, August 1997. doi:10.1007/BFb0052225.

FV12. Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012. URL: https://eprint.iacr.org/2012/144.

GKR08. Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. In Richard E. Ladner and Cynthia Dwork, editors, 40th Annual

ACM Symposium on Theory of Computing, pages 113–122. ACM Press, May 2008. doi:10.1145/1374376.1374396.

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, Advances in Cryptology – CRYPTO 2023, Part II, volume 14082 of Lecture Notes in Computer Science, pages 193–226. Springer, Cham, August 2023. doi:10.1007/978-3-031-38545-2_7.

GMN22. Nicolas Gailly, Mary Maller, and Anca Nitulescu. SnarkPack: Practical SNARK aggregation. In Ittay Eyal and Juan A. Garay, editors, FC 2022: 26th International Conference on Financial Cryptography and Data Security, volume 13411 of Lecture Notes in Computer Science, pages 203–229. Springer, Cham, May 2022. doi:10.1007/978-3-031-18283-9_10.

GNS23. Chaya Ganesh, Anca Nitulescu, and Eduardo Soria-Vazquez. Rinocchio: SNARKs for ring arithmetic. Journal of Cryptology, 36(4):41, October 2023. doi:10.1007/s00145-023-09481-3.

GPPS24. Chaya Ganesh, Sikhar Patranabis, Shubh Prakash, and Nitin Singh. GAPP: Generic aggregation of polynomial protocols. Cryptology ePrint Archive, Report 2024/1685, 2024. URL: https://eprint.iacr.org/2024/1685.

GPR21. Lior Goldberg, Shahar Papini, and Michael Riabzev. Cairo – a Turing-complete STARK-friendly CPU architecture. Cryptology ePrint Archive, Report 2021/1063, 2021. URL: https://eprint.iacr.org/2021/1063.

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. URL: https://eprint.iacr.org/2019/953.

GWHD25. Albert Garreta, Hendrik Waldner, Katerina Hristova, and Luca Dall'Ava. Zinc: Succinct arguments with small arithmetization overheads from IOPs of proximity to the integers. Cryptology ePrint Archive, Report 2025/316, 2025. URL: https://eprint.iacr.org/2025/316.

GWV+25. Albert Garreta, Hendrik Waldner, Ilia Vlasov, Katerina Hristova, Luca Dall'Ava, Marko Cupic, and Matthew Klein. Zinc: Succinct arguments with small arithmetization overheads from IOPs of proximity to the integers. In Yael Tauman Kalai and Seny F. Kamara, editors, Advances in Cryptology – CRYPTO 2025, Part VII, volume 16006 of Lecture Notes in Computer Science, pages 259–291. Springer, Cham, August 2025. doi:10.1007/978-3-032-01907-3_9.

Kil92. Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In 24th Annual ACM Symposium on Theory of Computing, pages 723–732. ACM Press, May 1992. doi:10.1145/129712.129782.

KPS18. Ahmed E. Kosba, Charalampos Papamanthou, and Elaine Shi. xJsnark: A framework for efficient verifiable computation. In 2018 IEEE Symposium on Security and Privacy, pages 944–961. IEEE Computer Society Press, May 2018. doi:10.1109/SP.2018.00018.

KZG10. Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In Masayuki Abe, editor, Advances in Cryptology – ASIACRYPT 2010, volume 6477 of Lecture Notes in Computer Science, pages 177–194. Springer, Berlin, Heidelberg, December 2010. doi:10.1007/978-3-642-17373-8_11.

Lab. Protocol Labs. Snarkpack v2: A new version of filecoin's proof aggregator. https://filecoin.io/blog/posts/snarkpack-v2-a-new-version-of-filecoin-s-proof-aggregator/.

Lay. LayerEdge. What is layeredge? https://docs.layeredge.io/docs/introduction-to-layerEdge/what-is-layerEdge.

LFKN90. Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. In 31st Annual Symposium on Foundations of Computer Science, pages 2–10. IEEE Computer Society Press, October 1990. doi:10.1109/FSCS.1990.89518.

LM19. Russell W. F. Lai and Giulio Malavolta. Subvector commitments with application to succinct arguments. In Alexandra Boldyreva and Daniele Micciancio, editors, Advances in Cryptology – CRYPTO 2019, Part I, volume 11692 of Lecture Notes in Computer Science, pages 530–560. Springer, Cham, August 2019. doi:10.1007/978-3-030-26948-7_19.

LS24. Hongyin Luo and Wei Sun. Addition is all you need for energy-efficient language models, 2024. URL: https://arxiv.org/abs/2410.00907, arXiv:2410.00907.

OKMZ24. Michele Orrù, George Kadianakis, Mary Maller, and Greg Zaverucha. Beyond the circuit: How to minimize foreign arithmetic in ZKP circuits. Cryptology ePrint Archive, Report 2024/265, 2024. URL: https://eprint.iacr.org/2024/265.

PT18. Paul Pollack and Enrique Treviño. Finding the four squares in lagrange's theorem. Integers, 18(A15):7–17, 2018.

Ros41. Barkley Rosser. Explicit bounds for some functions of prime numbers. American Journal of Mathematics, 63(1):211–232, 1941. URL: http://www.jstor.org/stable/2371291.

RSA78. Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 21(2):120–126, 1978.

SB23. István András Seres and Péter Burcsi. Behemoth: transparent polynomial commitment scheme with constant opening proof size and verifier time. Cryptology ePrint Archive, Report 2023/670, 2023. URL: https://eprint.iacr.org/2023/670.

Set20. Srinath Setty. Spartan: Efficient and general-purpose zkSNARKs without trusted setup. In Daniele Micciancio and Thomas Ristenpart, editors, Advances in Cryptology – CRYPTO 2020, Part III, volume 12172 of Lecture Notes in Computer Science, pages 704–737. Springer, Cham, August 2020. doi:10.1007/978-3-030-56877-1_25.

SL20. Srinath Setty and Jonathan Lee. Quarks: Quadruple-efficient transparent zksnarks. Cryptology ePrint Archive, 2020.

Sor22. Eduardo Soria-Vazquez. Doubly efficient interactive proofs over infinite and non-commutative rings. In Eike Kiltz and Vinod Vaikuntanathan, editors, TCC 2022: 20th Theory of Cryptography Conference, Part I, volume 13747 of Lecture Notes in Computer Science, pages 497–525. Springer, Cham, November 2022. doi:10.1007/978-3-031-22318-1_18.

STW24. Srinath T. V. Setty, Justin Thaler, and Riad S. Wahby. Unlocking the lookup singularity with Lasso. In Marc Joye and Gregor Leander, editors, Advances in Cryptology – EUROCRYPT 2024, Part VI, volume 14656 of Lecture Notes in Computer Science, pages 180–209. Springer, Cham, May 2024. doi:10.1007/978-3-031-58751-1_7.

SY. Kaz Sato and Cliff Young. An in-depth look at google's first tensor processing unit. https://cloud.google.com/blog/products/ai-machine-learning/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu.

Tha23. Justin Thaler. Proofs, Arguments, and Zero-Knowledge, 2023.

TV20. Patrick Towa and Damien Vergnaud. Succinct diophantine-satisfiability arguments. In Shiho Moriai and Huaxiong Wang, editors, Advances in Cryptology – ASIACRYPT 2020, Part III, volume 12493 of Lecture Notes in Computer Science, pages 774–804. Springer, Cham, December 2020. doi:10.1007/978-3-030-64840-4_26.

Wes19. Benjamin Wesolowski. Efficient verifiable delay functions. In Yuval Ishai and Vincent Rijmen, editors, Advances in Cryptology – EUROCRYPT 2019, Part III, volume 11478 of Lecture Notes in Computer Science, pages 379–407. Springer, Cham, May 2019. doi:10.1007/978-3-030-17659-4_13.

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. doi:10.1109/SP.2018.00060.

XZS22. Tiancheng Xie, Yupeng Zhang, and Dawn Song. Orion: Zero knowledge proof with linear prover time. In Yevgeniy Dodis and Thomas Shrimpton, editors, Advances in Cryptology – CRYPTO 2022, Part IV, volume 13510 of Lecture Notes in Computer Science, pages 299–328. Springer, Cham, August 2022. doi:10.1007/978-3-031-15985-5_11.

YZRM24. Xiao Yang, Chengru Zhang, Mark Ryan, and Gao Meng. Multivariate multi-polynomial commitment and its applications. Cryptology ePrint Archive, Report 2024/827, 2024. URL: https://eprint.iacr.org/2024/827.

ZGK+17. Yupeng Zhang, Daniel Genkin, Jonathan Katz, Dimitrios Papadopoulos, and Charalampos Papamanthou. vSQL: Verifying arbitrary SQL queries over dynamic outsourced databases. In 2017 IEEE Symposium on Security and Privacy, pages 863–880. IEEE Computer Society Press, May 2017. doi:10.1109/SP.2017.43.

ZGK+18. Yupeng Zhang, Daniel Genkin, Jonathan Katz, Dimitrios Papadopoulos, and Charalampos Papamanthou. vRAM: Faster verifiable RAM with program-independent preprocessing. In 2018 IEEE Symposium on Security and Privacy, pages 908–925. IEEE Computer Society Press, May 2018. doi:10.1109/SP.2018.00013.

Supplementary Material

A Further Preliminaries

A.1 The Sumcheck Protocol

Let $p(x_1 \ldots, x_n)$ be a multivariate polynomial in n variables defined over a field $\mathbb{F}$. Consider the value $a = \sum_{i \in \{0,1\}^n} p(i)$, i.e., the sum of the value of p on all the vertices of the Boolean hypercube. This computation takes $N = 2^n$ time and the sumcheck protocol [LFKN90] described in Figure 4, is a way for a Prover to convince a Verifier that a is correct in $O(n)$ time, plus a single query to the polynomial p on a random point in $\mathbb{F}^n$.

P sends the polynomial $p_1(x) = \sum_{i \in \{0,1\}^{n-1}} p(x, i)$.
V checks that $p_1(0) + p_1(1) = a$ and sends back $r_1 \in_R \mathbb{F}$.
P sends the polynomial $p_2(x) = \sum_{i \in \{0,1\}^{n-2}} p(r_1, x, i)$.
V checks that $p_2(0) + p_2(1) = p_1(r_1)$ and sends back $r_2 \in_R \mathbb{F}$.
[...]
At round j P sends the polynomial $p_j(x) = \sum_{i \in \{0,1\}^{n-j}} p(r_1, \ldots, r_{j-1}, x, i)$.
V checks that $p_j(0) + p_j(1) = p_{j-1}(r_{j-1})$ and sends back $r_j \in_R \mathbb{F}$.
[...]
At the last round P sends the polynomial $p_{n-1}(x) = p(r_1, \ldots, r_{n-1}, x)$.
V checks that $p_{n-1}(0) + p_{n-1}(1) = p_{n-2}(r_{n-2})$, selects $r_n \in_R \mathbb{F}$ and checks that $p_{n-1}(r_n) = p(r_1, \ldots, r_n)$ via a single query to p.

Fig. 4. The Sumcheck Protocol

B Generic Group of Unknown Order (GGUO) Model

A generic group of unknown order $\mathbb{G}_{\hat{N}}$ is an idealized model where the operations are performed via oracles, without access to the actual representation of elements to the parties. The model consists of two oracles: one that samples random group elements and another that performs the group operation.

Definition 17 (Generic Group of Unknown Order (GGUO) Model [DK02]). *The parties have access to two oracles, one which produces random group elements and the other which computes the group operation. Initially* $n = 0$.

$\mathcal{O}_1()$:
- $n \leftarrow n + 1$
- *Sample* $x_n \leftarrow\!\!\$\ \mathbb{Z}_N$
- *Output* $\sigma(x_n)$

$\mathcal{O}_2(i, j, b)$: *takes two indexes and a sign bit:*
- $n \leftarrow n + 1$
- *Define* $x_n \leftarrow x_i + (-1)^b \cdot x_j$
- *Output* $\sigma(x_n)$

C Wesołowski's Proof-of-Exponentiation

The subprotocol Σ_{PoE} is used in the $\Sigma_{\mathsf{MultiEval}}$ to reduce the computation of the verifier: in a naive protocol the verifier would have to compute $\mathbf{B} = [\mathsf{q}^{2^t}] \cdot \mathbf{A} \in \mathbb{G}_{\hat{N}}^{\lambda}$ where q^{2^t} is linear in the number of coefficients of the original polynomial f. This would prevent the verifier from being succinct, the solution is to outsource the computation to the prover and use Σ_{PoE} (see Fig. 5) to verify the correctness of the computation. Note that the only dependency between the verifier's running time in Σ_{PoE} and the exponent Δ is the time required to compute Δ mod p which for $\Delta = \mathsf{q}^{2^t}$ can be done in polylogarithmic time. The protocol relies on the adaptive root assumption (Definition 18) in the group $\mathbb{G}_{\hat{N}}$.

Definition 18 (Adaptive Root Assumption [Wes19]). *Let $(\mathcal{A}_0, \mathcal{A}_1)$ be arbitrary PPT algorithms. The adaptive root assumption states:*

$$\mathsf{negl}(\lambda) \geq \Pr\left[\, [p] \cdot R = H \neq 0 \in \mathbb{G}_{\hat{N}} \;\middle|\; \begin{array}{c} \mathbb{G}_{\hat{N}} \leftarrow \mathcal{G}(1^\lambda) \\ (H, \mathsf{st}) \leftarrow \mathcal{A}_0(\mathbb{G}_{\hat{N}}) \\ p \leftarrow\!\$\ \mathbb{P}_\lambda \\ R \leftarrow \mathcal{A}_1(p, H, \mathsf{st}) \end{array} \right]$$

For some negligible function $\mathsf{negl}(\lambda)$.

$\Sigma_{\mathsf{PoE}}(\mathbf{A} \in \mathbb{G}_{\hat{N}}^{\mathsf{d}}, \mathbf{B} \in \mathbb{G}_{\hat{N}}^{\mathsf{d}}, \Delta)$

V samples $p \leftarrow\!\$\ \mathbb{P}_\lambda$ and sends p to P

P computes:

$\Phi_{(\Delta)} \leftarrow \lfloor \Delta / p \rfloor \,; \mathbf{Q} \leftarrow [\Phi_{(\Delta)}] \cdot \mathbf{A} \in \mathbb{G}_{\hat{N}}^{\mathsf{d}}$

P sends $(\Psi_{(\Delta)}, \mathbf{Q})$ to V

V checks: $[p] \cdot \mathbf{Q} + [\Delta \bmod p] \cdot \mathbf{A} \stackrel{?}{=} \mathbf{B}$

Fig. 5. The Proof-of-Exponentiation Protocol [Wes19] by Wesolowski.

Lemma 4 (Soundness of Σ_{PoE} [Wes19]). *Assuming the Adaptive Root Assumption holds in $\mathbb{G}_{\hat{N}}$, Σ_{PoE} (Fig. 5) is an argument system for relation $\mathcal{R}_{\Sigma_{\mathsf{PoE}}}$:*

$$\mathcal{R}_{\Sigma_{\mathsf{PoE}}} = \left\{ ((\mathbf{A}, \mathbf{B}, \Delta), \perp) : \mathbf{B} = [\Delta] \cdot \mathbf{A} \in \mathbb{G}_{\hat{N}}^{\lambda} \right\}$$

With negligible soundness error.

Observe that the relation is in P; membership can be verified in $\log(\Delta)$ time using a double-and-add algorithm.

D Our Core mod-PC Construction

In this section we propose our construction for mod-PC. Our starting point are the techniques in [BHR+21]. We start from a brief overview of the protocols $\Sigma_{\mathsf{Eval}}/\Sigma_{\mathsf{MultiEval}}$ (Fig. 7) in their work. We include the protocols for completeness and to unify the notation with the rest of the paper. Besides replacing the Σ_{PoE} used in Block et al. with the Wesolowski Σ_{PoE} (see Section C), the section contains no original contributions. We refer the reader to the original paper for further details. The use of the Wesolowski Σ_{PoE} does not affect extraction as the PoE relation $\mathcal{R}_{\Sigma_{\mathsf{PoE}}}$ has no witness.

Notation. Define the multi-linear evaluation:

$$\mathsf{ML}(f, \vec{x}) = \sum_{\vec{t} \in \{0,1\}^\mu} f_{\vec{t}} \cdot \left(\prod_{i=1}^{\mu} x_i^{t_i} \right) \mod q$$

Note that these protocols use λ-dimensional integer commitments $\mathbf{C} = (C_1, \ldots, C_\lambda) \in \mathbb{G}_{\hat{N}}^{\lambda}$, we denote these in bold. Addition on these commitments is done component-wise and scalar multiplication is done component-wise as well:

$$\mathbf{A} + [s] \cdot \mathbf{B} = (A_1 + [s] \cdot B_1, \ldots, A_\lambda + [s] \cdot B_\lambda) \in \mathbb{G}_{\hat{N}}^{\lambda}$$

Generalizing, denote by $M \star \mathbf{C}$ the commitment resulting from applying the matrix $M \in \mathbb{Z}^{\lambda \times \lambda}$ to the commitment $\mathbf{C} \in \mathbb{G}_{\hat{N}}^{\lambda}$ in the natural way.

$\mathsf{isValid}(\mathsf{pp} = (\mathbb{G}_{\hat{N}}, G), C \in \mathbb{G}_{\hat{N}}, \vec{x} \in \mathbb{Z}^{\mu}, q \in \mathbb{P}, y \in \mathbb{Z}, f \in \mathbb{Z}^{2^{\mu}})$

if $\|f\|_{\infty} > \mathsf{B}$ **then return** 0
if $C \neq [\mathsf{Enc}_{\mathsf{q}}(f)] \cdot G$ **then return** 0
$y \neq \mathsf{ML}(f, \vec{x}) \bmod q$ **then return** 0
return 1

Fig. 6. Binding notion for the multi-linear commitment scheme.

$\Sigma_{\mathsf{MultiEval}}(\mathsf{pp}, \mathbf{C}, \mathsf{r}, \vec{x}, \vec{y}, m, q; Z \in \mathbb{Z}^{\lambda \times 2^{\mu - \mathsf{r}+1}})$

output:accept or reject
if $\mathsf{r} = \mu$:
P sends $Z \in \mathbb{Z}^{\lambda}$ to V
V Checks:
$\|Z\|_{\infty} \leq 2^m \cdot (2\lambda)^{\mu}$
$\vec{y} \stackrel{?}{\equiv} Z \bmod q$
$\mathbf{C} \stackrel{?}{=} ([Z] \cdot G, \ldots, [Z] \cdot G)$
else
// Prover computes the evaluation over each subcube over the field:
// One for $x_{\mathsf{r}} = 0$ and one for $x_{\mathsf{r}} = 1$
P Computes:

$$\vec{y}_L \leftarrow \sum_{\vec{t} \in \{0,1\}^{\mu - \mathsf{r} - 1}} Z_{(*, 0 \| \vec{t})} \cdot \prod_{j=1}^{\mu - \mathsf{r} - 1} \chi(t_j, x_{j+\mathsf{r}+1}) \mod q$$

$$\vec{y}_R \leftarrow \sum_{\vec{t} \in \{0,1\}^{\mu - \mathsf{r} - 1}} Z_{(*, 1 \| \vec{t})} \cdot \prod_{j=1}^{\mu - \mathsf{r} - 1} \chi(t_j, x_{j+\mathsf{r}+1}) \mod q$$

// Prover commits to the "split polynomials"
P Computes:

$$\mathbf{C}_L \leftarrow [l] \cdot G \text{ where } l \leftarrow \sum_{\vec{t} \in \{0,1\}^{\mu - \mathsf{r} - 1}} \mathsf{q}^{\vec{t}} \cdot Z_{(*, 0 \| \vec{t})}$$

$$\mathbf{C}_R \leftarrow [r] \cdot G \text{ where } r \leftarrow \sum_{\vec{t} \in \{0,1\}^{\mu - \mathsf{r} - 1}} \mathsf{q}^{\vec{t}} \cdot Z_{(*, 1 \| \vec{t})}$$

P sends $(\vec{y}_L, \vec{y}_R)$ and $(\mathbf{C}_L, \mathbf{C}_R)$ to V
// Verifier checks the decomposition using the Σ_{PoE} protocol.
V : Check $\vec{y} \stackrel{?}{=} \vec{y}_L \cdot (1 - x_{\mathsf{r}+1}) + \vec{y}_R \cdot x_{\mathsf{r}+1}$
$\Sigma_{\mathsf{PoE}}(\mathbf{C}_R, \mathbf{C} - \mathbf{C}_L, \mathsf{q}^{\mu - \mathsf{r} - 1})$
// Verifier samples random binary matrices U_L, U_R and sends them to the prover.
V : $U = [U_L \| U_R] \leftarrow\!\!\$\ \{0,1\}^{\lambda \times 2\lambda}$ where $U_L, U_R \in \{0,1\}^{\lambda \times \lambda}$
V sends U to P
P and V :
$\vec{y}' \leftarrow U_L \cdot \vec{y}_L + U_R \cdot \vec{y}_R$
$\mathbf{C}' \leftarrow (U_L \star \mathbf{C}_L) + (U_R \star \mathbf{C}_R)$
P : For $Z_L, Z_R \in \mathbb{Z}^{\lambda \times 2^{\mu - \mathsf{r} - 1}}$ such that $Z = [Z_L \| Z_R]$
$Z' \leftarrow U_L \cdot Z_L + U_R \cdot Z_R$
return $\Sigma_{\mathsf{MultiEval}}(\mathbf{C}', \mathsf{r}+1, x, \vec{y}', m, q, ; Z')$

Fig. 7. MultiEval protocol by Block, Holmgren, Rosen, Rothblum and Soni [BHR+21], included here for completeness and to unify notation.

$$\frac{\Sigma_{\mathsf{Eval}}(\mathsf{pp}, C \in \mathbb{G}_{\hat{N}}, \vec{x} \in \mathbb{F}_q^{\mu}, y \in \mathbb{F}_q, m, q; f \in \mathbb{Z}^{2^{\mu}})}{\begin{array}{l}\vec{y} = (y, \ldots, y) \in \mathbb{F}_q^{\lambda} \\ \mathbf{C} = (C, \ldots, C) \in \mathbb{G}^{\lambda} \\ Z = (f, \ldots, f) \in \mathbb{Z}^{\lambda \times 2^{\mu}} \\ \Sigma_{\mathsf{MultiEval}}(\mathbf{C}, \mathsf{r} = 1, \vec{x}, \vec{y}, m, q; Z)\end{array}}$$

Fig. 8. Evaluation protocol by Block, Holmgren, Rosen, Rothblum and Soni [BHR+21], invoking the $\Sigma_{\mathsf{MultiEval}}$ protocol on λ parallel instances.

D.1 Evaluation Protocol in [BHR+21]

The binding notion (unlike that of the original DARK paper) is defined for an integer polynomial with coefficients in the range $[-\mathsf{B}, \mathsf{B}] \subseteq \mathbb{Z}$, this is captured by a procedure $\mathsf{isValid}$ which defines a valid opening (see Fig. 6).

Theorem 5 (Knowledge Soundness of Σ_{Eval}). *The protocol Σ_{Eval} is an argument-of-knowledge protocol for the relation:*

$$\mathcal{R}_{\Sigma_{\mathsf{MultiEval}}} = \{((C, \vec{x}, y), f) : \mathsf{isValid}(\mathsf{pp}, C, \vec{x}, f) \wedge y = \mathsf{ML}(f, \vec{x}) \mod q\}$$

Proof. See the original paper [BHR+21].

D.2 Our mod-PC construction

Below is our final construction obtained through the approach in [BHR+21].

Definition 19. *Define the* $\mathsf{modPC} = (\mathsf{Setup}, \mathsf{Com}, \mathsf{Prv}, \mathsf{Vfy})$ *as in Fig. 9. Then* modPC *is a secure mod-PC.*

Proof. Follows from Theorem 5.

$$\frac{\mathsf{Prv}(\mathsf{pp}, q, c, \mathsf{opn}, \vec{x})}{\begin{array}{l}y \leftarrow f(\vec{x}) \bmod q \\ \text{Run } \Sigma_{\mathsf{Eval}}(\mathsf{pp}, c, \vec{x} \bmod q, m, \\ \qquad q; \mathsf{Coeff}(f))\end{array}} \qquad \frac{\mathsf{Setup}(1^{\lambda}) \to \mathsf{pp}}{\begin{array}{l}\mathbb{G}_{\hat{N}} \leftarrow \mathcal{G}(1^{\lambda}) \\ G \leftarrow\!\$\ \mathbb{G}_{\hat{N}} \\ \textbf{return } (\mathbb{G}_{\hat{N}}, G)\end{array}}$$

$$\frac{\mathsf{Vfy}(\mathsf{pp}, q, c, \mathsf{opn}, \vec{x}, y)}{\text{Run } \Sigma_{\mathsf{Eval}}(\mathsf{pp}, c, \vec{x} \bmod q, m, q)} \qquad \frac{\mathsf{Com}(\mathsf{pp}, f) \to c}{\begin{array}{l}\mathsf{opn} \leftarrow f \\ c \leftarrow [\mathsf{Enc}_{\mathsf{q}}(f)] \cdot G \\ \textbf{return } (c, \mathsf{opn})\end{array}}$$

Fig. 9. Our mod-PC revisiting the techniques in Block et al. [BHR+21]. Here $\mathsf{Coeff}(f)$ denotes the coefficients of the polynomial f: a vector with indices in $\mathcal{H}_{\mu}$ and values in $\mathbb{Z}$.

Theorem 6 (Complexity of Σ_{Eval}). *Let $m^* = m + \log_2(\lambda) \cdot \mu$. The complexity of the* modPC *(see Fig. 9) is:*

$$\begin{aligned}\mathsf{C}_{com}(m, \lambda, \mu) &= O\left(2^{\mu} \cdot m\right) \\ \mathsf{C}_{prv}(m, \lambda, \mu) &= O\left(2^{\mu} \cdot \lambda^2 \cdot m^*\right) \\ \mathsf{C}_{vfy}(m, \lambda, \mu) &= O\left(\mu \cdot \lambda^2 + \lambda \cdot m^*\right)\end{aligned}$$

Measured in group operations (addition/subtraction/negation) in $\mathbb{G}_{\hat{N}}$.

Proof. The complexity arises from the following operations:

Com : The prover performs a single multi-scalar multiplication (MSM) of 2^μ coefficients with bit-size m, requiring $O(2^\mu \cdot m)$ group operations. The verifier simply receives one group element.

Prv : Each round of folding reduces μ by one and increases the norm of the coefficients by at most λ; each new coefficient is a $\{0,1\}$-combination of λ coefficients. This means $m' = m + \log_2(\lambda)$ and $\mu' = \mu - 1$ in the next round. Computing $U_L \star \mathbf{C}_L + U_R \star \mathbf{C}_R$ takes $O(\lambda^2)$ group operations per round. Committing to the cross terms l, r and running Σ_{PoE} requires $O(\lambda \cdot 2^\mu \cdot m)$ group operations per round. If we bound the bitsize of the coefficients in any rounds by $m^* = m + \log_2(\lambda) \cdot \mu$, then we obtain claimed complexity.

Vfy : In each round, the verifier sends $\{0,1\}$-matrices U_L, U_R of size $\lambda \times \lambda$ (requiring $O(\lambda^2)$ time), computes a vector-matrix multiplication $U_L \cdot \vec{y}_L + U_R \cdot \vec{y}_R$ ($O(\lambda^2)$ group operations), runs Σ_{PoE} ($O(\lambda^2)$ group operations), and processes each commitment coordinate ($O(\lambda)$ group operations). The final prover message has bit-size at most $m^* = m + \log_2(\lambda) \cdot \mu$ and dimension λ, requiring $O(\lambda \cdot m^*)$ group operations. This gives the claimed complexity.

Observe that the verifier complexity $\mathsf{C}_{\mathrm{vfy}}$ is *linear* in m, the bitsize of the largest coefficient. This implies that the protocol is not succinct when the witness has large norm. We tackle this issue in the next section with a generic transformation.

E Reverse Kronecker Substitution

The following states that the complexity of a mod-PC in the norm of the coefficients can be reduced to logarithmic at the cost of increasing the number of variables.

Theorem 7 (Reverse Kronecker Substitution). *Let $\hat{m} \in \mathbb{N}_{>0}$ be an arbitrary constant. Given a $\mathsf{modPC} = (\mathsf{Setup}, \mathsf{Com}, \mathsf{Prv}, \mathsf{Vfy})$ with complexity $\mathsf{C}_{com}(m,\lambda,\mu)$, $\mathsf{C}_{prv}(m,\lambda,\mu)$, $\mathsf{C}_{vfy}(m,\lambda,\mu)$ for the operations $\mathsf{Com}, \mathsf{Prv}, \mathsf{Vfy}$ respectively. Then there exists a $\mathsf{modPC}^* = (\mathsf{Setup}^*, \mathsf{Com}^*, \mathsf{Prv}^*, \mathsf{Vfy}^*)$ with complexity:*

$$\begin{aligned}
\mathsf{C}^*_{com}(m,\lambda,\mu) &= \mathsf{C}_{com}(\hat{m},\lambda,\mu+\hat{\mu}) + c \cdot m \\
\mathsf{C}^*_{prv}(m,\lambda,\mu) &= \mathsf{C}_{prv}(\hat{m},\lambda,\mu+\hat{\mu}) + c \cdot \hat{\mu} \cdot \log_2(m) \\
\mathsf{C}^*_{vfy}(m,\lambda,\mu) &= \mathsf{C}_{vfy}(\hat{m},\lambda,\mu+\hat{\mu}) + c \cdot \hat{\mu} \cdot \log_2(m)
\end{aligned}$$

For some constant c, where $\hat{\mu} = \lfloor \log_2(m)/\hat{m} \rfloor$ is logarithmic in the number of bits m of the largest coefficient. Assuming field operations in $\mathbb{F}_q$ are unit cost ($q < 2^\lambda$).

Proof. The proof is by construction and the new protocol is defined in Fig. 10. Note that $v^{2^i} \bmod q$ can be computed in $O(i)$ operations in $\mathbb{F}_q$ by repeated squaring $v^{2^i} = (v^{2^{i-1}})^2 \bmod q$, from which we see that computing $\vec{z} = (2^{\hat{m} \cdot 2^i} \bmod q)_{i \in [\hat{\mu}]}$ in the way done in Prv^* and Vfy^* requires $O(\hat{m} \cdot \hat{\mu})$ operations in $\mathbb{F}_q$.

- Com^* consists of: decomposing coefficients into limbs and committing to the new polynomial. This takes $c \cdot m$ and $\mathsf{C}_{\mathrm{com}}(\hat{m},\lambda,\mu+\hat{\mu})$ time respectively.
- Prv^* consists of: computing $\vec{z}$, running Prv on the new polynomial and $\vec{z}$. This takes $c \cdot \hat{m} \cdot \hat{\mu}$ and $\mathsf{C}_{\mathrm{prv}}(\hat{m},\lambda,\mu+\hat{\mu})$ time respectively.
- Vfy^* consists of: computing $\vec{z}$, running Vfy on the new polynomial and $\vec{z}$. This takes $c \cdot \hat{m} \cdot \hat{\mu}$ and $\mathsf{C}_{\mathrm{vfy}}(\hat{m},\lambda,\mu+\hat{\mu})$ time respectively.

Corollary 3 (Complexity of mod-PC). *For an arbitrary constant $\hat{m} \in \mathbb{N}_{>0}$, let $\hat{\mu} = \lfloor \log_2(m)/\hat{m} \rfloor$. There exists a* modPC *with complexity:*

$$\begin{aligned}
\mathsf{C}_{com}(m, \lambda, \mu) &= O\left(2^{\mu+\hat{\mu}} \cdot \hat{m}\right) \\
&= O\left(2^{\mu+\log(m)} \cdot 1\right) \\
&= O\left(2^{\mu} \cdot m\right) \\
\mathsf{C}_{prv}(m, \lambda, \mu) &= O\left(2^{\mu+\hat{\mu}} \cdot \lambda^2 \cdot (\hat{m} + \log(\lambda) \cdot (\mu + \hat{\mu})\right) \\
&= O\left(2^{\mu} \cdot 2^{\log(m)} \cdot \lambda^2 \cdot \log(\lambda) \cdot (\mu + \log(m))\right) \\
&= O\left(2^{\mu} \cdot m \cdot \lambda^2 \cdot \log(\lambda) \cdot (\mu + \log(m))\right) \\
\mathsf{C}_{vfy}(m, \lambda, \mu) &= O\left((\mu + \hat{\mu}) \cdot \lambda^2 + \lambda \cdot (\hat{m} + \log(\lambda) \cdot (\mu + \hat{\mu}))\right) \\
&= O\left((\log(m) + \mu) \cdot (\log(\lambda) \cdot \log(m) + \lambda) \cdot \lambda\right)
\end{aligned}$$

Where complexity is measured as the number of group operations (additions / subtractions / negations) in $\mathbb{G}_{\hat{N}}$. In particular, the complexity of the verifier (C_{vfy}) is poly-logarithmic in the description of the polynomial $f(\vec{X})$: both the number of the coefficients 2^{μ} and their bitlength m. Meanwhile, the complexity of the prover ($\mathsf{C}_{com} + \mathsf{C}_{prv}$) is linear in the number of coefficients 2^{μ} and quasi-linear in the bitlength of the largest coefficient m.

Proof. Follows from composing Theorem 7 with the protocol of Fig. 7 by Block et al. [BHR+21] (see Theorem 6).

$\underline{\mathsf{Setup}^*(1^\lambda) \to \mathsf{pp}}$

return $\mathsf{Setup}(1^\lambda)$

$\underline{\mathsf{Com}^*(\mathsf{pp}, f) \to (c, \mathsf{opn})}$

// Split every coefficient into $2^{\hat{\mu}}$ $\hat{m}$-bit limbs

$\hat{\mu} \leftarrow \lfloor \log_2(m)/\hat{m} \rfloor$

for $\vec{i} \in \mathcal{H}_\mu$:

$(g_{\vec{i},0}, \ldots, g_{\vec{i},(2^{\hat{\mu}}-1)}) \leftarrow \mathsf{Limbs}(f_{\vec{i}}, \hat{m}, \hat{\mu})$

// Commit to the coefficients

$$g(\vec{X}, \vec{Z}) \leftarrow \sum_{\vec{i} \in \mathcal{H}_\mu, \vec{j} \in \mathcal{H}_{\hat{\mu}}} g_{\vec{i},\vec{j}} \cdot \vec{Z}^{\vec{j}} \cdot \vec{X}^{\vec{i}}$$

$(c, \mathsf{opn}) \leftarrow \mathsf{Com}(\mathsf{pp}, g)$

return (c, opn)

$\underline{\mathsf{Prv}^*(\mathsf{pp}, q, c, \mathsf{opn}, \vec{x}) \to \pi}$

$z_1 \leftarrow 2^{\hat{m}} \bmod q$

for $i \in [2, \hat{\mu}]$:

$z_i \leftarrow {z_{i-1}}^{2^{\hat{m}}} \bmod q$

$\vec{z} \leftarrow (\vec{z}_1, \ldots, \vec{z}_{\hat{\mu}})$

$\pi \leftarrow \mathsf{Prv}(\mathsf{pp}, g, (\vec{x}, \vec{z}), q)$

return π

$\underline{\mathsf{Vfy}^*(\mathsf{pp}, q, c, \vec{x}, y, \pi, \mu) \to b}$

$z_1 \leftarrow 2^{\hat{m}} \bmod q$

for $i \in [2, \hat{\mu}]$:

$z_i \leftarrow {z_{i-1}}^{2^{\hat{m}}} \bmod q$

$\vec{z} \leftarrow (\vec{z}_1, \ldots, \vec{z}_{\hat{\mu}})$

return $\mathsf{Vfy}(\mathsf{pp}, \pi, (\vec{x}, \vec{z}), q)$

Fig. 10. New modPC^* obtained by reverse Kronecker substitution. The definition of Limbs, which extracts the $\hat{m}$-bit limbs from an integer, is given in Fig. 11.

$\underline{\mathsf{Limbs}(v \in \mathbb{Z}, \hat{m} \in \mathbb{N}, \hat{\mu})}$

// Take the absolute value of v

if $v < 0$ **then** $\mathsf{sign} \leftarrow -1$ **else** $\mathsf{sign} \leftarrow 1$

// Write v as limbs in $[0, 2^{\hat{m}})$

for $i \in [0, 2^{\hat{\mu}}]$:

$\mathsf{l}_i \leftarrow (\mathsf{sign} \cdot v) \bmod 2^{\hat{m} \cdot i}$

// Negate the limbs if v was negative

return $\mathsf{sign} \cdot (\mathsf{l}_0, \ldots, \mathsf{l}_{\hat{\mu}-1})$

Fig. 11. Procedure extracting $\hat{m}$-bit limbs from an integer, handling both positive and negative inputs. Note that on a RAM machine with a constant wordsize, the time complexity of this procedure is $O(\log(v))$ as it corresponds to reading off the bits of v into $\hat{m}$-bit chunks.

F Reducing the Requirements for Indexing Commitments in our Compiler

We observe that in the proof of Theorem 3 we do not really need to invoke the extractability of the commitments to the indexing polynomials. All that is required for them is that weak evaluation binding holds. This suggests the following modified compiler:

- Let $\mathsf{modPC}_{\mathrm{idx}}$ and $\mathsf{modPC}_{\mathrm{w}}$ be two mod-PCs satisfying respectively weak evaluation binding and knowledge soundness.
- Apply the compiler in Fig. 13, with the only difference that we use $\mathsf{modPC}_{\mathrm{idx}}$ for the indexing polynomials and $\mathsf{modPC}_{\mathrm{w}}$ for the oracle polynomials.

The following theorem follows from the proof of Theorem 3.

Theorem 8. *Let* modAHP *be a knowledge-sound mod-AHP. Let* modPC_{idx} *and* modPC_{w} *be two mod-PCs satisfying respectively weak evaluation binding and knowledge soundness. Then applying the variant of the compiler in Fig. 13 described above yields a complete, full knowledge-sound interactive argument.*

G Missing Proofs

G.1 Proof of Theorem 2

We propose only a sketch of the proof since it is easy and formal versions of some of these observations are in the proof of Theorem 3.

Let $\mathsf{P}^* = (\mathsf{P}^*_{\mathrm{orcl}}, \mathsf{P}^*_{\mathrm{rst}})$ and consider the following extractor:

$\mathsf{Ext}^{\mathsf{P}^*}\left(1^\lambda, \mathsf{i}, \mathsf{x}, \mathsf{aux}\right)$

Obtain polynomial $g^*(\vec{X})$ from $\mathsf{P}^*_{\mathrm{orcl}}$

Run Decoder on $g^*(\vec{X})$ to obtain w

Return w

Now assume by contradiction that the following event has a non-negligible probability: the output of the extractor above is not a witness for the original (full) integer relation $\mathcal{R}$ and yet the verifier accepts when interacting with P^*. Now consider the prime q sampled during the interaction. Either w is a witness for $[\![\mathcal{R}_{\mathsf{n}}]\!]_q$ or it is not. The probability that the extracted string w is a "fingerprint" witness modulo q (while being not a witness for the integer relation) is negligible because of the assumptions on the good membership testing property. If it is not a fingerprint witness, however, we can invoke weak knowledge soundness (Definition 13) and conclude we reached a contradiction. □

G.2 Proof of Theorem 3

To argue completeness we need to argue that for an honestly generated proof, the decision algorithm will accept. The latter consists of two checks: those from underlying mod-AHP and the mod-PC verification. Invoking completeness of the two primitives suffices to claim completeness of the overall argument.

We now show knowledge soundness for our compiler. Our proof strategy is standard and resembles the one used in previous papers with AHP-like compilers, such as [ABC+22,CFF+21,BFS20,CHM+20]. Consider an adversary $\tilde{\mathcal{P}}$ producing an accepting transcript with probability $\tilde{p}$. We show an extractor for $\tilde{\mathcal{P}}$ in Fig. 12. Our approach at the high level:

- The extractor works by invoking the mod-AHP extractor which interacts with a mod-AHP prover P^*.
- P^* is related to $\tilde{\mathcal{P}}$ and, intuitively, is the prover that, at each round i^* before the prime is sampled returns the polynomials "behind" the commitments returned by $\tilde{\mathcal{P}}$ at the same round. For the later rounds it simply follows the prover of the plain interactive protocol part of the mod-AHP.

– To define such a P^* we need to invoke the extractor of the mod-PC. Formally, in order to do this we need to define an adversary for each of the polynomial/commitment that will be exchanged during the interaction. Such a family of adversaries is defined in Fig. 13.

Consider the knowledge soundness game for interactive arguments. Below we bound the probability that the extractor $\mathsf{Ext}_{\mathrm{ARG}}$ fails to output a witness (event $\mathsf{Ext}_{\mathrm{ARG}}$ ✗) while $\tilde{\mathcal{P}}$ successfully produces an accepting transcript in the knowledge soundness game (event $\tilde{\mathcal{P}}$ ✓). In order to do this, we make observations related to events for $\mathsf{Ext}_{\mathrm{AHP}}^{\mathsf{P}^*}$ and P^* in the context of the AHP knowledge soundness game; denote by $\mathsf{Ext}_{\mathrm{AHP}}^{\mathsf{P}^*}$ ✗ the event where $\mathsf{Ext}_{\mathrm{AHP}}^{\mathsf{P}^*}$ fails to produce a valid witness and by P^* ✓ the event where P^* succeeds in producing outputting oracle polynomials that make the AHP verifier accept. Below when expressing conjunctions, we consider the correlated events where there is only one sampling of the random coins of $\tilde{\mathcal{P}}$ and the random coins of the AHP and argument verifiers (for P^* and $\tilde{\mathcal{P}}$ respectively) uses the same random tape.

We can then observe:

$$
\begin{aligned}
&\Pr[\mathsf{Ext}_{\mathrm{ARG}} \text{ ✗ } \wedge \tilde{\mathcal{P}} \text{ ✓}] \\
=&\Pr[\mathsf{Ext}_{\mathrm{AHP}}^{\mathsf{P}^*} \text{ ✗ } \wedge \tilde{\mathcal{P}} \text{ ✓}] && (2)\\
=&\Pr[\mathsf{Ext}_{\mathrm{AHP}}^{\mathsf{P}^*} \text{ ✗ } \wedge \tilde{\mathcal{P}} \text{ ✓ } \wedge \mathsf{P}^* \text{ ✓}] + \Pr[\mathsf{Ext}_{\mathrm{AHP}}^{\mathsf{P}^*} \text{ ✗ } \wedge \tilde{\mathcal{P}} \text{ ✓ } \wedge \mathsf{P}^* \text{ ✗}] && (3)\\
\leq&\Pr[\mathsf{Ext}_{\mathrm{AHP}}^{\mathsf{P}^*} \text{ ✗ } \wedge \mathsf{P}^* \text{ ✓}] + \Pr[\mathsf{P}^* \text{ ✗} \wedge \tilde{\mathcal{P}} \text{ ✓ }] && (4)\\
\leq&\ \mathsf{negl}(\lambda) + \mathsf{negl}(\lambda) && (5)
\end{aligned}
$$

– Eq. (2) follows by construction of $\mathsf{Ext}_{\mathrm{ARG}}$.
– In Eq. (3) we apply a simple marginalization.
– In Eq. (4) we apply the elementary fact $X \to Y \implies \Pr[X] \leq \Pr[Y]$.
– We bound the left-hand summand in Eq. (5) by simply invoking knowledge soundness of the underlying mod-AHP. For the right-hand summand we invoke Lemma 5.

This concludes the proof. □

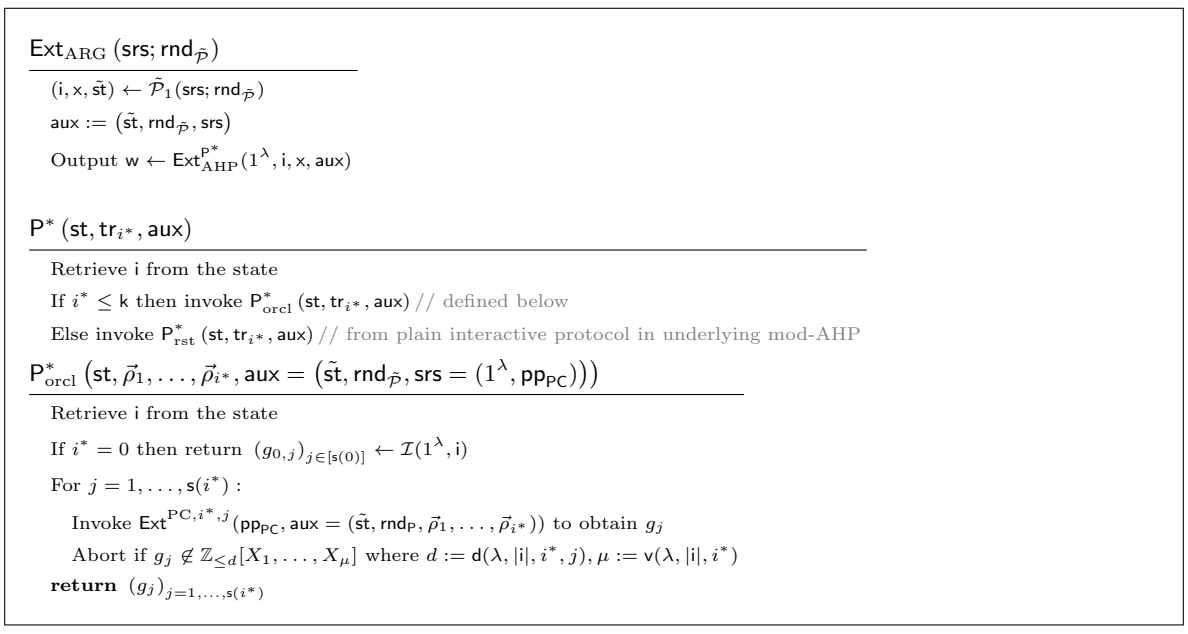


Fig. 12. Extractor for proof of Theorem 3.For each i, j the extractor $\mathsf{Ext}^{\mathrm{PC}, i, j}$ is defined as the polynomial commitment extractor for adversary $\mathcal{A}^{\mathrm{PC}, i, j}$ according to the knowledge soundness property in Definition 14. We assume that the prover P^* obtains as initial state the index i.

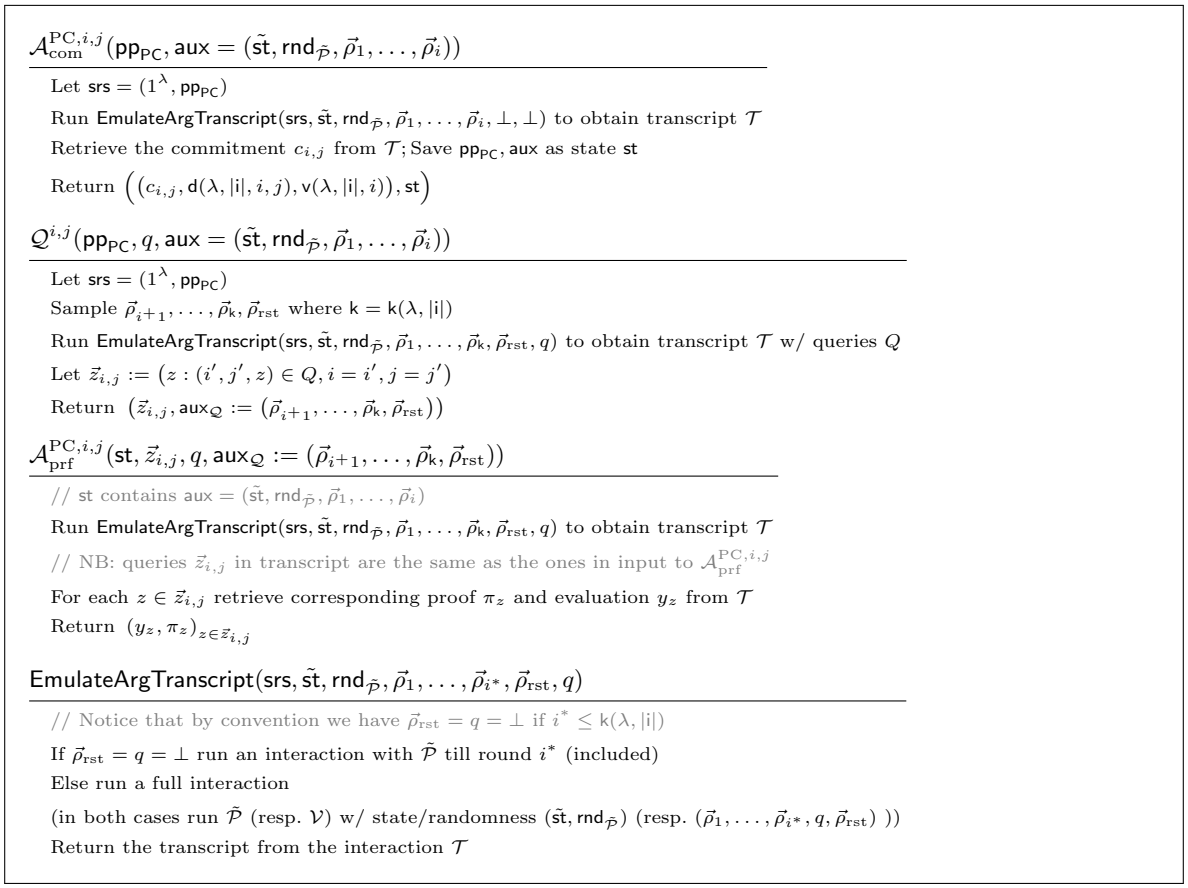


Fig. 13. Auxiliary algorithm definitions for Fig. 12. We assume each $\mathcal{A}^{\mathrm{PC},i,j} = \Big(\mathcal{A}^{\mathrm{PC},i,j}_{\mathrm{com}}, \mathcal{A}^{\mathrm{PC},i,j}_{\mathrm{prf}}\Big)$ has embedded the parameter $|\mathsf{i}|$.

Lemma 5. *In the proof of Theorem 3 the quantity* $\Pr[\mathsf{P}^* \not\checkmark \wedge \tilde{\mathcal{P}} \checkmark]$ *in Eq.* (4) *is negligible.*

Proof. By inspection of P^* we can observe that there are two ways the event $\mathsf{P}^* \not\checkmark$ may occur: some of the polynomials $g_{i^*,j}$ may:

(i) have the wrong degree or number of variables after extraction (for some $i^* \geq 1$); or
(ii) disagree with the output of $\tilde{\mathcal{P}}$ in the following way (recall that we consider the same verifier's challenges for both the AHP and argument interaction):
- Let z be some challenge point for $g_{i^*,j}$ in the transcript.
- Let y be the evaluation output y claimed by $\tilde{\mathcal{P}}$ for the polynomial opening $c_{i^*,j}$ when evaluated on z taking the value $\mod q$ (the prime from the transcript).
- We say that the event $\mathsf{Bad}(g_{i^*,j})$ occurs if $g_{i^*,j}(z) \not\equiv y \mod q$ for some challenge z and corresponding output y claimed by $\tilde{\mathcal{P}}$.

The event above intuitively means that the oracle polynomial output $g_{i^*,j}$ by P^* "does not agree" with the claims by $\tilde{\mathcal{P}}$. Notice that, if condition *(i)* does not occur but $\mathsf{P}^* \not\checkmark$ and $\tilde{\mathcal{P}}$ do, then it must be the case that condition *(ii)* occurred (otherwise the decision algorithm would have accepted for P^* as well).

We can easily observe that the probability that *(ii)* occurs is negligible because of the negligible knowledge soundness error of the polynomial commitment (we will show a more formal reduction below for a similar case). Therefore, by applying the observations above and a simple union bound we can conclude that $\Pr[\mathsf{P}^* \not\checkmark \wedge \tilde{\mathcal{P}} \checkmark]$ is at most:

$$\sum_j \Pr[\tilde{\mathcal{P}} \checkmark \wedge \ \mathsf{Bad}(g_{0,j}) : (g_{0,j})_j \leftarrow \mathcal{I}(1^\lambda, \mathsf{i})] + \sum_{i^* \neq 0, j} \Pr[\tilde{\mathcal{P}} \checkmark \wedge \ \mathsf{Bad}(g_{i^*,j}) : g_{i^*,j} \leftarrow \mathsf{Ext}^{i^*,j,\mathrm{PC}}] + \mathsf{negl}(\lambda)$$

If we assume, for sake of contradiction, that $\Pr[\mathsf{P}^* \not\checkmark \wedge \tilde{\mathcal{P}} \checkmark]$ is non-negligible then at least some term in the two sums above must be non-negligible. We now show that we can recast the event encoded by the Bad predicate as a knowledge soundness game of the polynomial commitment or as a weak-evaluation binding game (for the case of the polynomials from the indexer).

By knowledge soundness of the mod-PC we know the following probability is at most negligible:

$$\Pr\left[\begin{array}{l|l} \left(g_{i^*,j} \notin \mathbb{Z}_{\leq 1}[X_1, \ldots, X_\mu] \vee \right. & \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ \left. \exists k \in [m] \ g_{i^*,j}(z_k) \not\equiv y_k \mod q \right) \wedge & ((c_{i^*,j}, \mu), \mathsf{st}) \leftarrow \mathcal{A}_{\mathrm{com}}^{\mathrm{PC},i^*,j}(\mathsf{pp}, \mathsf{aux}) \\ & q \leftarrow\!\!\$\ \mathbb{P}_\lambda \\ \bigwedge_k \mathsf{Vfy}\left(\mathsf{pp}, q, c_{i^*,j}, z_k, y_k, \pi_k^{(\mathrm{eval})}\right) = 1 & \left((z_k)_{k\in[m]}, \mathsf{aux}_{\mathcal{Q}}\right) \leftarrow \mathcal{Q}(\mathsf{pp}, \mathsf{aux}, q) \\ & \left(y_k, \pi_k^{(\mathrm{eval})}\right)_{k\in[m]} \leftarrow \mathcal{A}_{\mathrm{prv}}(\mathsf{st}, q, (z_k)_k, \mathsf{aux}_{\mathcal{Q}}) \\ & g_{i^*,j} \leftarrow \mathsf{Ext}^{\mathrm{PC},i^*,j}(\mathsf{pp}, \mathsf{aux}) \end{array}\right]$$

Now, for a more succinct notation, let us define the following event:

$$\mathsf{E}_{\mathrm{adv,KSND}} := \left\{\begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ ((c_{i^*,j}, \mu), \mathsf{st}) \leftarrow \mathcal{A}_{\mathrm{com}}^{\mathrm{PC},i^*,j}(\mathsf{pp}, \mathsf{aux}) \\ q \leftarrow\!\!\$\ \mathbb{P}_\lambda \\ \left((z_k)_{k\in[m]}, \mathsf{aux}_{\mathcal{Q}}\right) \leftarrow \mathcal{Q}(\mathsf{pp}, \mathsf{aux}, q) \\ \left(y_k, \pi_k^{(\mathrm{eval})}\right)_{k\in[m]} \leftarrow \mathcal{A}_{\mathrm{prv}}(\mathsf{st}, q, (z_k)_k, \mathsf{aux}_{\mathcal{Q}}) \\ g_{i^*,j} \leftarrow \mathsf{Ext}^{\mathrm{PC},i^*,j}(\mathsf{pp}, \mathsf{aux}) \end{array}\right\}$$

and let us observe that, by how we defined Bad, we have:

$$\mathsf{Bad}(g_{i^*,j}) \iff \left(g_{i^*,j} \notin \mathbb{Z}_{\leq 1}[X_1, \ldots, X_\mu] \ \vee \ \exists k \in [m] \ g_{i^*,j}(z_k) \not\equiv y_k \mod q\right)$$

By observing that the coin tosses in $\mathsf{E}_{\mathrm{adv,KSND}}$ are distributed exactly as in an interaction with an honest argument verifier, we can then conclude that:

$$\begin{aligned}\mathsf{negl}(\lambda) &\geq \Pr\left[\mathsf{Bad}(g_{i^*,j}) \wedge \bigwedge_k \mathsf{Vfy}\left(\mathsf{pp}, q, c_{i^*,j}, z_k, y_k, \pi_k^{(\mathrm{eval})}\right) = 1 \middle| \mathsf{E}_{\mathrm{adv,KSND}}\right] \\ &\geq \Pr\left[\mathsf{Bad}(g_{i^*,j}) \wedge \tilde{\mathcal{P}} \checkmark\right]\end{aligned}$$

Now for the indexer polynomials, assume that $\Pr[\tilde{\mathcal{P}} \checkmark \wedge \ \mathsf{Bad}(g_{0,j}) : (g_{0,j})_j \leftarrow \mathcal{I}(1^\lambda, \mathsf{i})]$ is non-negligible for some j. Consider some evaluation proof π from $\tilde{\mathcal{P}}$ with respect to commitment $c_{0,j}$ and some evaluation point z with claimed output y (modulo q). By is definition, if the event $\mathsf{Bad}(g_{0,j})$ occurs then $g_{0,j}(z) \not\equiv y \mod q$. Let p the probability π passes the verification and $\mathsf{Bad}(g_{0,j})$ occurs. This probability should be negligible by definition of weak-evaluation binding because $c_{0,j}$ is generated honestly from $g_{0,j}$; however, this probability is at least $\Pr[\tilde{\mathcal{P}} \checkmark \wedge \ \mathsf{Bad}(g_{0,j}) : (g_{0,j})_j \leftarrow \mathcal{I}(1^\lambda, \mathsf{i})]$, which we assumed to be non-negligible. Absurd. □

G.3 Proof of Theorem 4

It is immediate to see that Fig. 3 satisfies the syntactic properties of the mod-AHPs (oracles, prime sampling, subsequent interactive argument) and that it has a simple prover (Definition 12). Notice that the multilinear encoding of the witness, $\tilde{w} \in \mathbb{Z}[X_1, \ldots, X_{\log N}]$, can be computed over the integers.

We need to show weak knowledge soundness, that is, informally, that for any efficient adversary there is a decoder such that for any prime $q \in \mathbb{P}_\lambda$ with overwhelming probability we are able to extract a witness for the associated fingerprinting relation. (see Definition 7) or the prover fails to pass the verification checks. Here we can easily invoke the results from [Set20]. We observe that after the prime is sampled the protocol in Fig. 3 is exactly the one in the original description of Spartan. The only difference is in the language used to describe since we use the language of AHP over $\mathbb{Z}$ with modular remainder queries.[25]

We now show how we can invoke results from the proof of Theorem 5.1 from [Set20], which shows the knowledge soundness of Spartan. The proof (which is phrased in the language of witness-extended emulation) essentially proves that Spartan is an extractable AHP (in the standard sense). All we need to bridge this fact into our proof is:

1. to show that the proof in Spartan is not impacted by the polynomial $\tilde{w}(\vec{X})$ being over the integers instead of in the finite field at the start of the protocol;
2. to show that what is extracted is actually a witness for $[\![\mathcal{R}^{\mathrm{R1}}]\!]_q$;

Let q be some prime and let Ext_q be the Spartan extractor for the field $\mathbb{F}_q$. Such an extractor exists as by Theorem 5.1 in [Set20]. This extractor works by looking at the polynomial $\tilde{w}$, evaluating it on the boolean hypercube and returning the resulting vector. The security result we cite above shows that this produces a valid witness in $\mathbb{F}_q$ with overwhelming probability.

We define our decode exactly as the extractor above. We want to claim that this algorithm will produce a vector w such that $(\mathsf{i}, [\![\mathsf{x}]\!]_q, [\![\mathsf{w}]\!]_q) \in [\![\mathcal{R}_\mathsf{n}]\!]_q$ unless the adversary does not pass verifier with substantial probability.

To argue the above we proceed as follows. Let (A, B, C) be a $\mathbb{Z}$-$R1CS$ and let q be a prime. For any adversary $\mathcal{A}'$ for the relation yielded by (A, B, C) against the experiment in Definition 13, we can observe that there exists an adversary $\mathcal{A}$ for the original Spartan for the "fingerprinted" R1CS (which is a valid $\mathbb{F}$-R1CS and therefore a valid constraint system for the original Spartan) that engages in the knowledge soundness game with the original extractor Ext with the same success probability. We first observe that in our protocol the indexing polynomials are $\mathsf{MLE}(\mathsf{A}), \mathsf{MLE}(\mathsf{B}), \mathsf{MLE}(\mathsf{C})$ but all the evaluations of those multilinear extensions at the end of protocol are modulo q. Therefore, by Lemma 1 these evaluations are the same as the evaluations of the multilinear extensions corresponding to the R1CS fingerprinted relation (see Definition 7). This guarantees correspondence between the R1CS as described above.

[25] It has already been observed in other works that Spartan is essentially an algebraic holographic proof (in the standard sense of "finite-field" AHPs) over multivariate polynomials [CBBZ23].

Then we can construct $\mathcal{A}$ as the adversary that internally runs $\mathcal{A}'$ but provides oracle access to the polynomial $\tilde{w}_q$ that behaves exactly the same as $\tilde{w}$ in $\mathbb{Z}_q$. Because of the observations on the MLE of the matrices, we can finally observe that the output of a verifier having access to only evaluations modulo q of an integer polynomial $\tilde{w}(X)$ (as it is the case in Fig. 3) would be the same of that of a verifier having access to $\tilde{w}_q$. This proves the claim above and concludes the proof. □

G.4 Proof of Theorem 9

Consider the compiler in Fig. 2 and apply modPC to the indexer oracle polynomials (recall that we have no other oracle polynomials). We construct modPC^* from the argument obtained from the compiler as follows:

- The setup consists of the setup of mod-PC for dense polynomial.
- The commitment stage consists of the indexing stage: it receives as input the index description (a polynomial) and the output commitment is the vector of commitments from the indexing stage.
- The opening stage (recall we consider interactive opening) takes as input a randomly sampled prime q as well as the commitment and the pair point–evaluation and it consists of the online stage of the argument *after the sampling of the prime*.

Now we need to argue this construction satisfies weak evaluation binding. Assume that, by sake of contradiction, it is not. That is, there exists an efficient adversary $\mathcal{A}$ that:

- outputs a multilinear polynomial g;
- after seeing a random prime q, outputs $\vec{x}$ and y such that $g(\vec{x}) \neq y$;
- with non-negligible probability gets to convince the verifier $g(\vec{x}) = y$ w.r.t the honestly generated commitment c_g.

Consider the opening transcript of $\mathcal{A}$. Intuitively it is either containing false claims for the output of the indexing polynomials (which would entail breaking the delayed-input soundness of the underlying mod-AHP) or it is producing convincing mod-PC proofs for false outputs of the indexing polynomials (breaking weak-evaluation binding of modPC). We thus reach a contradiction[26]. □

G.5 Proof of Theorem 10

The core efficiency properties of the construction are argued in [Set20]; the resulting efficiency of our construction follow straightforwardly from those observations. For what concerns security, we can also easily rely on the security arguments in [Set20]: The proof of Lemma 7.6 in [Set20] essentially argues that the construction in Fig. 14 is an interactive argument with negligible soundness if $\mathsf{PC}_{\mathrm{agn}}$ is an extractable polynomial commitment. The original proof also assumes that the indexing polynomials are not provided through oracle access but are instead committed and then evaluated through the opening of a polynomial commitment (with weak evaluation binding properties). The only significant change with the proof is then the fact that we are assuming oracle access instead of polynomial commitments opening but this can clearly only strengthen the claims in the original proof.

The proof of Lemma 7.6 in [Set20] argues for soundness of the protocol as an argument for $\mathcal{R}^{\mathrm{poly}}$; this translates directly to delayed-input soundness for mod-AHPs. □

H Building mod-PCs for Sparse Polynomials and Delayed-Input Soundness

In this section we show how to build efficient mod-PCs for sparse integer polynomials. The result of this section can be used to instantiate the polynomial commitments for the indexing R1CS polynomials in Zaratan.

[26] A formal version of these last steps is analogous to the ones in the proof of Theorem 3, to which we refer the reader.

H.1 Delayed-Input (Deterministic) Soundness

A mod-AHP with delayed-input soundness can be thought of as the deterministic analogue of a "fingerprint-only" knowledge sound mod-AHP (Definition 13). The latter notion states informally states that, from a prover with good success probability during a mod-AHP interaction, we are able to "extract" a valid witness *for the associated fingerprinting relation* (Definition 3. The definition we provide in this section is very similar but it will focus on deterministic relations (there is no witness, just an index and a statement). What we require is that if a prover is successful during a mod-AHP interaction related to index i and statement x, then the input's fingerprint should be a valid statement for the associated fingerprint relation, i.e. $[\![\mathcal{R}]\!]_q(\mathsf{i}, [\![\mathsf{x}]\!]_q) = 1$. The reason why this form of soundness is "delayed-input" is because we want its security to hold even if the statement were to be provided *after* the prime is sampled. This is crucial in order to obtain secure polynomial commitments over the integers (as we define them in Section 5).

The following definition is, from a syntactical standpoint the same as that for mod-AHP. For this reason we do not define it completely from scratch.

Definition 20. *We say a mod-AHP is "for deterministic relations only" if the prover sends no oracle polynomials before the prime is sampled. We denote the behavior of the interactive verifier after the prime is sampled as* V_{post}

Notice that our next definition is well-formed since the verifier in any mod-AHP is public coin and therefore it does not need to know the public input before sampling the prime or producing any other challenge.

Definition 21. *Consider a mod-AHP for deterministic relations only (Definition 20) for an indexed (deterministic) relation* $\mathcal{R}$*. Let* $[\![\cdot]\!]$ *be an associated fingerprinting relation for* $\mathcal{R}$ *(Definition 3). We say the mod-AHP has delayed-input soundness error* ϵ *if for all* $\lambda, n \in \mathbb{N}$*, index* i *and auxiliary input* aux*, for all PPT adversaries* $\mathcal{A} = (\mathcal{A}_{inp}, \mathcal{A}_{post})$*:*

$$\Pr\left[\begin{array}{c} (\mathsf{i}, [\![\mathsf{x}]\!]_q, \bot) \notin [\![\mathcal{R}_{\mathsf{n}}]\!]_q \wedge \\ \langle \mathcal{A}_{post}(\mathsf{st}), \mathsf{V}_{post}^{\mathcal{I}(1^\lambda,\mathsf{i})}(1^\lambda, \mathsf{x}, q)\rangle = 1 \end{array} : \begin{array}{c} q \leftarrow\!\$\ \mathbb{P}_\lambda \\ (\mathsf{x}, \mathsf{st}) \leftarrow \mathcal{A}_{inp}(1^\lambda, \mathsf{i}, \mathsf{aux}, q) \end{array}\right] \leq \epsilon$$

The following theorem shows that delayed-input soundness can be lifted to obtain weak evaluation binding for functional commitments. Notice that in the following result we do not require the associated fingerprinting relation to be a good membership test as in Theorem 8.

Remark 7. While we state the following theorem for polynomial commitments for integer polynomials with modular remainder opening it is immediately possible to show this result (and define equivalent notions) for *functional commitments over integer vectors* with opening to any "modular remainder" restriction of a function f. An easy example in such sense is commitments to vectors in $\mathbb{Z}^n$ where each element can be opened in $\mathbb{Z}_q$ for a sampled q.

The next definition formalizes what we mean for AHP over $\mathbb{Z}$ with modular remainder queries to be an efficient protocol for sparse polynomial evaluation. It intuitively states that the prover can run linearly the number of non-zero entries of a sparse polynomial.

Definition 22. *We say that a mod-AHP* modAHP *over* $\mathcal{R}^{poly}$ *is efficient for sparse polynomial evaluation if all of the following conditions hold. Let* g *be the sparse polynomial (see Section 2) describing some index for the relation and let* $c\mu$ *be the number of variables over which* g *is defined (for a parameter* $\mu \in \mathbb{N}$ *and a constant* $c > 1$*), then: 1) The output of the indexing step consists of a constant number of oracle multilinear polynomials each in* μ *variables. 2. The total running time of the prover is* $O_\lambda(2^\mu \cdot \log_2 \|g\|_\infty)$*.*

Theorem 9 (Delayed-Input Soundness ⇒ Sparse mod-PC). *Assume: (a) a mod-PC* modPC *with weak evaluation binding*[27]*; (b) a multilinear mod-AHP* modAHP *for* $\mathcal{R}^{poly}$ *that: (i) is efficient for sparse polynomial evaluation (Definition 22); (ii) has negligible delayed-input soundness. Then there exists a weak evaluation binding mod-PC* modPC^* *for sparse polynomials with interactive opening (see Definition 16 and Remark 3).*

[27] We stress that this mod-PC is for dense polynomials only.

H.2 A construction from SPARK [Set20]

In this section we reinterpret another building block from [Set20] as a mod-AHP. In particular the SPARK construction to lift a dense polynomial commitment to a sparse one. SPARK as a mod-AHP will require not only indexing oracle polynomials over the integers but also others (see next remark).

Remark 8 (Augmenting mod-AHPs). The results in this section will apply a natural generalization of the mod-AHPs we described in Definition 10 where we allow for oracle polynomials defined over $\mathbb{F}_q$ for a sampled prime q. These can be compiled through a "field-agnostic" polynomial commitment (that can take in input the field at commitment time), which we show that we can build from our modPC.

Indexing stage: Given in input a sparse multilinear polynomial $\widetilde{M}$ in 2μ variables, output multilinear polynomials in $O(\mu)$ variables

$$\widetilde{row}, \widetilde{col}, \widetilde{val}, \left(\widetilde{read\text{-}ts_X}, \widetilde{write\text{-}ts_X}, \widetilde{audit\text{-}ts_X}\right)_{X \in \{row, col\}}$$

as defined in [Set20, Section 7.2].

Opening stage: To claim that $\widetilde{M}(\vec{x}) \equiv y \mod q$:

- Compute multilinear polynomials in $O(\mu)$ variables $\widetilde{e}_{row}, \widetilde{e}_{col}$ as defined in [Set20, Section 7.2.1] (notice that these polynomials are defined over $\mathbb{Z}_q$).
- Send oracles $\widetilde{e}_{row}, \widetilde{e}_{col}$.
- Continue the protocol as described for $\mathsf{PC}_{\mathrm{SPARK}}$ in [Set20] with two nuances:
  - Whenever the prover provides an opening proof for one of the indexing polynomials, simply let the mod-AHP verifier query that polynomial through its oracle access (as we did in our variant of Spartan in Fig. 3).
  - Ditto for each of the polynomial commitment openings for $\widetilde{e}_{row}, \widetilde{e}_{col}$ during the execution of Hyrax[28].

Fig. 14. A variant of the SPARK construction from [Set20] as an augmented mod-AHP for $\mathcal{R}^{\mathrm{poly}}$. For oracle polynomials we use the color conventions: indexing polynomials in $\mathbb{Z}[\vec{X}]$ in cyan; prime-dependent polynomials in $\mathbb{Z}_q[\vec{X}]$ in magenta. For additional details on SPARK, see [Set20, Section 7.2]

Theorem 10. *The construction in Fig. 14 is an (augmented) mod-AHP for $\mathcal{R}^{poly}$ with negligible delayed-input soundness; it is efficient for sparse polynomials.*

Corollary 2 is implied by our construction of mod-PC for dense polynomials, Theorem 9, Theorem 10 and by the security of our modPC.

I Additional Definitions for mod-PCs

I.1 Weak Evaluation binding

Definition 23 (Weak Evaluation Binding). *For any PPT adversary $\mathcal{A}$, $\lambda \in \mathbb{N}$, $\mu \in \mathbb{N}$:*

$$\mathsf{negl}(\lambda) \geq \Pr\left[\begin{array}{l|ll} \mathsf{Vfy}\left(\mathsf{pp}, q, c, y, \pi\right) = 1 & : & \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ & & (f, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{pp}) \\ & & (c, \mathsf{opn}) \leftarrow \mathsf{Com}(\mathsf{pp}, f) \\ & & q \leftarrow\$ \mathbb{P}_\lambda \\ & & (x, y, \pi) \leftarrow \mathcal{A}_2(\mathsf{st}, q) \\ & & y \not\equiv f(x) \mod q \\ & & f \in \mathbb{Z}_{\leq 1}[X_1, \ldots, X_\mu] \end{array}\right]$$

[28] Like in [Set20], we do not need Hyrax's zero-knowledge compiler [WTs+18]. For us this is crucial because otherwise this would require hardness of DLOG for a group of order q for a freshly sampled prime q (which we would not be able to instantiate, at least efficiently; see, e.g. Footnote 1 in [GLS+23]).

For a negligible function $\mathsf{negl}(\lambda)$.

I.2 (Strong) Knowledge Soundness over Adversarial Primes

Below we define a stronger version of the knowledge soundness we provide in Definition 14. It is easy to see that the latter is implied by the one in Definition 24.

Definition 24 ((Strong) Knowledge soundness for mod-PC over adversarial primes). *We say a mod-PC has strong knowledge soundness error ϵ over adversarial primes if for any $\lambda, M \in \mathbb{N}$ and PPT $\mathcal{A} = (\mathcal{A}_{com}, \mathcal{A}_{prv})$ there exists a non-uniform polynomial time extractor Ext such that for any efficient query algorithm (with random tape independent from that of the adversary) $\mathcal{Q}$ auxiliary string $\mathsf{aux} \in \{0,1\}^{\mathsf{poly}(\lambda)}$, the following probability is at most ϵ:*

$$\Pr\left[\begin{array}{l} \Big(f \notin \mathbb{Z}_{\leq 1}[X_1,\ldots,X_\mu] \vee \\ \exists j \in [m]\ f(z_j) \not\equiv y_j \mod q\Big) \wedge \\ q \in \mathbb{P}_\lambda \wedge \\ \bigwedge_j \mathsf{Vfy}\left(\mathsf{pp}, q, c, z_j, y_j, \pi_j^{(eval)}\right) = 1 \end{array} : \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ ((c,\mu,q), \mathsf{st}) \leftarrow \mathcal{A}_{com}(\mathsf{pp}, \mathsf{aux}) \\ \left((z_j)_{j\in[m]}, \mathsf{aux}_{\mathcal{Q}}\right) \leftarrow \mathcal{Q}(\mathsf{pp}, \mathsf{aux}, q) \\ \left(y_j, \pi_j^{(eval)}\right)_{j\in[m]} \leftarrow \mathcal{A}_{prv}(\mathsf{st}, (z_j)_j, \mathsf{aux}_{\mathcal{Q}}) \\ f \leftarrow \mathsf{Ext}(\mathsf{pp}, \mathsf{aux}) \end{array}\right]$$

where above the extractor has access to the random tape of the adversary.

J Oracle Polynomials over $\mathbb{F}$ in mod-AHP

J.1 Augmented model

In Fig. 15 augment the model of mod-AHP presented in Definition 10 with an additional round[29] after prime sampling where the prover can send oracle polynomials *which do depend* on the sampled prime and are defined over $\mathbb{F}_q$. Below we mark in blue the extra steps in the protocol. We also mention that the protocol is now parametrized by three additional functions $\mathsf{s}^\star, \mathsf{d}^\star, \mathsf{v}^\star$ for the number of polynomials, degree and number of variables respectively.

[29] For simplicity we provide a presentation for one round only, but this is an arbitrary choice—one round is sufficient for us to model the setting in Section H.

– **Indexing phase** The indexer $\mathcal{I}$ receives as input a security parameter 1^λ and the index i for $\mathcal{R}$, and outputs $\mathsf{s}(0)$ polynomials $p_{0,1}, \ldots, p_{0,\mathsf{s}(0)} \in \mathbb{Z}[\vec{X}]$ of degrees at most $\mathsf{d}(\lambda, |\mathsf{i}|, 0, 1), \ldots, \mathsf{d}(\lambda, |\mathsf{i}|, 0, \mathsf{s}(0))$ respectively; $|\vec{X}| = \mathsf{v}(\lambda, |\mathsf{i}|, 0)$.
– **Online phase** The prover P receives $(1^\lambda, \mathsf{i}, \mathsf{x}, \mathsf{w})$, for an instance x and witness w such that $(\mathsf{i}, \mathsf{x}, \mathsf{w}) \in \mathcal{R}$. The verifier V receives 1^λ, x and oracle access to the polynomials output by $\mathcal{I}(1^\lambda, \mathsf{i})$[30]. The prover P and the verifier V interact over a number of rounds as follows:
  - **Integer Oracle Polynomials Phase:** In the i-th round, $i \in \{1, \ldots, \mathsf{k}(\lambda, |\mathsf{i}|)\}$, the verifier V sends messages $\vec{\rho}_i \in \{0,1\}^{\mathsf{poly}(\lambda)}$ to the prover P; the prover P responds with $\mathsf{s}(i)$ oracle polynomials $p_{i,1}, \ldots, p_{i,\mathsf{s}(i)} \in \mathbb{Z}[\vec{X}]$ where each is respectively of degree at most $\mathsf{d}(\lambda, |\mathsf{i}|, i, 1), \ldots, \mathsf{d}(\lambda, |\mathsf{i}|, i, \mathsf{s}(i))$ and $|\vec{X}| = \mathsf{v}(\lambda, |\mathsf{i}|, i)$.
  - **Prime Sampling Phase:** After k rounds, the verifier samples a prime $q \leftarrow\!\!\$\ \mathbb{P}_\lambda$ and sends it to P.
  - **Prime-Dependent Oracle Round:** The verifier sends random challenge $\vec{\rho}^\star$; the prover responds with $s^\star := \mathsf{s}^\star(\lambda, |\mathsf{i}|)$ oracle polynomials $p_1^\star, \ldots, p_{s^\star}^\star \in \mathbb{F}_q[\vec{X}]$ all of degree $\mathsf{d}^\star(\lambda, |\mathsf{i}|)$ and number of variables $\mathsf{v}^\star(\lambda, |\mathsf{i}|)$
  - **Plain Interaction Phase:** The prover and verifier engage in a plain interactive protocol (see Section 2) for k' rounds:

$$\mathsf{tr}_{\mathrm{rst}} := (\vec{m}_{\mathrm{rst}}, \vec{\rho}_{\mathrm{rst}}) \leftarrow \mathsf{transcript}_\lambda\big(\langle \mathsf{P}_{\mathrm{rst}}(\vec{\rho}_1, \ldots, \vec{\rho}_{\mathsf{k}}, q), \mathsf{V}_{\mathrm{rst}}\rangle\big)$$

– **Query phase** Using the whole transcript, the verifier outputs a set of queries for the oracle polynomials. The verifier outputs a query set Q for the integer oracle polynomials as well as a set $Q^\star$ for the oracle $(p_j^\star)_{j \in [s^\star]}$ which consists of pairs $(j, z \in \mathbb{F}_q)$, the response to which is $p_j^\star(z)$.
– **Decision phase** The verifier outputs accept or reject based on the answers received to the queries Q and $Q^\star$, its randomness and the whole transcript.

Fig. 15. Augmented mod-AHP

J.2 Prime-Agnostic Polynomial Commitments

A field-agnostic polynomial commitment is like an ordinary polynomial commitment but it is not restricted to work within a specific finite field fixed at setup time. We will use this type of polynomial commitments to compile the augmented mod-AHP described in Section J.1.

Below, whenever we write $\mathbb{F}$ we mean the finite field $\mathbb{F}_q$ for a prime q that will be obvious from the context. All the arithmetic in this subsection is over $\mathbb{F}$.

Definition 25. *A field-agnostic polynomial commitment consists of a tuple* $\mathsf{PC}_{agn} = (\mathsf{Setup}, \mathsf{Com}, \mathsf{ProveEval}, \mathsf{VfyEval})$ *such that:*

$\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$*: on input a security parameter* $\lambda \in \mathbb{N}$ *it outputs public parameters of the scheme.*
$\mathsf{Com}(\mathsf{pp}, q, g \in \mathbb{F}_{\leq 1}[X_1, \ldots, X_\mu]) \to (c, \mathsf{opn})$*: on input public parameters, a prime* q *a multilinear polynomial* g*, it outputs a commitment* c *and an additional opening string* opn *(used as auxiliary input for opening).*
$\mathsf{ProveEval}(\mathsf{pp}, q, c, \mathsf{opn}, z) \to \pi$*: on input public parameters* pp*, prime* q*, commitment* c*, opening* opn *and* $z \in \mathbb{F}$*, it outputs a proof* π *certifying the value* $g(z)$*.*
$\mathsf{VfyEval}(\mathsf{pp}, q, c, z, y, \pi) \to b \in \{0,1\}$*: on input pubic parameters, prime* q*, commitment* c*, claimed value* $y \in \mathbb{F}_q$ *and proof* π*, it outputs a bit accepting or rejecting the proof.*

Correctness. For any $\lambda \in \mathbb{N}$, $\mu \in \mathbb{N}$, prime $q \in \mathbb{P}_\lambda$, $g \in \mathbb{F}_{\leq 1}[X_1, \ldots, X_\mu]$ and $z \in \mathbb{F}$, the following probability is overwhelming:

$$\Pr\left[\mathsf{VfyEval}\left(\mathsf{pp}, q, c, y, \pi\right) = 1 \quad : \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ (c, \mathsf{opn}) \leftarrow \mathsf{Com}(\mathsf{pp}, q, g) \\ \pi \leftarrow \mathsf{ProveEval}(\mathsf{pp}, q, c, \mathsf{opn}, z) \\ y := g(z) \end{array}\right]$$

Knowledge soundness (with knowledge error ϵ). For any $\lambda \in \mathbb{N}$ and PPT $\mathcal{A} = (\mathcal{A}_{\text{com}}, \mathcal{A}_{\text{prv}})$ there exists a non-uniform polynomial time extractor such that for any efficient query algorithm (with random tape independent from that of the adversary) $\mathcal{Q}$ auxiliary string $\mathsf{aux} \in \{0,1\}^{\mathsf{poly}(\lambda)}$, the following probability is at most ϵ:

$$\Pr\left[\begin{array}{l} \Big(g \notin \mathbb{F}_{\leq 1}[X_1,\ldots,X_\mu] \vee \\ \exists j \in [m]\ g(z_j) \neq y_j\Big) \wedge \\ q \in \mathbb{P}_\lambda \wedge \\ \bigwedge_j \mathsf{VfyEval}\left(\mathsf{pp}, q, c, z_j, y_j, \pi_j\right) = 1 \end{array} : \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ \left((q,c,\mu), \mathsf{st}\right) \leftarrow \mathcal{A}_{\text{com}}(\mathsf{pp}, \mathsf{aux}) \\ \Big(\left(z_j\right)_{j\in[m]}, \mathsf{aux}_{\mathcal{Q}}\Big) \leftarrow \mathcal{Q}(\mathsf{pp}, \mathsf{aux}, q) \\ \left(y_j, \pi_j\right)_j \leftarrow \mathcal{A}_{\text{prv}}(\mathsf{st}, q, \left(z_j\right)_j, \mathsf{aux}_{\mathcal{Q}}) \\ g \leftarrow \mathsf{Ext}(\mathsf{pp}, \mathsf{aux}) \end{array}\right]$$

where above the extractor has access to the random tape of the adversary[31].

Remark 9 (Existing constructions). We remark that field-agnostic constructions such as Brakedown [GLS+23] and Orion [XZS22] satisfy our definition.

Constructions from mod-PCs with Strong Extractability It is rather straightforward to produce a field-agnostic polynomial commitment from a mod-PC: at commitment stage we ignore the prime q in input (since the mod-PC's commitment algorithm is for polynomials over the integers); the other algorithms follow the same syntax and can be trivially in a straightforward manner. In order to argue security, however, we need the mod-PC to satisfy the stronger form of knowledge soundness where the prime can be provided by the adversary (Definition 24). Showing security of the resulting construction is trivial.

Theorem 11. *If there exists a mod-PC with negligible strong knowledge error (Definition 24) then there exists a field-agnostic polynomial commitment with the same efficiency with negligible knowledge error.*

From the fact that our construction satisfies Definition 24 (see Remark 9) we have the following corollary.

Corollary 4. *There exists a field-agnostic polynomial commitment with negligible knowledge error secure in the GGUO.*

J.3 Extending the compiler

In Fig. 16 we present an extended version of the compiler in Fig. 2 to apply to the augmented mod-AHPs defined in Fig. 15. The approach we use is straightforward and consists of applying a field-agnostic polynomial commitment to the extra round of oracle polynomials in $\mathbb{F}_q$. In a sense, this stage of the protocol is compiled almost exactly as done in standard AHP compilers in finite fields [CFF+21, CHM+20, ABC+22]. As a consequence the proof of security of Theorem 12 also follows directly from a minor variant of the proofs of security from these works and the one we present for Theorem 3.

Theorem 12. *Let* modAHP *be an agumented AHP over $\mathbb{Z}$ with modular remainder queries (Definition 10 and Fig. 15) for $\mathcal{R}$, let PC_{prj} be a mod-PC (Definition 14) satisfying weak-evaluation binding and with negligible knowledge soundness error, let PC_{agn} be a field-agnostic PC_{agn} with negligible knowledge soundness error then the construction in Fig. 16 is an interactive argument with preprocessing (Definition 2) for $\mathcal{R}$.*

[31] And therefore does not need to get q as input.

Setup$(1^\lambda, N)$: ... $\mathsf{pp}_{\mathsf{agn}} \leftarrow \mathsf{PC}_{\mathrm{agn}}.\mathsf{Setup}(1^\lambda)$ Return $\mathsf{srs} = \left(1^\lambda, \mathsf{pp}_{\mathsf{PC}}, \mathsf{pp}_{\mathsf{agn}}\right)$
Indexer$(\mathsf{srs}, \mathsf{i})$: ...
Interaction:

- Integer Polynomial Oracle phase: ...
- Prime sampling: After k rounds, $\mathcal{V}$ obtains a prime q and sends it to $\mathcal{P}$.
- Prime-dependent Oracle round:
  1. $\mathcal{V}$ receives random challenges $\vec{\rho}^\star$ from V and forwards them to $\mathcal{P}$.
  2. $\mathcal{P}$ forwards $\vec{\rho}^\star$ to P which replies with polynomials $(g^\star)_{j=1}^{s^\star}$.
  3. $\mathcal{P}$ compute commitments

$$\left(c_j^\star, \mathsf{opn}_j^\star\right) = \mathsf{PC}_{\mathrm{agn}}.\mathsf{Com}\left(\mathsf{pp}_{\mathsf{agn}}, q, g_j^\star\right), \quad j = 1, \ldots, s^\star$$

  and sends $\left(c_j^\star\right)_{j=1,\ldots,s^\star}$ to $\mathcal{V}$.
- Plain-Interaction phase: ...
- Query Phase:
  1. ... compute query set $Q^\star$. Recall from Fig. 15 that each entry in $Q^\star$ is of the form (j, z).
  2. ...
  3. $\mathcal{P}$ computes and sends $\mathbf{v}^\star := \mathbf{g}^\star(Q^\star)$ and $\boldsymbol{\pi}^\star$ where

$$\mathbf{g}^\star(Q^\star) := \left(g_j^\star(z) : (j, z) \in Q^\star\right)$$

$$\boldsymbol{\pi}^\star := \left(\mathsf{PC}_{\mathrm{agn}}.\mathsf{ProveEval}\left(\mathsf{pp}_{\mathsf{agn}}, q, c_j^\star, \mathsf{opn}_j^\star, z\right) : (j, z) \in Q^\star\right)$$

- Decision Phase: $\mathcal{V}$ accepts if and only if all the following conditions hold:
  - ...
  - ... the evaluation proofs verify including the ones from $\boldsymbol{\pi}^\star$ (using $\mathsf{PC}_{\mathrm{agn}}.\mathsf{VfyEval}$)

Fig. 16. Compiler for augmented mod-AHPs. In blue are additions to Fig. 2.

We observe that, in general, the other results we have on mod-AHPs, including those in Section H.1, extend immediately to the setting mod-AHPs augmented as we do in Fig. 15. The result we need the most among these is the following (which we use Section H).

Theorem 13 (Delayed-Input Soundness ⇒ Sparse mod-PC). *Assume: (a) a mod-PC* modPC *with weak evaluation binding; (b) a field-agnostic polynomial commitment* PC_{agn} *with negligible knowledge soundness (for dense polynomials); (c) a (possibly augmented) multilinear mod-AHP* modAHP *for* $\mathcal{R}^{poly}$ *that: (i) is efficient for sparse polynomial evaluation (Definition 22); (ii) has negligible delayed-input soundness. Then there exists a weak evaluation binding mod-PC* modPC^* *for sparse polynomials with interactive opening (see Definition 16 and Remark 3).*

Proof. The proof is almost completely the proof for Theorem 9. The fact that we have additional oracle polynomials does not change the essence of the proof above. Instead of applying the compiler from Fig. 2, we apply its extended variant in Fig. 16. The remaining observations follow mutatis mutandis. □

K More on Motivation of Native SNARKs over $\mathbb{Z}$

From a theoretical perspective, moving beyond finite fields enables broader expressivity. In fact, satisfiability of Diophantine equations (what $\mathbb{Z}$ computations are at their essence) cannot be reduced to the satisfiability of arithmetic circuits over a finite field. This is because field emulation requires the introduction of a maximum norm bound on the witness. Examples include proving the existence of integer points on elliptic curves which may have surprisingly large norms.

Having native arguments over $\mathbb{Z}$ is also important from an engineering perspective. Emulating arithmetic over the integers is commonly done through emulation in the field. The latter has a substantial effect over circuit size (a 5000× increase for RSA signature verification for example; see later in this

section). This increases the complexity of real-world proof systems in several ways. If done "by hand", this is detrimental to developer experience and can be an additional source of bugs (contrast designing a 16-wire circuit to one with 90K wires). Automatic tooling (e.g., [KPS18]) can help but they still need to be designed, developed, documented and maintained.

Other applications of SNARKs over the integers:

Range checks through "range gates" We can apply known techniques to embed range checks in our framework with a small constant number of constraints (regardless of the range size). Consider a computation over the integers that can be expressed as showing knowledge of a witness such that some arbitrary set of constraints is satisfied *plus* some of the wires are required to be in the range $\{1, \ldots, U\}$ where U is some publicly known (potentially large) upper bound. A little more formally, this setting would reduce to showing knowledge of $\vec{v} \in \mathbb{Z}^n, \vec{w} \in \mathbb{Z}^{n'}$ s.t. $C(\vec{v}||\vec{w}) \wedge \forall i \in [n]\ v_i \in [U]$ where C is some arbitrary set of constraints. The full constraint system will then consist of the constraints to check $C(\vec{v}||\vec{w})$ extended with $4n$ extra wires $\alpha_i, \beta_i, \gamma_i, \delta_i$ for $i \in [n]$, and with n extra constraints of the form:

$$\alpha_i^2 + \beta_i^2 + \gamma_i^2 + \delta_i^2 = v_i - U$$

Above we are effectively building a *range gate* through Lagrange's four squares theorem (which states that an integer is positive if and only if it can be written as the sum of four squares[32]). We observe that U might even be supplied on the fly as a public input (without having to be encoded in the R1CS). This could be achieved thanks to the homomorphic properties of our mod-PC and with some simple adaptation of our protocol in Fig. 3.

Computations on rational numbers We can represent rationals as pairs of integers in the natural way and perform computations over the rationals by introducing the following constraints:

- *addition:* $\frac{a}{b} + \frac{c}{d} \to \frac{e}{f} \implies e = ad + bc \wedge f = bd$;
- *multiplication:* $\frac{a}{b} \cdot \frac{c}{d} \to \frac{e}{f} \implies e = ac \wedge f = bd$.

If we need to check equality between two rational numbers $\frac{a}{b} \stackrel{?}{=} \frac{c}{d}$ we can simply add two constraints by letting the prover show $\alpha a = \beta c$ and $\alpha b = \beta d$ for two additional integer wires α, β. For rational arithmetic to work we also need to check that some of the quantities provided by the prover are non-zero (this is true of α and β in the equality check, as well as the denominators of "input" wires, those that are not the output of any other rational gate); we describe a non-zero gate for integers with a constant number of constraints in Section K.1.

K.1 Non-Zero Gates for Integers

Consider a wire v whose value we want to guarantee to be different from zero. Then we can add the following constraint over $\mathbb{Z}$ with additional wires $\sigma, \alpha, \beta, \gamma, \delta$:

$$\sigma v = \alpha^2 + \beta^2 + \gamma^2 + \delta^2 + 1$$

Notice that the right-hand side is guaranteed to always be positive. As a consequence, if $v = 0$ there is no way to satisfy this constraint. On the other hand, if $v \neq 0$ it is easy to find a "sign wire" σ to make the quantity σv positive; $\alpha, \beta, \gamma, \delta$ can be found through algorithms for Lagrange's four squares theorem as described in the introduction.

An alternative approach A common technique to show that a value v is non-zero in finite fields is to simply provide a non-deterministic wire ω such that $\omega v = 1$ (i.e. $\omega = v^{-1}$). Since in our family of constructions we sample a prime q at a specific point of the protocol, we can perform the same trick *after* the prime is sampled by working over the finite field $\mathbb{F}_q$. This requires a little bit of care and to slightly adapt our constructions to do the following: the prover would commit to the integer value(s) v *before* q is sampled (as usual); after sampling q the prover would send additional commitments (or oracle polynomials) to the inverse(s) ω as defined above; the protocol would then use these additional wires to guarantee the constraints of the form above (we leave this step unspecified since its concrete

[32] For a number of m bits, such a representation can be found in expectation in $\approx m \cdot \left(\frac{m}{\log m}\right)$ steps [PT18]. Notice that this is not necessarily a dominating factor: even just *computing* the witness of a standard integer computation may require quadratic time (this is the case, depending on the implementation, for standard multiplication, which in any event will require superlinear time).

description would depend on the specific of the protocol itself; similar techniques to "extend a witness" have been used before in the MLE world, e.g., in [ZGK+17, §II-C]). Formally describing this approach (which we will not do in this work) can be seen as a further application of the augmented model in Section J.

L On the Efficiency of Zinc [GWV+25]

Since Zinc [GWV+25] does not provide explicit asymptotic costs, we derived the bounds in Table 1 ourselves (we were unable to obtain clear bounds for proving time); here we justify those bounds.

Our starting point is Remark 5.3 in [GWHD25][33], which describes the number of bits read by the verifier in one of the subprotocols in Zinc. It reads:

> At Step 4 [of Protocol 9], [...] V reads the first $\log\big(\|M_{C_\lambda}\|_\infty \cdot \mathsf{dim}\big) + 2 \cdot B_{\mathbf{v}} + (6\,\mathsf{dim}+2) \cdot \log(q_0 \cdot \mathsf{dim})$ bits [from part of the transcript].

Let us now define:

$$X := \log\big(\|M_{C_\lambda}\|_\infty \cdot \mathsf{dim}\big) + 2 \cdot B_{\mathbf{v}} + (6\,\mathsf{dim} + 2) \cdot \log(q_0 \cdot \mathsf{dim}) \tag{6}$$

From the quotation above, it is clear that X provides a lower bound for both proof size and verification time. We can observe that $X \geq \mathsf{dim} \cdot \log(\mathsf{dim})$, since all the other additive and multiplicative factors in Eq. (6) are non-negative. Therefore, we can find a lower bound for X by finding a lower bound for dim.

In order to bound dim we observe that:

- $\mathsf{dim} = \sqrt{2^\mu}$, where μ is the number of variables in a committed multivariate polynomial (Section 5.2, specifically page 47);
- the parameter μ above will then be instantiated to $\mu := \log n_a + \log n_t$ in the relation described on page 43;
- the quantities n_a and n_t are explained in Section 2.2 and correspond respectively to the number of elements in the witness vector and the size of the largest integer in the witness.

Using the notation we employed in Table 1, we have $n_a = N$ and $n_t = 2^{2^\nu} = 2^m$. Putting all our observations together we conclude that

$$X \geq \mathsf{dim} \cdot \log(\mathsf{dim}) = \sqrt{2^\mu} \cdot \log(\sqrt{2^\mu}) = \Omega\left(\sqrt{2^\nu \cdot N} \cdot \log\left(2^\nu \cdot N\right)\right) = \omega\left(\sqrt{2^\nu \cdot N}\right)$$

and the bounds described in Table 1 follow[34].

[33] We mostly refer to the eprint version of Zinc for the observations in this section since several of these details are not present in the conference version of the same work.

[34] In [GWHD25, page 44] the authors point out that the costs due to n_t could be mitigated by "SOS"-type decomposability techniques from Lasso [STW24]. We first remark that, even with these techniques, the proof size and verification time in Zinc would still be exponentially larger than those in Zaratan. Moreover, it is unclear to which extent these techniques are actually applicable (or, applicable *simply*) for the case of large integers, since they might still require the order of the underlying field of the PIOP to be larger than n_t, the size of the largest elements in the witness; this may not be the case because the field will be of prime order q with $q = O(2^\lambda)$ while it could be that $\log(n_t) \approx \lambda^c$ for $c > 1$.