Fully-Succinct Arguments over the Integers from First Principles

Matteo Campanelli and Mathias Hall-Andersen

Abstract. Succinct arguments of knowledge allow an untrusted prover to establish that they know a witness for an NP relation. Many recent efficient constructions of such schemes work over arithmetic computations expressed in finite fields. Several common settings, however, have an extremely simple representation when expressed over the integers (e.g., RSA signatures/accumulators, range checks for committed values, computations over rational numbers). Efficient arguments of knowledge working natively over $\mathbb{Z}$ could be applied to such computations without the overhead from emulating integer arithmetic over a finite field.

We propose the first native construction of SNARKs over the integers that is *fully* succinct, thus resolving an open problem from Towa and Vergnaud (Asiacrypt 2020). By fully succinct, we mean that *both* the proof size and the verifier's running time should be sublinear in *both* $|\vec{w}|$—the size of the witness as a vector of integers—and $\log_2\|\vec{w}\|_\infty$—the size in bits of the largest integer in the witness vector (in absolute value).

As a stepping stone for our results we provide a general theoretical framework for building succinct arguments over the integers. Its most attractive feature is that it allows to reuse already existing constructions of SNARKs in a modular way and can be used as a starting point for constructions following up our work.

We build these systematic foundations by leveraging a common technique in theoretical computer science—*fingerprinting*—and applying it to a new setting. Our framework consists of two main ingredients: idealized protocols and polynomial commitments such that an object "committed over the integers" can however be "queried modulo q", for a randomly sampled prime q.

We obtain our final construction, $\mathbb{Z}$aratan, by lifting the Spartan construction (Setty, CRYPTO 2020) to the integers and applying a form of polynomial commitment based on the techniques from DARK (Bünz et al., Eurocrypt 2020). $\mathbb{Z}$aratan has a transparent setup, is proven secure in the generic group model for groups of unknown order and can be heuristically made non-interactive in the ROM via the Fiat-Shamir transform.

Table of Contents

1 Introduction

A succinct argument of knowledge—and its non-interactive counterpart, dubbed SNARK [29]—is a cryptographic protocol that allows an untrusted prover $\mathcal{P}$ to convince a verifier $\mathcal{V}$ that they possess a witness w satisfying a certain predicate. The most straightforward form of proof of this type would have $\mathcal{P}$ sending directly w to $\mathcal{V}$, who could then independently verify the required predicate. A succinct argument accomplishes the same goal but with greater efficiency. In particular, "succinct" means that the proof is shorter than the witness w itself, and verifying the proof should be significantly faster than checking it directly.

Arguments Over the Integers. Most of the current efficient constructions for succinct arguments are designed for computations *over finite fields* (e.g., arithmetic circuits), usually of prime order. Intuitive this is because finite fields offer various basic tools for a cryptographic design. Some of these tools are cryptographic primitives we can leverage (e.g., homomorphic commitments based on elliptic curves where DLOG holds and with efficient arithmetic), but others are even more foundational for blueprints behind SNARK protocols, most notably the Schwartz-Zippel lemma. For these reasons, it has been relatively challenging to propose arguments that can *natively* prove computations over more general algebraic structures, such as the ring of integers, $\mathbb{Z}$ where these tools are not available. This is the focus of our work.

Several computations[1] can naturally be expressed over $\mathbb{Z}$. These include computations for applications such as batching RSA signatures, RSA accumulators [4], cryptography based on rings, range checks(see also discussion in [34]). In finite fields, such applications usually incur at least the price of splitting a long integer in limbs of $\approx \lambda$ size and performing extra checks. For a native proof system for $\mathbb{Z}$, divisibility of integers, for example, may consists in contrast of a single gate. Besides the efficiency gain, this also simplifies the design of the circuit, reducing the probability of bugs.

Succinctness for computations over $\mathbb{Z}$. Our goal in this work is to propose *succinct* arguments for integer computations. A standard definition of succinctness is that the proof size should be sublinear in the dimension of the witness $N = |\vec{w}|$ (for example, the number of wires in a satisfying assignment). This notion is at times strengthened by requiring that the verification time is also sublinear in this parameter. Adapting these notions for the case of integer computations, requires care. In the previous setting all the elements of the witness are scalars in a finite field, each of a bit size that is usually linear in the security parameter λ. On the other hand the integers in the witness may have a bit representation significantly larger than λ. Arguably, for an argument to be actually succinct, it should be succinct in both N and m where $m = \log_2 \|\vec{w}\|_\infty$ is the bit size of the norm of the witness vector[2]. We refer to this notion as *full succinctness*.

As the only other work addressing succinctness of SNARKs over $\mathbb{Z}$, we mention the previous work by Towa and Vergnaud [34] who proposed a zero-knowledge argument for integer computations whose design is loosely inspired by Bulletproofs [8]. Their construction has a proof size that is logarithmic in N but linear in m and a verifier running linearly N and therefore not fully succinct. Their work left the open question of whether it is possible to build a fully-succinct SNARK for $\mathbb{Z}$, which we answer in the positive in this paper.

This Work. In this paper we initiate the study of SNARKs whose underlying computational model is defined over the ring of integers and that are fully-succinct. We identify a general paradigm through which to construct efficient SNARKs over $\mathbb{Z}$, introduce key abstractions and show how they can be used to obtain a concrete efficient SNARKs over the integers. More in detail, our contributions are:

- *The first fully-succinct construction for arguments natively over the integers*, $\mathbb{Z}$aratan[3]. Our construction supports R1CS over the integers (Definition 5) and can be instantiated through class groups in order to have a transparent setup. A summary of its efficiency properties are in Table 1.

[1] For now the reader can think of a computation over the integers as an arithmetic circuit with multiplication and addition gates (over $\mathbb{Z}$) and where wires are integers of (known) bounded size growing polynomially in some parameter. By "proving a computation" we mean proving knowledge of a wire assignment satisfying the circuit.

[2] Notice that an alternative definition, one requiring sublinearity in the total size of the witness, $N \cdot m$, may not be the right one since m may be large but N may not even in a useful computation (consider, for example, a circuit with relatively few gates performing arithmetic over huge integers).

[3] Zaratan is a mythological sea turtle, allegedly of incredible size, featured in the work of Jorge Luis Borges [7].

Scheme	Proof Size	Prover	Verifier
Our mod-PC	$O_\lambda(\nu + \log^2 N)$	$O_\lambda(m \cdot N)$	$O_\lambda\left(\max\{\nu \cdot \log N, \log^2 N\}\right)$
$\mathbb{Z}$aratan	$O_\lambda(\nu + \log^2 N)$	$O_\lambda(m \cdot N)$	$O_\lambda\left(\max\{\nu \cdot \log N, \log^2 N\}\right)$

Table 1: Efficiency summary for our constructions. 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 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 $m \cdot N$ is essentially the size of the witness/polynomial, and the prover's running time is 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.

- *A general methodology for building arguments for integers.* Our framework partly mirrors the common approach based on *i) idealized protocols* (AHP/PIOP/PHP[4]) and *ii) polynomial commitments* [26] to build SNARKs. We introduce respective analogues for these primitives on which we elaborate in the technical overview. For now, the reader can think of them as protocols allowing some form of commitment to a polynomial over $\mathbb{Z}$ but that provide soundness guarantees even when the evaluation queries to the polynomials do not return the actual integer output but some "fingerprint" (see technical overview). We provide a compiler that constructs a knowledge-sound argument over the integers from our new building blocks. (see also Fig. 1).
- *General results to instantiate the idealized protocols from existing proof systems* (even if they are not designed to be over the integers). In particular we identify a very "weak" form of extractability property that our AHP-like protocols (item *(ii)* above) need to satisfy in order to be plugged into our compiler (see top-left side of Fig. 1); most importantly this property does not require to be able to extract a witness. This property will make extremely easy to prove and identify Spartan [31] as a building block for our final construction.
- *A new lens over DARK techniques.* We show how techniques from DARK ([10] and [5]) can be used to construct a polynomial commitment of the flavor we introduce. Thanks to our compiler, this construction can be used a stepping stone in other arguments over the integers other than $\mathbb{Z}$aratan.

1.1 Technical Overview

Our Approach in a Nutshell. Our whole approach can be summarized as follows: we identify the technique of *fingerprinting through a random prime* (see, e.g., [2, §7.2.3]) as a tool to map a relation over the integers to one over a finite field; we then observe that *some* existing SNARK constructions have features that make them easy to pair with fingerprinting. The bulk of our technical challenges consisted in identifying formal properties and efficient building blocks in order to carry out this plan.

Let us briefly recall what fingerprinting[5] consists in through a standard example. Consider a multivariate polynomial g over $\mathbb{Z}$ expressed as a polynomial-sized circuit and imagine to want to test whether it is identically zero. A first line of attack exploits Schwartz-Zippel for a simple probabilistic check: sample random inputs for g and see if the result of the evaluation y is zero. If the sampling domain is large enough relative to the total degree of g, we can be fairly confident of the polynomial being zero or not from the result of the test. But there is a catch: since g is expressed as a circuit its concrete degree may be exponential, making y and the intermediate results of the evaluation simply too large. Fingerprinting comes to the rescue though: sample a (large enough) integer n and carry out the evaluation of g over $\mathbb{Z}_n$ instead of over the integers.

Let us now move back to our focus in this work: succinct arguments for non-deterministic relations over the integers. How can we leverage the technique above to our advantage? A first intuitive attempt

[4] Algebraic Holographic Proofs [16], Polynomial Interactive Oracle Proofs [10] and Polynomial Holographic IOPs [11].

[5] In this work we consider a specific type of fingerprinting: sampling a random number n (specifically a prime) to then reduce a "problem over large objects" to a problem on "objects modulo n". There are other types of fingerprinting techniques in literature, but we will always implicitly assume those with this specific flavor.

for a blueprint for succinct arguments over $\mathbb{Z}$ would be the following. On input a computation[6] $\mathcal{C}$ over $\mathbb{Z}$:

1. Let the prover commit to the witness w over the integers.
2. *Appy fingerprinting:* sample a large number q and consider the "reduced computation mod q" (which we denote by $[\![\mathcal{C}]\!]_q$);
3. Use a succinct argument for computations over $\mathbb{Z}_q$ to verify $[\![\mathcal{C}]\!]_q$.[7].

We are far from done since the sketch above leaves several questions unanswered. *(i)* What are arguments we can apply in step (3.) and what properties should we require from them? *(ii)* What does it precisely mean to reduce a computation "mod q"? *(iii)* If the witness is committed over $\mathbb{Z}$, how can we efficiently switch to something modulo q afterwards (which would probably be required by the argument for $[\![\mathcal{C}]\!]_q$)? The bulk of our technical contributions consists in providing formally rigorous answers to these questions and making design choices that would make our formal treatment as general as possible.

We now provide some intuitions about how we address questions *(i)* and *(ii)*; we will provide additional details throughout this technical overview.

First we turn to recent efficient constructions of succinct arguments and we notice that several of them work over a finite field of prime order $\mathbb{F}_q$,. This suggest that sampling a *prime* q might be the right approach above (rather than sampling an arbitrary integer). But we need even more properties from the argument at step (3.): in particular the latter should be able to "work effectively" even if the order q of the field is not known at the beginning of the protocol. For the sake of this technical overview, we call such constructions "fingerprinting-friendly" [8]. Examples in this sense are constructions mainly based on multivariate techniques such as Spartan [31] and HyperPLONK [15]. Here, one can, in principle, compute and commit encodings of the relations and the witness (through their multilinear extensions (MLE); see Section 2) over the integers and only later run the protocol over $\mathbb{F}_q$. This is the case because MLEs preserve their key properties over $\mathbb{Z}$.

Let us now address question *(ii)*: how to go from a relation over the integers to one over $\mathbb{Z}_q$ and when would that preserve soundness? The first part is easy to answer. Consider a computation over the integers expressed as a Rank-1 Constraint Satisfiability (R1CS). After the prover has committed to a witness w, instead of checking each constraint through the equation $\langle \vec{a}, \vec{z} \rangle \circ \langle \vec{b}, \vec{z} \rangle - \langle \vec{c}, \vec{z} \rangle = 0$ over the integers, we check instead $\langle \vec{a}, \vec{z} \rangle \circ \langle \vec{b}, \vec{z} \rangle - \langle \vec{c}, \vec{z} \rangle \equiv 0 \mod q$. What we are checking now is satisfiability of a standard R1CS over $\mathbb{F}_q$ for which we can use a fingerprinting-friendly argument!

The previous observation gives us correctness, but we also need to argue why we are not losing soundness when in approach above. One intuition for the case of R1CS over the integers is this: each of the constraint equations is testing whether a polynomial evaluated in $\vec{z}$ is zero. What is crucial is that this polynomial is of relatively low degree (an R1CS encodes a quadratic polynomial) and hence we can apply arguments similar to the ones we use to show the soundness of the standard fingerprinting-based approaches for zero-testing of polynomials. We stress that our core framework will not be limited to R1CS and we will provide general sufficient conditions for computations to be "fingerprinting-friendly" (through the notion of "good test" defined in Definition 4; the reader can see a formal version of the proof we just sketched in the proof of Lemma 2 in the appendix).

In the remainder of this technical overview we describe our general framework, how we instantiate its building blocks and how we apply them to obtain our final construction, Zaratan.

[6] In this part, we vaguely refer to a "computation" over $\mathbb{Z}$. For concreteness, the reader can think of the computation being expressed as a Rank-1 Constraint Satisfiability (R1CS) with integer coefficients. An R1CS is described by matrices A, B, C; here we aim at showing knowledge of $\vec{z} \in \mathbb{Z}^N$ (each element of bounded size, albeit potentially large), s.t. $\langle \vec{a}, \vec{z} \rangle \circ \langle \vec{b}, \vec{z} \rangle - \langle \vec{c}, \vec{z} \rangle = 0$ for each row $\vec{a}, \vec{b}, \vec{c}$ in the respective matrices.

[7] We warn the reader that this blueprint is for didactic purposes only. Later, when presenting our framework, we will slightly deviate from some of the intuitions we used in this part of the overview. For example, while we mentioned committing to an integer witness as a first step of the blueprint, we will never do that (explicitly) in our framework. In any event, our formalism still morally captures the same principles and intuitions we are providing on fingerprinting-friendliness in this part of the text.

[8] Examples of constructions that are *not* fingerprinting-friendly include those such as the original PLONK [21]. These may require the field to have additional properties, e.g., being DLOG- and FFT-friendly, or having one or more additive/multiplicative subgroups of predetermined sizes. We refer the reader to the excellent discussion in a context other than fingerprinting in [24].

Our General Framework: mod-AHP + mod-PC ⇒ SNARKs for $\mathbb{Z}$. We now describe the general ideas behind our framework. 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) [16]. 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 g with an evaluation point z to obtain $v = g(z)$. V can iterate this process for several different polynomials and evaluation points (all arithmetic being performed in $\mathbb{F}$). Finally, V outputs a 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) [26]: the prover can commit to the oracle polynomials and later, upon receiving an evaluation point z, can send an evaluation proof to convince the verifier that evaluation $v = g(z)$ 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 z for a polynomial g but will be constrained to obtain only $g(z) \mod q$. 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*[9] 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.

Intermezzo: let us talk about succinctness. Recall that our goal is to obtain a proof and verifier succinct in both $N = |\vec{w}|$ and $m = \log\|\vec{w}\|_\infty$ where $\vec{w}$ is a witness. Our succinctness in N is, in a sense, directly inherited from the AHP+PC approach where a few polynomial evaluations (with adequately succinct proofs) "guarantee knowledge" of a witness of size N. While the case of succinctness in m is a little different, we are able to anticipate why we would be able to achieve it: all our evaluations in this approach are modulo q, a prime of λ bits (where λ is 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. Now that we defined what the requirements for our building blocks are, let us discuss how to instantiate them.

From DARK to mod-PC constructions. We use techniques from the DARK compiler [10] to construct mod-PCs. The original construction of Bünz et al. [10] is not directly applicable to our setting, because the binding notion is too weak. Instead we rely on a protocol by Block et. al [5] which allows extraction of an integer polynomial. To reduce the verifier computation of this protocol we compose it with an Argument-of-Knowledge for the language of accepting last round messages. 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.

[9] 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).

"Weak" Fingerprinting-Friendly Constructions Suffice. 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. While our compiler requires what we earlier called "full" knowledge soundness (Full KSND)—where we should be able to extract an *integer* witness—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[10] 13, we need to be able to only 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. $\mathsf{w}_q^\star \equiv 42^{150} \pmod{13}$.

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 [31], 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 [23] 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 [31] can be leveraged for the case of $\tilde{w}$ as a polynomial over the integers.

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 [13]). 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[11]. Our approach works by showing that the SPARK compiler in [31] (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 soundess. 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.

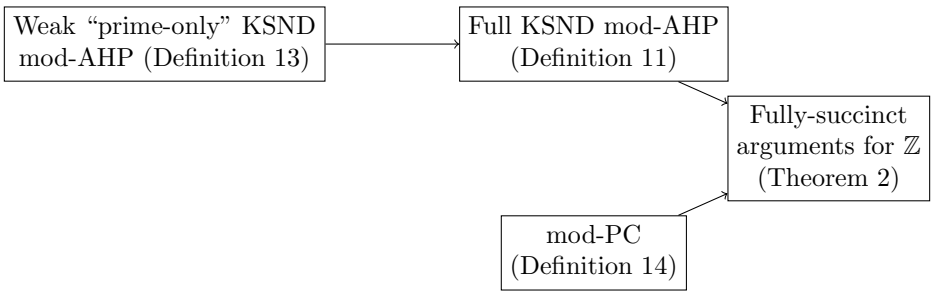

Fig. 1: Relationships among some of our core abstractions.

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

[11] 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.

1.2 Related Work

Poly. Comm from Groups of Unknown Order. Several works, starting with Bünz et. al [10] [9] and Block [5] 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.

Arguments for Rings. Prior works (e.g. [22]) has constructed SNARKs for computations over finite rings. Towa and Vergnaud [34] constructed a Bulletproof [8] inspired argument for Diophantine equations over the integers. In [32] Soria-Vazquez constructs interactive proofs for *deterministic computations* expressed over infinite and non-commutative rings. Our works differs in several respects. Our focus are *non-deterministic computations* over the ring of integers. While the rings considered in [32] are more general, the constructions proposed in it do not apply to the non-deterministic case.

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. 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 (i.e., entry-wise) product between vectors. We write $\vec{0}$ to denote vector with entries equal to the 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 $\|\vec{v}\|_\infty$, where $\vec{v}$ a vector of integers, its infinity norm, i.e. the quantity $\max_i |v_i|$. We sometimes abuse notation and, given a polynomial f we denote by $\|f\|_\infty$ the infinity norm of its vector of coefficients.

Prime Sampling We denote by $\mathbb{P}_\lambda$ the set of primes of λ bits, i.e. in the interval $[2^{\lambda-1}, 2^\lambda)$. We use random prime sampling as done in previous works, e.g. [36,10,4]. By the prime number theorem it is easy to show that $|\mathbb{P}_\lambda| = \Theta\left(\frac{2^\lambda}{\lambda}\right)$.

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 circuits itself. We say that a relation is *deterministic* is it a set of pairs index–instance rather than triples (or equivalently if w is always $\perp$). For any indexed relation we will 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 function on the boolean hypercube. We refer the reader to [33] for a broader discussion of multilinear extensions.

Definition 1. *Let $f : \{0,1\}^t \to \mathbb{Z}$ be a function. The multilinear extension of f (which we denote by $\mathsf{MLE}(f)$ or $\tilde{f}$) is the unique polynomial $\tilde{f} : \mathbb{Z}^t \to \mathbb{Z}$ of individual degree one and such that for all $\vec{x} \in \{0,1\}^t$ $\tilde{f}(\vec{x}) = f(\vec{x})$. This polynomial can be constructed as*

$$\tilde{f}(\vec{X}) = \sum_{\vec{b} \in \{0,1\}^t} \chi_{\vec{b}}\left(\vec{X}\right) \cdot f(\vec{b})$$

where $\chi_{\vec{b}}(\vec{s}) = \prod_{i=1}^t \left(s_i b_i + (1-s_i)(1-b_i)\right)$ 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 $\mathsf{MLE}(\vec{v})$ to denote $\mathsf{MLE}(f)$.

We will use this result whose proof is immediate from the construction of $\mathsf{MLE}(f)$. It essentially states that, for a function f, the MLE of f evaluated modulo q "matches" the evaluation modulo q of $\mathsf{MLE}(f)$.

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

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

Dense and Sparse MLEs We recall some of the observations on sparse/dense representations of MLEs from [31]. A multilinear polynomial $g : \mathbb{Z}^\mu \to \mathbb{Z}$ can be represented uniquely by the list of evaluations of g over the boolean hypercube $\{0,1\}^\mu$. We denote this representation as the *dense* representation of g, or $\mathsf{DenseRepr}(g)$. It is easy to show that if $g(x)$ is zero on any point x of the hypercube, this does not need to be included in $\mathsf{DenseRepr}(g)$.

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 [16]). *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 an arbitrary string aux. The security parameter is passed implicitly to both parties as a unary string. We denote by $\mathsf{tr} \leftarrow \mathsf{transcript}_\lambda\big(\langle \mathsf{P}_\mathrm{ip}(\mathsf{aux}), \mathsf{V}_\mathrm{ip}\rangle\big)$ 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 a 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 test). *An associated fingerprinting relation for $\mathcal{R}$ is said to provide a good 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:*

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

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*

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

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 that 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 $\|\mathsf{x}||\mathsf{w}\|_\infty \leq 2^{b(N)}$ for a fixed bound function b implicitly parametrizing the relation[12]. 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 defined above is admissible. It also provides a good test (the proof is in the appendix).

Lemma 2. *The associated fingerprinting relation for integer R1CS in Definition 7 provides a good test.*

[12] We require this bound function to be polynomial in λ.

3.3 Multilinear Polynomial Evaluation and its Fingerprint

We also define another indexed relation that will be useful for our results in Section 8—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 variable, a statement consisting of a pair $(\vec{x} \in \mathbb{Z}^{\mu}, y \in \mathbb{Z})$. It returns 1 if and only if $f(\vec{x}) = y$. The size of the index described by a multilinear polynomial f is its maximum number of non-zero coefficients, i.e., 2^{μ}.*

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 [16,11,10] 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 denotes the number of variables in the multivariate oracle polynomials[13]*; s denotes the number of polynomials in each round; d specifies degree bounds (in each variable) on these polynomials.*
The protocol proceeds as follows:

- ***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 $\mathcal{I}(1^{\lambda}, \mathsf{i})$*[14]*. 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.*

[13] 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.

- ***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* $(1^\lambda, \mathsf{x}, \mathbf{p}(Q); \vec{\rho}_1, \ldots, \vec{\rho}_\mathsf{k}, \mathsf{tr}_{rst}, q)$*.*

The function d *determines what kind of provers are considered for the completeness and soundness properties of the proof system. 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 11) 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.)*

4.2 Weak Knowledge Sound mod-AHPs

We now 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 6. Together with the result of Theorem 2 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 in Section 7.

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*:*

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

$\Big(g^*(\vec{X}), q, \dots\Big) \leftarrow \mathsf{transcript}\Big(\langle \mathsf{P}^*(1^\lambda, \mathsf{i}, \mathsf{x}, \mathsf{aux}), \mathsf{V}^{\mathcal{I}(1^\lambda,\mathsf{i})}(1^\lambda, \mathsf{x})\rangle\Big)$ *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 test (Definition 4). By applying Lemma 2 we can interpret this as a lifting theorem for weak knowledge sound mod-AHPs for R1CS.

Theorem 1. *Let* modAHP *be a mod-AHP with negligible weak knowledge soundess error over $\mathcal{R}$ and $[\![\cdot]\!]$ (Definition 13). If $[\![\cdot]\!]$ provides a good 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 construct a form of polynomial commitment that can be use to compile a mod-AHP into an argument.

5.1 Model

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

- $\mathsf{Setup}(1^\lambda, D, \mu) \rightarrow \mathsf{pp}$*: on input a security parameter $\lambda \in \mathbb{N}$, an individual degree parameter $D \in \mathbb{N}$ and a number of variables $M \in \mathbb{N}$ outputs public parameters of the scheme.*
- $\mathsf{Com}(\mathsf{pp}, g \in \mathbb{Z}_{\leq d}[X_1, \dots, X_\mu]) \rightarrow (c, \mathsf{opn})$*: on input public parameters, a polynomial over the integers g, it outputs a commitment c and an additional opening string* opn *(used as auxiliary input for opening).*
- $\mathsf{ProveEvalMod}(\mathsf{pp}, q, c, \mathsf{opn}, z) \rightarrow \pi^{(eval)}$*: on input public parameters* pp*, prime q, commitment c, opening* opn *and $z \in \mathbb{Z}$, it outputs a proof $\pi^{(eval)}$ certifying the value $g(z) \mod q$.*
- $\mathsf{VfyEvalMod}(\mathsf{pp}, q, c, z, y, \pi^{(eval)}) \rightarrow b \in \{0, 1\}$*: on input pubic parameters, prime q, commitment c, claimed value $y \in \mathbb{Z}_q$ and proof $\pi^{(eval)}$, it outputs a bit accepting or rejecting the proof.*

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

$$\Pr\left[\mathsf{VfyEvalMod}\left(\mathsf{pp}, q, c, y_q, \pi^{(\text{eval})}\right) = 1 \quad : \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, D, M) \\ (c, \mathsf{opn}) \leftarrow \mathsf{Com}(\mathsf{pp}, g,) \\ \pi^{(\text{eval})} \leftarrow \mathsf{ProveEvalMod}(\mathsf{pp}, q, c, \mathsf{opn}, z) \\ y_q := g(z) \mod q \end{array}\right]$$

Weak evaluation binding. The following property is the analogue for mod-PCs of weak evaluation binding for functional commitments (see, e.g., [14,12]). 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 the same flavor as the one in [16].

Definition 15. *For any $\lambda, D, M \in \mathbb{N}$ and PPT $\mathcal{A} = (\mathcal{A}_{com}, \mathcal{A}_{prf})$ 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:*

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

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

Remark 2 (Interactive opening). All our definitions in this sections 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 4.

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* $\mathsf{ProveEvalMod}$ *stage is linear in $\ell \cdot \log_2 \|g\|_\infty$ where ℓ is the number of non-zero coefficients of the dense representation of g.*

5.2 A mod-PC from the DARK Side

Our construction of mod-PC derives from the techniques originally proposed by Bünz et. al [10] which construct polynomial commitments for polynomials $f \in \mathbb{F}_p[\vec{X}]$ from groups of unknown order. The scheme by Bünz et. al 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 therefore commitments to integer polynomials it is natural starting point for application.

DARK Commitment. To commit to a multi-linear integer polynomial $f(\vec{X}) \in \mathbb{Z}[\vec{X}]$, it is evaluated a point $(\mathsf{q}_1, \ldots, \mathsf{q}_k) \in \mathbb{Z}^k$ where the q_i's are sufficiently large compared to the norm of the coefficients of $f(\vec{X})$ to ensure that the evaluation uniquely determines the polynomial: letting $\mathsf{q}_i = \mathsf{q}^{2^i}$ for a sufficient 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 [20] [18] to this evaluation: $\mathsf{modPC.Com}(\mathsf{pp}, f(\vec{X})) = [f(\mathsf{q}_1, \ldots, \mathsf{q}_k)] \cdot G \in \mathbb{G}_N$ in a group of unknown order $\mathbb{G}_N$. Furthermore, this commitment is (bounded) linearly homomorphic: $\mathsf{modPC.Com}(\mathsf{pp}, f_0(\vec{X})) + [v] \cdot \mathsf{modPC.Com}(\mathsf{pp}, f_1(\vec{X}) = \mathsf{modPC.Com}(\mathsf{pp}, f_0(\vec{X}) + v \cdot f_1(\vec{X}))$ assuming the q_i's are large enough relative to the coefficients on the polynomial $f_0(\vec{X}) + v \cdot f_1(\vec{X}) \in \mathbb{Z}[\vec{X}]$.

Soundness of DARK. The original DARK paper shows how to construct an interactive protocol for proving openings of the commitments above, however the soundness proof of the protocol in the Eurocrypt DARK paper [10] had a significant flaw which was subsequently uncovered by Block et. al [5]. A new preprint of the DARK paper [9] was posted to remedy this soundness gap in the original DARK construction, however the updated proof shows a weaker notion of binding, which suffices for constructing polynomial commitments over the field, but turns out to be is insufficient for our application. We explore this now. The DARK extractor recovers a rational function from the set:

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

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: $g(\vec{X}) \coloneqq f(\vec{X}) \cdot N^{-1} \mod q$. However an issue arises if we attempt to extend the same idea to recovering integer witnesses, because there exists relations satisfied by such 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 for *integer polynomials* we cannot hope to argue about the existence of integer witnesses from the satisfiability of relations over random prime fields.

Why Rational Functions. The weaker binding notion in the DARK paper stems from the extractor having to "divide" by a difference $\alpha_1 - \alpha_2$ of challenges where $\alpha_1, \alpha_2 \in \mathbb{Z}$. Since the division of an integer polynomial by an integer is not (generally) possible, the product N of the challenge differences are moved to the "other side" of the commitment verification equation, verifying: $N \cdot C = \mathsf{Enc}(f) \cdot G \in \mathbb{G}_N$. The result is a scheme which can only be proven to satisfy binding for rational functions of the form outlined above, although a concrete attack has not been exhibited.

Construction for Integer Polynomials. Luckily for our application, the paper of Block et. al [5] proposed an alternative construction which allows the extractor to recover an *integer polynomial with coefficients of bounded norm*. Their scheme circumvents the issue by using matrixes with $\{0, 1\}$ entries as challenges (rather than integers) which are invertible over the integer ring with overwhelming probability. The drawback is that the communication complexity of the resulting scheme is greater by a factor proportional to the statistical security parameters. The only difference between the scheme of Block et. al [5] compared to the original by Bünz et. al is the evaluation proof – the commitment procedure remains the same. We outline their protocol (with our notation) in Appendix C.

Dealing With Large Norm Witnesses. The final round of the protcol by Block et. al (see Appendix C) has the prover sending a vector $Z \in \mathbb{Z}^\lambda$ of openings with $\|Z\|_\infty \leq 2^m \cdot (2\lambda)^k$ where $m = \lceil \log_2 \|f\|_\infty \rceil$ of the original multilinear integer polynomial $f \in \mathbb{Z}[X_1, \ldots, X_k]$. As a result the verification depends linearly on m and is not poly-logarithmic in the size of the witness which might have a large norm. To reduce the computation of $\Sigma_{\mathsf{MultiEval}}$ we observe that having the prover send $Z \in \mathbb{Z}^\lambda$ to the verifier is a trivial Proof-of-Knowledge for the relation of accepting last round messages:

$$\mathcal{R}_{\mathsf{final}} \coloneqq \left\{ \left(\begin{pmatrix} \mathbf{C} = (C_{(Z_1)}, \ldots, C_{(Z_\lambda)}) \\ \vec{y} = (y_1, \ldots, y_\lambda) \end{pmatrix}, Z \right) : \begin{matrix} \forall i. y_i \equiv Z_i \bmod q \\ \forall i. C_{(Z_i)} = [Z_i] \cdot G \\ \forall i. \|Z_i\|_\infty \leq \mathsf{bound} \end{matrix} \right\} \tag{1}$$

We replace this simple Proof-of-Knowledge with a more efficient Argument-of-Knowledge Σ_{Final} (see Fig. 11). The inuition is to apply the extractor for Σ_{Final} to recover a transcript for $\Sigma_{\mathsf{MultiEval}}$, then apply the existing extractor for $\Sigma_{\mathsf{MultiEval}}$ to recover the witness. We prove that the scheme mod-PC satisfies the Weak Evaluation Binding (Definition 22) and Weak Knowledge Soundness (Definition 13) in Appendix G.

6 A Compiler from Algebraic Holographic Proofs with Modular Remainder Queries to SNARKs over $\mathbb{Z}$

At the high level the compiler follows the blueprint of the standard compilers in this space [16,11,1]: we use a polynomial commitment to commit to each of the oracle polynomials; at the end of the interaction

$\mathsf{Com}(\mathsf{pp}, f)$

// Compute the coefficient size and the evaluation points:

1 : $m = \lceil \log_2(\|f\|_\infty) \rceil$

2 : $\mathsf{q} = 2^{m \cdot k \cdot \mathsf{poly}(\lambda)} + 1 \in \mathbb{Z}$

3 : **for** $i \in [1, \ldots, k] : \mathsf{q}_i = \mathsf{q}^{2^{i-1}}$

// Evaluate f at $\mathsf{q}_1, \ldots, \mathsf{q}_k$ over $\mathbb{Z}$

4 : $C = [f(\mathsf{q}_1, \ldots, \mathsf{q}_k)] \cdot G \in \mathbb{G}_N$

5 : **return** (m, C)

$\mathsf{ProveEvalMod}(\mathsf{pp}, q \in \mathbb{P}, c = (m \in \mathbb{N}, C \in \mathbb{G}_N), \mathsf{opn} = f, \vec{x} \in \mathbb{F}_q^k,)$

1 : $y = f(\vec{x})$

2 : Run $\Sigma_{\mathsf{Open}}(\mathsf{pp}, C, \vec{x}, y, m, q; f)$

$\mathsf{VfyEvalMod}(\mathsf{pp}, q, c = (m, C), \vec{x}, y, \pi^{(\text{eval})})$

1 : Run $\Sigma_{\mathsf{Open}}(\mathsf{pp}, C, \vec{x}, y, m, q; f)$

Fig. 2: Our construction of modPC. The setup is simply $\mathsf{pp} = (\mathbb{G}_N, G)$ where $\mathbb{G}_N \leftarrow \mathcal{G}(1^\lambda)$ and $G \leftarrow\!\!\$\ \mathbb{G}_N$. The polynomial $\mathsf{poly}(\lambda)$ is derived from the knowledge soundness parameter of the opening proof (see Appendix C for details).

the oracle queries are provided by the prover and proven through the polynomial commitment opening. The key difference beween the works cited above and this paper is that we have integer-flavored primitives and that all the queries happen after (and on the basis of) a prime that is sampled at an intermediate point of the interaction.

Theorem 2. *Let* modAHP *be a 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. 3 is an interactive argument with preprocessing (Definition 2) for $\mathcal{R}$.*

It is possible to reduce the requirements for the commitments for the indexing polynomials as we discuss in Appendix H.

7 $\mathbb{Z}$aratan: Efficient Spartan over the Integers

7.1 Background on Spartan

In this section we review Spartan [31], 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[15]. 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 $\mathbb{F}$:

[15] **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.

Setup$(1^\lambda, N)$**:** Let $D := \max\{\mathsf{d}(\lambda, N, i, j) \mid i \in [\mathsf{k}(N)], j \in [\mathsf{s}(i)]\}$. Let $M := \max\{\mathsf{v}(\lambda, N, i) \mid i \in \{0, 1, \ldots, \mathsf{k}(N)\}\}$. $\mathsf{pp}_{\mathsf{PC}} \leftarrow \mathsf{PC}_{\mathrm{prj}}.\mathsf{Setup}(1^\lambda, D, M)$ 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
  $$\mathbf{g}(Q) := \left(g_{i,j}(z) \bmod q : ((i, j), z) \in Q\right)$$
  $$\boldsymbol{\pi} := \left(\mathsf{PC}_{\mathrm{prj}}.\mathsf{ProveEvalMod}\left(\mathsf{pp}_{\mathsf{PC}}, q, c_{i,j}, \mathsf{opn}_{i,j}, z\right) : ((i, j), z) \in Q\right)$$
- 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. $\mathsf{PC}_{\mathrm{prj}}.\mathsf{VfyEvalMod}\left(\mathsf{pp}_{\mathsf{PC}}, q, c_{i,j}, v, \pi\right) = 1$ for each $((i, j), z)$ in Q and corresponding value v in $\mathbf{v}$ and proof π in $\boldsymbol{\pi}$.

Fig. 3: Compiler from 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.

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 (Appendix 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.
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.

7.2 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. 4. Compared to the presentation above we made a few changes, some to remain close to the original treatment in [31] (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 [31] for details. To maintain visual similarity with [31] we do not use the notation $\vec{v}$ for vectors in most of the figure.

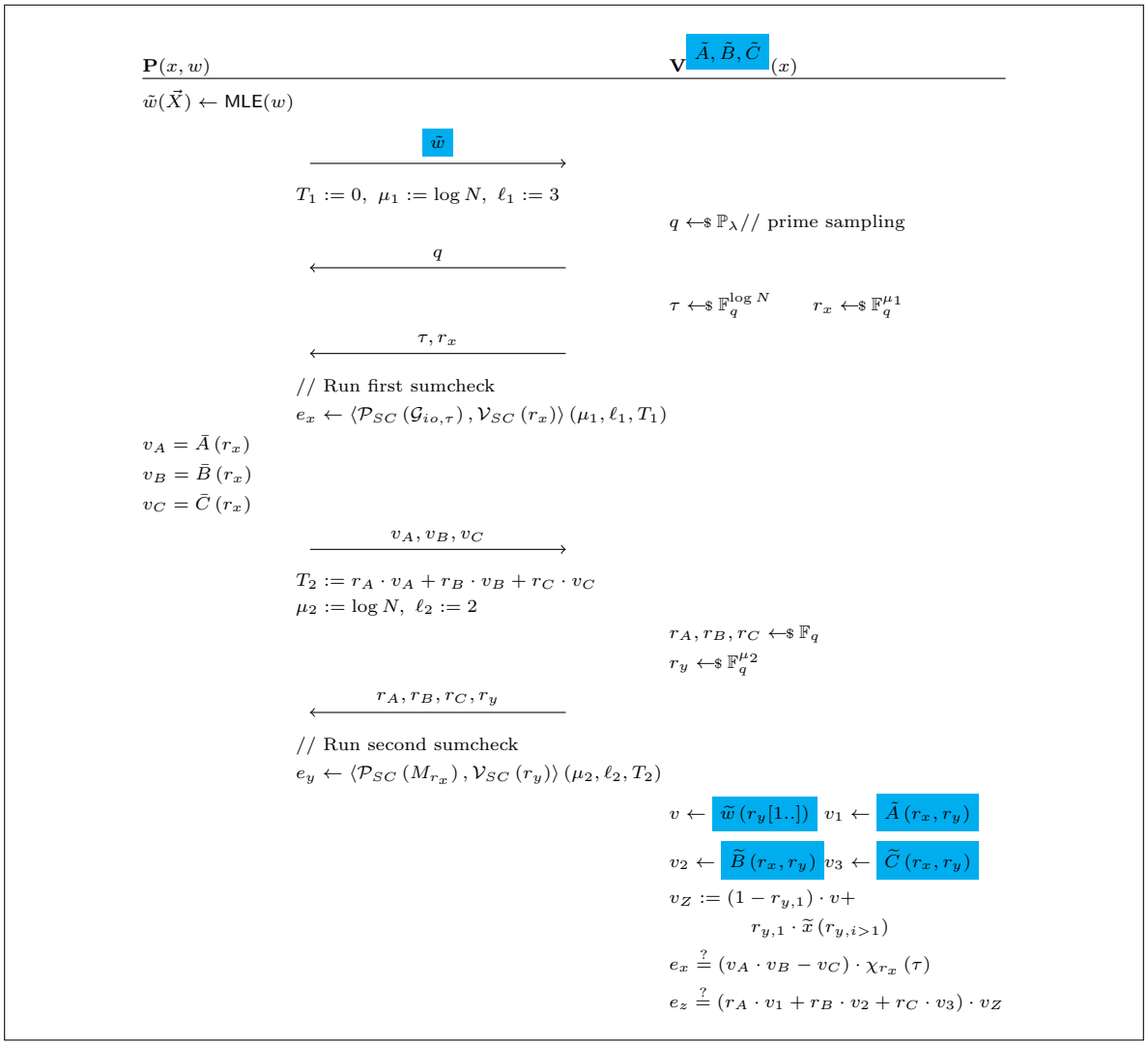

Fig. 4: Spartan as as an AHP over $\mathbb{Z}$ with modular remainder queries. The parameter N denotes the (known) size of the witness. In cyan we denote the integer oracle polynomials and queries to them. Notice that $\tilde{w}, \tilde{A}, \tilde{B}, \tilde{C}$ are all oracle polynomials over the integers. 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 $r_{y,i>1} := (r_{y,2}, \dots, r_{y,\mu_2})$. The figure above simply consists of the online stage; indexing simply returns $\tilde{A}, \tilde{B}, \tilde{C}$.

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

7.3 Putting it All Together: ℤaratan

Our final construction ℤaratan is the result of compiling Fig. 4 with Fig. 3. We instantiate the polynomial commitment for $\tilde{w}$ through our construction in Section 5.2. In order to have efficient polynomial

commitments for the indexing polynomials we use a construction for sparse polynomials we give in Section 8.2. 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. The final efficiency is in Table 1. We stress that the prover has a linear dependency (not quasi-linear) w.r.t. m.

8 Building mod-PCs for Sparse Polynomials

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.

8.1 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 this form of soundness is "delayed-input" is because we want its security to holds even if 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 17. *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 18. *Consider a mod-AHP for deterministic relations only (Definition 17) 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 test as in Theorem 11.

Remark 3. 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 19. *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 4 (Delayed-Input Soundness ⇒ Sparse mod-PC). *Assume: (a) a mod-PC* modPC *with weak evaluation binding*[16]*; (b) a mod-AHP* modAHP *for* $\mathcal{R}^{poly}$ *that: (i) is efficient for sparse polynomial evaluation (Definition 19); (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 2).*

8.2 A construction from SPARK [31]

In this section we reinterpret another building block from [31] 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 4 (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 "prime-agnostic" polynomial commitment (that can take in input the field at commitment time), which we show that we can build from our modPC. A formal treatment of the result in this section is in the appendix.

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 [31, 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 [31, 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 $\mathsf{PC}_{\text{SPARK}}$ in [31] 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. 4).
  - Ditto for each of the polynomial commitment openings for $\widetilde{e}_{row}, \widetilde{e}_{col}$ during the execution of Hyrax[17].

Fig. 5: A variant of the SPARK construction from [31] as an augmented mod-AHP for $\mathcal{R}^{\text{poly}}$. 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 [31, Section 7.2]

Theorem 5. *The construction in Fig. 5 is an (augmented) mod-AHP for* $\mathcal{R}^{poly}$ *with negligible delayed-input soundness; it is efficient for sparse polynomials.*

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

[17] Like in [31], we do not need Hyrax's zero-knowledge compiler [35]. 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 [24]).

The following is implied by our construction of mod-PC for dense polynomials, Theorem 4, Theorem 5 and by the security of our modPC.

Corollary 1. *There exists a mod-PC for sparse polynomials with weak evaluation binding secure in the GGMUO In particular, for a sparse multilinear polynomial g in 2μ variables and dense representation of size $O(2^\mu)$, the prover runs in time $O_\lambda(2^\mu \cdot \log_2 \|g\|_\infty)$.*

Acknowledgements

We thank Mahak Pancholi for useful feedback on early drafts of our work and for useful discussions on compilers for idealized protocols.

References

1. Diego F. Aranha, Emil Madsen Bennedsen, Matteo Campanelli, Chaya Ganesh, Claudio Orlandi, and Akira Takahashi. ECLIPSE: Enhanced compiling method for pedersen-committed zkSNARK engines. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *PKC 2022, Part I*, volume 13177 of *LNCS*, pages 584–614. Springer, Cham, March 2022.
2. S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2006.
3. Thomas Attema and Ronald Cramer. Compressed Σ-protocol theory and practical application to plug & play secure algorithmics. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 513–543. Springer, Cham, August 2020.
4. 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, Part I*, volume 12674 of *LNCS*, pages 393–414. Springer, Berlin, Heidelberg, March 2021.
5. 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, *CRYPTO 2021, Part IV*, volume 12828 of *LNCS*, pages 123–152, Virtual Event, August 2021. Springer, Cham.
6. Dan Boneh, Benedikt Bünz, and Ben Fisch. Batching techniques for accumulators with applications to IOPs and stateless blockchains. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 561–586. Springer, Cham, August 2019.
7. Jorge Luis Borges, Margarita Guerrero, Francisco Toledo, and Francisco Toledo. *Manual de zoología fantástica*, volume 125. Fondo de cultura económica México, 1957.
8. 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.
9. Benedikt Bünz, Ben Fisch, and Alan Szepieniec. Transparent SNARKs from DARK compilers. Cryptology ePrint Archive, Report 2019/1229, 2019.
10. Benedikt Bünz, Ben Fisch, and Alan Szepieniec. Transparent SNARKs from DARK compilers. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 677–706. Springer, Cham, May 2020.
11. 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, *ASIACRYPT 2021, Part III*, volume 13092 of *LNCS*, pages 3–33. Springer, Cham, December 2021.
12. Matteo Campanelli, Dario Fiore, and Hamidreza Khoshakhlagh. Witness encryption for succinct functional commitments and applications. In Qiang Tang and Vanessa Teague, editors, *PKC 2024, Part II*, volume 14602 of *LNCS*, pages 132–167. Springer, Cham, April 2024.
13. Matteo Campanelli, Nicolas Gailly, Rosario Gennaro, Philipp Jovanovic, Mara Mihali, and Justin Thaler. Testudo: Linear time prover SNARKs with constant size proofs and square root size universal setup. In Abdelrahaman Aly and Mehdi Tibouchi, editors, *LATINCRYPT 2023*, volume 14168 of *LNCS*, pages 331–351. Springer, Cham, October 2023.
14. Dario Catalano, Dario Fiore, and Ida Tucker. Additive-homomorphic functional commitments and applications to homomorphic signatures. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part IV*, volume 13794 of *LNCS*, pages 159–188. Springer, Cham, December 2022.
15. Binyi Chen, Benedikt Bünz, Dan Boneh, and Zhenfei Zhang. HyperPlonk: Plonk with linear-time prover and high-degree custom gates. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part II*, volume 14005 of *LNCS*, pages 499–530. Springer, Cham, April 2023.
16. Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Psi Vesely, and Nicholas P. Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 738–768. Springer, Cham, May 2020.
17. Geoffroy Couteau, Thomas Peters, and David Pointcheval. Removing the strong RSA assumption from arguments over the integers. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 321–350. Springer, Cham, April / May 2017.
18. Ivan Damgård and Eiichiro Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In Yuliang Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 125–142. Springer, Berlin, Heidelberg, December 2002.
19. Ivan Damgård and Maciej Koprowski. Generic lower bounds for root extraction and signature schemes in general groups. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 256–271. Springer, Berlin, Heidelberg, April / May 2002.
20. Eiichiro Fujisaki and Tatsuaki Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In Burton S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 16–30. Springer, Berlin, Heidelberg, August 1997.

21. 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.
22. Chaya Ganesh, Anca Nitulescu, and Eduardo Soria-Vazquez. Rinocchio: SNARKs for ring arithmetic. *Journal of Cryptology*, 36(4):41, October 2023.
23. Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 113–122. ACM Press, May 2008.
24. Alexander Golovnev, Jonathan Lee, Srinath T. V. Setty, Justin Thaler, and Riad S. Wahby. Brakedown: Linear-time and field-agnostic SNARKs for R1CS. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part II*, volume 14082 of *LNCS*, pages 193–226. Springer, Cham, August 2023.
25. Jens Groth. Non-interactive zero-knowledge arguments for voting. In John Ioannidis, Angelos Keromytis, and Moti Yung, editors, *ACNS 05International Conference on Applied Cryptography and Network Security*, volume 3531 of *LNCS*, pages 467–482. Springer, Berlin, Heidelberg, June 2005.
26. Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 177–194. Springer, Berlin, Heidelberg, December 2010.
27. Helger Lipmaa. On diophantine complexity and statistical zero-knowledge arguments. In Chi-Sung Laih, editor, *ASIACRYPT 2003*, volume 2894 of *LNCS*, pages 398–415. Springer, Berlin, Heidelberg, November / December 2003.
28. Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. In *31st FOCS*, pages 2–10. IEEE Computer Society Press, October 1990.
29. Silvio Micali. A secure and efficient digital signature algorithm. Technical Memo MIT/LCS/TM-501b, Massachusetts Institute of Technology, Laboratory for Computer Science, April 1994.
30. Barkley Rosser. Explicit bounds for some functions of prime numbers. *American Journal of Mathematics*, 63(1):211–232, 1941.
31. Srinath Setty. Spartan: Efficient and general-purpose zkSNARKs without trusted setup. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 704–737. Springer, Cham, August 2020.
32. Eduardo Soria-Vazquez. Doubly efficient interactive proofs over infinite and non-commutative rings. In Eike Kiltz and Vinod Vaikuntanathan, editors, *TCC 2022, Part I*, volume 13747 of *LNCS*, pages 497–525. Springer, Cham, November 2022.
33. Justin Thaler. Proofs, Arguments, and Zero-Knowledge, 2023.
34. Patrick Towa and Damien Vergnaud. Succinct diophantine-satisfiability arguments. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part III*, volume 12493 of *LNCS*, pages 774–804. Springer, Cham, December 2020.
35. 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.
36. Benjamin Wesolowski. Efficient verifiable delay functions. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 379–407. Springer, Cham, May 2019.
37. Tiancheng Xie, Yupeng Zhang, and Dawn Song. Orion: Zero knowledge proof with linear prover time. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part IV*, volume 13510 of *LNCS*, pages 299–328. Springer, Cham, August 2022.

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 [28] described in Figure 6, 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. 6: The Sumcheck Protocol

B 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 $\mathbf{B} = [\mathsf{q}^{2^t}] \cdot \mathbf{A} \in \mathbb{G}_N^\lambda$ where q^{2^t} is linear in the number of coefficients of the orional 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. 7) to verify the correctness of the computation. Note that the only dependency between the verifiers 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 20) in the group $\mathbb{G}_N$.

Definition 20 (Adaptive Root Assumption [36]). *Let $(\mathcal{A}_0, \mathcal{A}_1)$ be arbitrary PPT algorithms. The adaptive root assumption states:*

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

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

Lemma 3 (Soundness of Σ_{PoE} [36]). *Assuming the Adaptive Root Assumption holds in $\mathbb{G}_N$, Σ_{PoE} (Fig. 7) is an argument system for relation $\mathcal{R}_{\Sigma_{\mathsf{PoE}}}$:*

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

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.

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

1 : V samples $p \leftarrow\!\!\$\ \mathbb{P}_\lambda$ and sends p to P
2 : P computes:
3 : $\quad \Phi_{(\Delta)} \leftarrow \lfloor \Delta / p \rfloor \, ; \mathbf{Q} \leftarrow [\Phi_{(\Delta)}] \cdot \mathbf{A} \in \mathbb{G}_N^{\mathsf{d}}$
4 : P sends $(\Psi_{(\Delta)}, \mathbf{Q})$ to V
5 : V checks: $[p] \cdot \mathbf{Q} + [\Delta \bmod p] \cdot \mathbf{A} \stackrel{?}{=} \mathbf{B}$

Fig. 7: The Proof-of-Exponentiation Protocol [36] by Wesolowski.

C Protocols of Block et al.

In this section we include a brief overview of the protocols $\Sigma_{\mathsf{Eval}}/\Sigma_{\mathsf{MultiEval}}$ (Fig. 9) by Block et. al [5]. 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 Weosolowski Σ_{PoE} (see Appendix B), the section contains no original contributions. We refer the reader to the original paper for further details. The use of the Weosolowski Σ_{PoE} does not affect extraction as the PoE relation $\mathcal{R}_{\Sigma_{\mathsf{PoE}}}$ has no witness.

Notation. Define the multi-linear evaluation:

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

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

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

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

C.1 Evaluation Protocol

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. 8).

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

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

Fig. 8: Binding notion for the multi-linear commitment scheme.

Theorem 6 (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 [5].

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

output:accept or reject

if $\mathsf{r} = k$:

  P sends $Z \in \mathbb{Z}^{\lambda}$ to V

  V Checks:

    $\|Z\|_{\infty} \leq 2^m \cdot (2\lambda)^k$

    $\vec{y} \stackrel{?}{\equiv} Z \bmod q$

    $\mathbf{C} \stackrel{?}{=} ([Z] \cdot G, \ldots, [Z] \cdot G)$

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:

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

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

  // Prover commits to the "split polynomials"

  P Computes:

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

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

  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 $\vec{y} \stackrel{?}{=} \vec{y}_L \cdot (1 - x_{\mathsf{r}+1}) + \vec{y}_R \cdot x_{\mathsf{r}+1}$

  $\Sigma_{\mathsf{PoE}}(\mathbf{C}_R, \mathbf{C} - \mathbf{C}_L, \mathsf{q}^{k-\mathsf{r}-1})$

  // Verifier samples random binary matrixes U_L, U_R and sends them to the prover.

  $\mathsf{V} : U = [U_L \| U_R] \leftarrow\!\!\$\ \{0,1\}^{\lambda \times 2\lambda}$ 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 $Z_L, Z_R \in \mathbb{Z}^{\lambda \times 2^{k-\mathsf{r}-1}}$ 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. 9: MultiEval protocol by Block, Holmgren, Rosen, Rothblum and Soni [5], included here for completeness and unify the notation.

$\Sigma_{\mathsf{Eval}}(\mathsf{pp}, C \in \mathbb{G}_N, \vec{x} \in \mathbb{F}_q^k, y \in \mathbb{F}_q, m, q; \mathcal{Y}, \mathcal{Z} \in \mathbb{Z}^{2^k})$

$\vec{y} = (y, \ldots, y) \in \mathbb{F}_q^{\lambda}$

$\mathbf{C} = (C, \ldots, C) \in \mathbb{G}^{\lambda}$

$Z = (\mathcal{Z}, \ldots, \mathcal{Z}) \in \mathbb{Z}^{\lambda \times 2^k}$

$\Sigma_{\mathsf{MultiEval}}(\mathbf{C}, \mathsf{r} = 1, \vec{x}, \vec{y}, m, q; Z)$

Fig. 10: Evaluation proof by Block, Holmgren, Rosen, Rothblum and Soni [5], invoking the $\Sigma_{\mathsf{MultiEval}}$ protocol on λ parallel instances.

D Fingerprinting of Polynomial Evaluation

Theorem 7 (Number of Primes [30]). *Let $\pi(x)$ denote the prime-counting function, i.e. the number of primes less than or equal to x. Then:*

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

Lemma 4 (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. Fix $\vec{x} \in \mathbb{Z}^k$ and denote by m the smallest m st. $\|\vec{x}\|_\infty \leq 2^m$. Then $f(\vec{x}) \neq 0$ implies:*

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

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

Proof (Proof of Lemma 4). The evaluation $y = f(\vec{x})$ has a norm $\|y\|_\infty \leq 2^{\ell+m\cdot d}$: it is a summation of at most 2^ℓ non-zero terms, each of norm at most $2^{m\cdot d}$. 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 the probability that $y \equiv 0 \mod q$ for a uniformly sampled prime $q \in \mathbb{P}_\lambda$ is:

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

For any λ with $2^\lambda > 55$.

E The Generic Group of Unknown Order Model

In this section, we prove the knowledge soundness of Σ_{Final}. The proof is in the Generic Group of Unknown Order (GGUO) model: the adversary is a generic algorithm with black-box access to the group $\mathbb{G}_N$:

Definition 21 (Generic Group of Unknown Order (GGUO) Model [19]). *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* $\sigma(x_n)$

We use a simple lemma which states that computing non-trivial discrete-log relations in a generic group is hard:

Lemma 5 (Discrete Logarithm (Bünz et. al [6])). *Let $\mathbb{G}_N$ be a generic group where $|\mathbb{G}_N|$ is a uniformly chosen integer in $[\mathsf{A}, \mathsf{B}]$, where $1/\mathsf{A}$ and $1/|\mathsf{B} - \mathsf{A}|$ are negligible in λ. Let $\mathcal{A}$ be a generic algorithm and let $\vec{G} = \{G_1, \ldots, G_\mathsf{m}\}$ be the outputs of $\mathcal{O}_1$. Then if $\mathcal{A}$ runs in polynomial time, it succeeds with at most negligible probability in outputting $\vec{\alpha}, \vec{\beta} \in \mathbb{Z}^\mathsf{m}$ such that $\langle \vec{\alpha}, \vec{G} \rangle = \langle \vec{\beta}, \vec{G} \rangle$ and $\vec{\alpha} \neq \vec{\beta}$. We call this event DLOG.*

Proof (Proof of Lemma 5). See Bünz et. al [6].

F Proof-of-Knowledge for Last Round Messages

We outlined in the main body that to make communication and computation of $\Sigma_{\mathsf{MultiEval}}$ poly-logarithmic in the size (m) of the coefficients of the polynomial $f(\vec{X})$ our idea is to replace the final round of $\Sigma_{\mathsf{MultiEval}}$ by a proof-of-knowledge Σ_{Final} of an accepting last round message. Recall the relation of accepting last round messages in the protocol $\Sigma_{\mathsf{MultiEval}}$ from section 9:

$$\mathcal{R}_{\mathsf{final}} \coloneqq \left\{ \left(\left(\begin{matrix} \mathbf{C} = (C_{(Z_1)}, \ldots, C_{(Z_\lambda)}) \\ \vec{y} = (y_1, \ldots, y_\lambda) \end{matrix} \right), Z \right) : \begin{matrix} \forall i. y_i \equiv Z_i \bmod q \\ \forall i. C_{(Z_i)} = [Z_i] \cdot G \\ \forall i. \|Z_i\|_\infty \leq \mathsf{bound} \end{matrix} \right\} \tag{2}$$

If we decompose this relation coordinate-wise:

$$((\mathbf{C}, \vec{y}), Z) \in \mathcal{R}_{\mathsf{final}} \iff \forall i \in [\lambda]. ((C_{(Z_i)}, y_i), Z_i) \in \mathcal{R}_{\mathsf{single}} \tag{3}$$

$$\text{Where } \mathcal{R}_{\mathsf{single}} \coloneqq \left\{ ((C_{(Z)}, y), Z) : \begin{matrix} y \equiv Z \bmod q \\ C_{(Z)} = [Z] \cdot G \\ \|Z\|_\infty \leq \mathsf{bound} \end{matrix} \right\} \tag{4}$$

Next we construct a more efficient Proof-of-Knowledge for $\mathcal{R}_{\mathsf{single}}$ and apply this protocol λ times in parallel to prove $\mathcal{R}_{\mathsf{final}}$. We obtain our Proof-of-Knowledge for $\mathcal{R}_{\mathsf{single}}$ by extending techniques of Boneh, Bünz and Fisch [6] building upon Wesolowski [36]. Our techniques allows proving that the openings of integer commitments vanish over any set of constant degree integer polynomials: a general commit-and-prove for constant depth circuits. We combine this with techniques of Couteau et. al [17] and Limpaa [27] to enable a very efficient (exact) range check. In summary, we construct a proof-of-knowledge for the following relation:

$$\mathcal{R}_{\Sigma_{\mathsf{Final}}} \coloneqq \left\{ \left(C_{(Z)}, (Z, \sigma_1, \sigma_2, \sigma_3, \phi)\right) : \begin{matrix} Z - y = \phi \cdot q \\ 1 + 4 \cdot (\mathsf{bound} + Z) \\ \quad \cdot (\mathsf{bound} - Z) = \sigma_1^2 + \sigma_2^2 + \sigma_3^2 \\ C_{(Z)} = [Z] \cdot G \end{matrix} \right\} \tag{5}$$

The first condition is equivalent to $y \equiv Z \bmod q$, while Couteau et. al, building upon an optimization by Groth [25], showed that the second condition is equivalent to $Z \in [-\mathsf{bound}, \mathsf{bound}]$ or $\|Z\|_\infty \leq \mathsf{bound}$. We refer to their paper [17] for details, including how to efficiently compute the decomposition $\sigma_1, \sigma_2, \sigma_3$. Although our techniques can be generalized for low-degree arithmetic constraints, we provide the exact protocol Σ_{Final} for $\mathcal{R}_{\Sigma_{\mathsf{Final}}}$ in Fig. 11. The communication complexity of the protocol is 3 group elements from $\mathbb{G}_N$ and 5 integers of size λ.
Because it derives from the techniques of Boneh, Bünz and Fisch the extractor for the protocol is in the Generic Group of Unknown Order model (GGUO) introduced by Damgård and Koprowski [19].

Theorem 8 (Knowledge Soundness of Σ_{Final}). *Σ_{Final} is a proof-of-knowledge for the relation of Eq. (5) in the Generic Group of Unknown Order (GGUO) model (see Appendix E).*

Proof (Proof of Theorem 8). Consider two accepting transcripts of Σ_{Final} sharing the same first-round message $C_{(\star)}$:

$$\left(C_{(\star)}, p, (Q_{(\star)}, Q_{(Z)}, \Psi_{(\phi)}, \Psi_{(\sigma_1)}, \Psi_{(\sigma_2)}, \Psi_{(\sigma_3)}, \Psi_{(Z)})\right)$$
$$\left(C_{(\star)}, p', (Q'_{(\star)}, Q'_{(Z)}, \Psi'_{(\phi)}, \Psi'_{(\sigma_1)}, \Psi'_{(\sigma_2)}, \Psi'_{(\sigma_3)}, \Psi'_{(Z)})\right)$$

Produced by a polynomial time generic adversary $\mathcal{A}^{\mathcal{O}_1, \mathcal{O}_2}(1^\lambda)$.
From the generic group oracle we recover the representations:

$$C_{(Z)} = \langle [\vec{a}], \vec{G} \rangle \qquad Q_{(Z)} = \langle [\vec{\alpha}], \vec{G} \rangle \qquad Q_{(\star)} = \langle [\vec{\beta}], \vec{G} \rangle$$
$$C_{(\star)} = \langle [\vec{b}], \vec{G} \rangle \qquad Q'_{(Z)} = \langle [\vec{\alpha}'], \vec{G} \rangle \qquad Q'_{(\star)} = \langle [\vec{\beta}'], \vec{G} \rangle$$

$\Sigma_{\mathsf{Final}}(C_{(Z)}, y, \mathsf{bound}; Z)$

// Prover computes circuit witness:

Compute: $\sigma_1, \sigma_2, \sigma_3$ s.t. $1 + 4 \cdot Z \cdot (\mathsf{bound} - Z) = \sigma_1^2 + \sigma_2^2 + \sigma_3^2$

Compute: $\phi \leftarrow \lfloor Z/q \rfloor$

// Prover commit to each variable (note Z already committed)

Send to V : $C_{(\star)} \leftarrow [\phi] \cdot G_1 + [\sigma_1] \cdot G_2 + [\sigma_2] \cdot G_3 + [\sigma_3] \cdot G_4 \in \mathbb{G}_N$

// Verifier samples a random prime

$p \leftarrow\$ \mathbb{P}_\lambda$

// Prover computes remainders and quotients of all variables

$\forall v \in \{\phi, \sigma_1, \sigma_2, \sigma_3, Z\}$:

$\Phi_{(v)} \leftarrow \lfloor v/p \rfloor$

$\Psi_{(v)} \leftarrow v \bmod p$

// Compute commitments to the (potentially large) quotients.

$Q_{(Z)} \leftarrow [\Phi_{(Z)}] \cdot G_1 \in \mathbb{G}_N$

$Q_{(\star)} \leftarrow [\Phi_{(\phi)}] \cdot G_1 + [\Phi_{(\sigma_1)}] \cdot G_2 + [\Phi_{(\sigma_2)}] \cdot G_3 + [\Phi_{(\sigma_3)}] \cdot G_4 \in \mathbb{G}_N$

// Send the two quotient commitments and individual remainders to the verifier

Send $Q_{(Z)}, Q_{(\star)}, \{\Psi_{(v)}\}_{v \in \{\phi, \sigma_1, \sigma_2, \sigma_3, Z\}}$ to V

// Verifier checks quotient & remainder decomposition

$C_{(Z)} \stackrel{?}{=} [p] \cdot Q_{(Z)} + [\Psi_{(Z)}] \cdot G$

$C_{(\star)} \stackrel{?}{=} [p] \cdot Q_{(\star)} + [\Psi_{(\phi)}] \cdot G_1 + [\Psi_{(\sigma_1)}] \cdot G_2 + [\Psi_{(\sigma_2)}] \cdot G_3 + [\Psi_{(\sigma_3)}] \cdot G_4$

// Verifier checks the arithmetic relation over the prime field $\mathbb{F}_p$

$\Psi_{(Z)} - y \stackrel{?}{\equiv} \Psi_{(\phi)} \cdot q \mod p$

$1 + 4 \cdot \left(\mathsf{bound} + \Psi_{(Z)}\right) \cdot \left(\mathsf{bound} - \Psi_{(Z)}\right) \stackrel{?}{\equiv} {\Psi_{(\sigma_1)}}^2 + {\Psi_{(\sigma_2)}}^2 + {\Psi_{(\sigma_3)}}^2 \mod p$

Fig. 11: A succinct protocol enabling the prover to convince the verifier that the opening of a commitment $C_{(Z)}$ is a positive integer less than bound and congruent to y modulo q. The verifier running time is bounded by the time required to compute bound mod p, which for structured bound, such as $\mathsf{bound} = 2^{2^m} \cdot (2\lambda)^k$, can be done in $O(\log(\log(\mathsf{bound})))$ time.

Where $\vec{G} \in \mathbb{G}_N^{\mathsf{m}}$ is the list of responses from $\mathcal{O}_1$ including $G_1, \ldots, G_4$ contained in the CRS of Σ_{Final}. Because the verifier accepts both transcripts we recover representations of $C_{(Z)}$ and $C_{(\star)}$:

$$\begin{aligned}
C_{(Z)} &= \langle [\vec{a}], \vec{G} \rangle \\
&= \langle [p \cdot \vec{\alpha}], \vec{G} \rangle + [\Psi_{(Z)}] \cdot G_1 \\
&= \langle [p' \cdot \vec{\alpha}'], \vec{G} \rangle + [\Psi'_{(Z)}] \cdot G_1 \\
C_{(\star)} &= \langle [\vec{b}], \vec{G} \rangle \\
&= \langle [p \cdot \vec{\beta}], \vec{G} \rangle + [\Psi_{(q)}] \cdot G_1 + [\Psi_{(\sigma_1)}] \cdot G_2 + [\Psi_{(\sigma_2)}] \cdot G_3 + [\Psi_{(\sigma_3)}] \cdot G_4 \\
&= \langle [p' \cdot \vec{\beta}'], \vec{G} \rangle + [\Psi'_{(q)}] \cdot G_1 + [\Psi'_{(\sigma_1)}] \cdot G_2 + [\Psi'_{(\sigma_2)}] \cdot G_3 + [\Psi'_{(\sigma_3)}] \cdot G_4
\end{aligned}$$

Because DLOG occurs with negligible probability (see Lemma 5), we conclude that, except with negligible probability, the following equalities hold:

$$\begin{aligned}
a_1 &= p \cdot \alpha_1 + \Psi_{(Z)} = p' \cdot \alpha'_1 + \Psi'_{(Z)} \\
b_1 &= p \cdot \beta_1 + \Psi_{(q)} = p' \cdot \beta'_1 + \Psi'_{(q)} \\
b_2 &= p \cdot \beta_2 + \Psi_{(\sigma_1)} = p' \cdot \beta'_2 + \Psi'_{(\sigma_1)} \\
b_3 &= p \cdot \beta_3 + \Psi_{(\sigma_2)} = p' \cdot \beta'_3 + \Psi'_{(\sigma_2)} \\
b_4 &= p \cdot \beta_4 + \Psi_{(\sigma_3)} = p' \cdot \beta'_4 + \Psi'_{(\sigma_3)} \\
\forall i > 1. a_i &= p \cdot \alpha_i = p' \cdot \alpha'_i \\
\forall i > 4. b_i &= p \cdot \beta_i = p' \cdot \beta'_i
\end{aligned}$$

We can conclude that, except with neglible probability, over $(p, p') \leftarrow\!\$\ \mathbb{P}_\lambda \times \mathbb{P}_\lambda$:

$$\forall i > 1.\ a_i = 0 \quad \text{and} \quad \forall i > 4. b_i = 0$$

In other words, the representations are actually of the form:

$$\begin{aligned}
C_{(Z)} &= [a_1] \cdot G \\
C_{(\star)} &= [b_1] \cdot G_1 + [b_2] \cdot G_2 + [b_3] \cdot G_3 + [b_4] \cdot G_4
\end{aligned}$$

Furthermore, for an adversary making $\mathfrak{q}$ queries to the $(\mathcal{O}_1, \mathcal{O}_2)$ generic group oracles, we can conclude that the extracted quantities are bounded by $2^{\mathfrak{q}}$:

$$\|a_1\|_\infty \leq 2^{\mathfrak{q}}, \|b_1\|_\infty \leq 2^{\mathfrak{q}}, \|b_2\|_\infty \leq 2^{\mathfrak{q}}, \|b_3\|_\infty \leq 2^{\mathfrak{q}}, \|b_4\|_\infty \leq 2^{\mathfrak{q}}$$

Let us rename $a_1 = Z$, $b_1 = q$, $b_2 = \sigma_1$, $b_3 = \sigma_2$, $b_4 = \sigma_3$ to relate the extracted values to the witness of an honest prover. Observe that we already established:

$$\begin{aligned}
\Psi_{(Z)} &\equiv Z \mod p, & \Psi_{(\phi)} &\equiv \phi \mod p \\
\Psi_{(\sigma_1)} &\equiv \sigma_1 \mod p, & \Psi_{(\sigma_2)} &\equiv \sigma_2 \mod p, & \Psi_{(\sigma_3)} &\equiv \sigma_3 \mod p
\end{aligned}$$

If we apply Lemma 4 to the integer polynomials:

$$\begin{aligned}
f_1(X_1, X_2, X_3, X_4, X_5) &\coloneqq (X_1 - y) - (X_2 \cdot q) \\
f_2(X_1, X_2, X_3, X_4, X_5) &\coloneqq 1 + 4 \cdot X_1 \cdot (\mathsf{bound} - X_1) - (X_2^2 + X_3^2 + X_4^2)
\end{aligned}$$

And evaluations $f_1(Z, \phi, \sigma_1, \sigma_2, \sigma_3) \in \mathbb{Z}$ or $f_2(Z, \phi, \sigma_1, \sigma_2, \sigma_3) \in \mathbb{Z}$, we can conclude that the probability over the choice of $p \leftarrow\!\$\ \mathbb{P}_\lambda$ that:

$$\begin{aligned}
\Psi_{(Z)} - y &\equiv \Psi_{(\phi)} \cdot q \mod p \\
1 + 4 \cdot \big(\mathsf{bound} + \Psi_{(Z)}\big) \cdot \big(\mathsf{bound} - \Psi_{(Z)}\big) &\equiv {\Psi_{(\sigma_1)}}^2 + {\Psi_{(\sigma_2)}}^2 + {\Psi_{(\sigma_3)}}^2 \mod p
\end{aligned}$$

Yet:

$$\begin{aligned}
Z - y &\neq \phi \cdot q \\
1 + 4 \cdot (\mathsf{bound} + Z) \cdot (\mathsf{bound} - Z) &\neq \sigma_1^2 + \sigma_2^2 + \sigma_3^2
\end{aligned}$$

Is negligible in λ since $\mathfrak{q}$ is polynomial in λ. We conclude that Σ_{Final} is a proof-of-knowledge for the relation $\mathcal{R}_{\Sigma_{\mathsf{Final}}}$ (see Eq. (5)).

Theorem 9 (The Running Time of Σ_{Final}). *The verifier* V *runs in time polynomial in* $\tilde{O}(\lambda \cdot \log\log \mathsf{bound})$, assuming $(\mathsf{bound} \bmod p)$ *can be computed in time* $O(\log\log \mathsf{bound})$.

Proof. By inspection of the protocol.

Corollary 2. *For the case where* $\mathsf{bound} = 2^m \cdot (2\lambda)^2$ *as in* $\Sigma_{\mathsf{MultiEval}}$, *then* $(\mathsf{bound} \bmod p)$ *can be computed in time* $O(\log(p)^2 \cdot (\log(m) + \log(\lambda)))$ *using a simple square-and-multiply algorithm. As a result, the verifiers running time in* Σ_{Final} *is polynomial in* $\log(m)$ *as desired.*

G Opening Proof: Composing Σ_{Eval} and Σ_{Final}

Our opening proof is obtained by composing the Σ_{Eval} and Σ_{Final} protocols: inspired by Compressed Σ-Protocols [3] we replace the final round of Σ_{Eval} with the Argument-of-Knowledge Σ_{Final}. To show that the composition is an Argument-of-Knowledge the strategy is to extract the final round from Σ_{Final} (see Appendix F) and then employ the extractor from Σ_{Eval} (from Block et. al [5]).

$\Sigma_{\mathsf{Open}}(\mathsf{pp} = (\mathbb{G}_N, G_1, G_2, G_3, G_4), C \in \mathbb{G}_N, \vec{x} \in \mathbb{F}_q^k, y \in \mathbb{F}_q, m, q; f)$

1 : $(\mathbf{C}, \vec{y}; \vec{Z}) \leftarrow \mathsf{LastRound}(\Sigma_{\mathsf{Eval}}(\mathsf{pp} = (\mathbb{G}_N, G_1), C, \vec{x}, y, m, q; f))$

2 : Decompose $(C_{(Z_1)}, \ldots, C_{(Z_\lambda)}) = \mathbf{C}$

3 : Decompose $(Z_1, \ldots, Z_\lambda) = \vec{Z}$

4 : Decompose $(y_1, \ldots, y_\lambda) = \vec{y}$

5 : In parallel: **for** $i \in [0, \lambda]$: $\Sigma_{\mathsf{Final}}(C_{(Z_i)}, y_i, \mathsf{bound} = 2^m \cdot (2\lambda)^k; Z_i)$

Fig. 12: Composition of Σ_{Eval} and Σ_{Final} to obtain Σ_{Open}. We denote by $\mathsf{LastRound}$ the function that extracts the last round statement of Σ_{Eval}.

Lemma 6 (Knowledge Soundness of Σ_{Open}). *The protocol* Σ_{Open} *is an Argument-of-Knowledge for the relation:*

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

In the GGUO (see Appendix E) model, with negligible knowledge error. Furthermore, for a fixed C, *finding* $f^* \neq f$ *such that* $((\mathsf{pp}, C, \vec{x}^*, y^*, q^*), f^*) \in \mathcal{R}_{\Sigma_{\mathsf{Open}}}$ *and* $((\mathsf{pp}, C, \vec{x}, y, q), f) \in \mathcal{R}_{\Sigma_{\mathsf{Open}}}$ *breaks the order assumption on* $\mathbb{G}_N$*: subtracting the two polynomials from each other, then encoding, yields a non-zero multiple of the group order* N.

Theorem 10 (Weak-evaluation binding of modPC). *The scheme* modPC *satisifies Weak Evaluation Binding (Definition 22)*

Proof (Proof of Theorem 10). We show that mod-PC satisfy Definition 22. Recall that weak evaluation binding states that it is hard to prove an incosistent polynomial evaluation for an *honestly computed commitment*. If we apply the extractor Ext to $\mathcal{A}_2$ producing accepting transcripts for Σ_{Open}, we recover f^* st:

$$((C, \vec{x}^*, y^*, q^*), f^*) \in \mathcal{R}_{\Sigma_{\mathsf{Open}}}$$

Where $C = \mathsf{Com}(\mathsf{pp}, f)$. If $f^* = f$ then the adversary does not break binding, since in particular $f(\vec{x}^*) = y^*$. On the other hand, if $f^* \neq f$ then the adversary breaks the order assumption on $\mathbb{G}_N$ as stated in Lemma 6.

We prove a slightly stronger notion (see Definition 23) of knowledge soundness for mod-PC, in which soundness holds when the adversary is allowed to choose the prime q. This stronger notion implies the weaker one (Definition 13), see Appendix J.2 for details.

Lemma 7 (Strong Knowledge Soundness of mod-PC). *The scheme mod-PC is a strong knowledge-sound (Definition 23) with negligible knowledge error* ϵ.

Proof (Proof of Lemma 7). Lemma 6 We show that mod-PC satisfies strong knowledge soundness. Recall that strong knowledge soundness states that the evaluations of the extracted polynomial f will coincide with the proven evaluations by $\mathcal{A}_{\text{prf}}$. This follows directly from Lemma 6 by observing that every evalaution proof allows us to extract:

$$((C, \vec{x}^*, y^*, q^*), f^*) \in \mathcal{R}_{\Sigma_{\mathsf{Open}}}$$

If $f^* \neq f$ we break binding. Otherwise, if $f^* = f$ then the evaluations of f and f^* coincide and $\forall j \in [m]\ f(z_j) \equiv y_j \mod q$ meaning that $\mathcal{A}_{\text{prf}}$ does not win the Strong Knowledge Soundness game.

H Reducing the Requirements for Indexing Commitments in our Compiler

We observe that in the proof of Theorem 2 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 $\mathsf{modPC}_{\text{idx}}$ and $\mathsf{modPC}_{\text{w}}$ be two mod-PCs satisfying respectively weak evaluation binding and knowledge soundness.
- Apply the compiler in Fig. 14, with the only difference that we use $\mathsf{modPC}_{\text{idx}}$ for the indexing polynomials and $\mathsf{modPC}_{\text{w}}$ for the oracle polynomials.

The following theorem follows from the proof of Theorem 2.

Theorem 11. *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. 14 described above yields a complete, full knowledge-sound interactive argument.*

I Missing Proofs

I.1 Proof of Lemma 2

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:

$$\text{for all } j,\ \langle A_j, \vec{z}\rangle \circ \langle B_j, \vec{z}\rangle - \langle C_j, \vec{z}\rangle \equiv 0 \mod q \tag{6}$$

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. (6) 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 4 seeing y as the evaluation of low degree polynomial in $\vec{z}$ (notice that for that we use the norm bound requirements from Definition 6). □

I.2 Proof of Theorem 1

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 2.

Let $\mathsf{P}^* = (\mathsf{P}^*_{\text{orcl}}, \mathsf{P}^*_{\text{rst}})$ 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 $\mathsf{P}^*_{\text{orcl}}$

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 testing property. If it is not a fingerprint witness, however, we can invoke weak knowledge soundness (Definition 13) and conclude we reached a contradiction. □

I.3 Proof of Theorem 2

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 [1,11,10,16]. Consider an adversary $\tilde{\mathcal{P}}$ producing an accepting transcript with probability $\tilde{p}$. We show an extractor for $\tilde{\mathcal{P}}$ in Fig. 13. 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. 14.

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:

$$
\begin{aligned}
&\Pr[\mathsf{Ext}_{\mathrm{ARG}}\ \text{✗}\ \wedge \tilde{\mathcal{P}}\ \text{✓}] \\
&= \Pr[\mathsf{Ext}_{\mathrm{AHP}}^{\mathsf{P}^*}\ \text{✗}\ \wedge \tilde{\mathcal{P}}\ \text{✓}] && (7)\\
&= \Pr[\mathsf{Ext}_{\mathrm{AHP}}^{\mathsf{P}^*}\ \text{✗}\ \wedge \tilde{\mathcal{P}}\ \text{✓}\ \wedge \mathsf{P}^*\ \text{✓}] + \Pr[\mathsf{Ext}_{\mathrm{AHP}}^{\mathsf{P}^*}\ \text{✗}\ \wedge \tilde{\mathcal{P}}\ \text{✓}\ \wedge \mathsf{P}^*\ \text{✗}] && (8)\\
&\leq \Pr[\mathsf{Ext}_{\mathrm{AHP}}^{\mathsf{P}^*}\ \text{✗}\ \wedge \mathsf{P}^*\ \text{✓}] + \Pr[\mathsf{P}^*\ \text{✗} \wedge \tilde{\mathcal{P}}\ \text{✓}\ ] && (9)\\
&\leq \mathsf{negl}(\lambda) + \mathsf{negl}(\lambda) && (10)
\end{aligned}
$$

- Eq. (7) follows by construction of $\mathsf{Ext}_{\mathrm{ARG}}$.
- In Eq. (8) we apply a simple marginalization.
- In Eq. (9) we apply the elementary fact $X \to Y \implies \Pr[X] \leq \Pr[Y]$.
- We bound the left-hand summand in Eq. (10) by simply invoking knowledge soundness of the underlying mod-AHP. For the right-hand summand we invoke Lemma 8.

This concludes the proof. □

$\mathsf{Ext}_{\mathrm{ARG}}\left(\mathsf{srs}; \mathsf{rnd}_{\tilde{\mathcal{P}}}\right)$

$(\mathsf{i}, \mathsf{x}, \tilde{\mathsf{st}}) \leftarrow \tilde{\mathcal{P}}_1(\mathsf{srs}; \mathsf{rnd}_{\tilde{\mathcal{P}}})$

$\mathsf{aux} := \left(\tilde{\mathsf{st}}, \mathsf{rnd}_{\tilde{\mathcal{P}}}, \mathsf{srs}\right)$

Output $\mathsf{w} \leftarrow \mathsf{Ext}^{\mathsf{P}^*}_{\mathrm{AHP}}(1^\lambda, \mathsf{i}, \mathsf{x}, \mathsf{aux})$

$\mathsf{P}^*\left(\mathsf{st}, \mathsf{tr}_{i^*}, \mathsf{aux}\right)$

Retrieve i from the state

If $i^* \leq \mathsf{k}$ then invoke $\mathsf{P}^*_{\mathrm{orcl}}\left(\mathsf{st}, \mathsf{tr}_{i^*}, \mathsf{aux}\right)$ // defined below

Else invoke $\mathsf{P}^*_{\mathrm{rst}}\left(\mathsf{st}, \mathsf{tr}_{i^*}, \mathsf{aux}\right)$ // from plain interactive protocol in underlying mod-AHP

$\mathsf{P}^*_{\mathrm{orcl}}\left(\mathsf{st}, \vec{\rho}_1, \ldots, \vec{\rho}_{i^*}, \mathsf{aux} = \left(\tilde{\mathsf{st}}, \mathsf{rnd}_{\tilde{\mathcal{P}}}, \mathsf{srs} = (1^\lambda, \mathsf{pp}_{\mathsf{PC}})\right)\right)$

Retrieve i from the state

If $i^* = 0$ then return $\left(g_{0,j}\right)_{j \in [\mathsf{s}(0)]} \leftarrow \mathcal{I}(1^\lambda, \mathsf{i})$

For $j = 1, \ldots, \mathsf{s}(i^*)$:

Invoke $\mathsf{Ext}^{\mathrm{PC}, i^*, j}(\mathsf{pp}_{\mathsf{PC}}, \mathsf{aux} = (\tilde{\mathsf{st}}, \mathsf{rnd}_{\mathsf{P}}, \vec{\rho}_1, \ldots, \vec{\rho}_{i^*}))$ to obtain g_j

Abort if $g_j \notin \mathbb{Z}_{\leq d}[X_1, \ldots, X_\mu]$ where $d := \mathsf{d}(\lambda, |\mathsf{i}|, i^*, j), \mu := \mathsf{v}(\lambda, |\mathsf{i}|, i^*)$

return $\left(g_j\right)_{j=1,\ldots,\mathsf{s}(i^*)}$

Fig. 13: Extractor for proof of Theorem 2.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.

$\mathcal{A}^{\mathrm{PC},i,j}_{\mathrm{com}}(\mathsf{pp}_{\mathsf{PC}}, \mathsf{aux} = (\tilde{\mathsf{st}}, \mathsf{rnd}_{\tilde{\mathcal{P}}}, \vec{\rho}_1, \ldots, \vec{\rho}_i))$

Let $\mathsf{srs} = (1^\lambda, \mathsf{pp}_{\mathsf{PC}})$

Run $\mathsf{EmulateArgTranscript}(\mathsf{srs}, \tilde{\mathsf{st}}, \mathsf{rnd}_{\tilde{\mathcal{P}}}, \vec{\rho}_1, \ldots, \vec{\rho}_i, \bot, \bot)$ to obtain transcript $\mathcal{T}$

Retrieve the commitment $c_{i,j}$ from $\mathcal{T}$; Save $\mathsf{pp}_{\mathsf{PC}}, \mathsf{aux}$ as state st

Return $\left(\left(c_{i,j}, \mathsf{d}(\lambda, |\mathsf{i}|, i, j), \mathsf{v}(\lambda, |\mathsf{i}|, i)\right), \mathsf{st}\right)$

$\mathcal{Q}^{i,j}(\mathsf{pp}_{\mathsf{PC}}, q, \mathsf{aux} = (\tilde{\mathsf{st}}, \mathsf{rnd}_{\tilde{\mathcal{P}}}, \vec{\rho}_1, \ldots, \vec{\rho}_i))$

Let $\mathsf{srs} = (1^\lambda, \mathsf{pp}_{\mathsf{PC}})$

Sample $\vec{\rho}_{i+1}, \ldots, \vec{\rho}_{\mathsf{k}}, \vec{\rho}_{\mathrm{rst}}$ where $\mathsf{k} = \mathsf{k}(\lambda, |\mathsf{i}|)$

Run $\mathsf{EmulateArgTranscript}(\mathsf{srs}, \tilde{\mathsf{st}}, \mathsf{rnd}_{\tilde{\mathcal{P}}}, \vec{\rho}_1, \ldots, \vec{\rho}_{\mathsf{k}}, \vec{\rho}_{\mathrm{rst}}, q)$ to obtain transcript $\mathcal{T}$ w/ queries Q

Let $\vec{z}_{i,j} := \left(z : (i', j', z) \in Q, i = i', j = j'\right)$

Return $\left(\vec{z}_{i,j}, \mathsf{aux}_{\mathcal{Q}} := \left(\vec{\rho}_{i+1}, \ldots, \vec{\rho}_{\mathsf{k}}, \vec{\rho}_{\mathrm{rst}}\right)\right)$

$\mathcal{A}^{\mathrm{PC},i,j}_{\mathrm{prf}}(\mathsf{st}, \vec{z}_{i,j}, q, \mathsf{aux}_{\mathcal{Q}} := (\vec{\rho}_{i+1}, \ldots, \vec{\rho}_{\mathsf{k}}, \vec{\rho}_{\mathrm{rst}}))$

// st contains $\mathsf{aux} = (\tilde{\mathsf{st}}, \mathsf{rnd}_{\tilde{\mathcal{P}}}, \vec{\rho}_1, \ldots, \vec{\rho}_i)$

Run $\mathsf{EmulateArgTranscript}(\mathsf{srs}, \tilde{\mathsf{st}}, \mathsf{rnd}_{\tilde{\mathcal{P}}}, \vec{\rho}_1, \ldots, \vec{\rho}_{\mathsf{k}}, \vec{\rho}_{\mathrm{rst}}, q)$ to obtain transcript $\mathcal{T}$

// NB: queries $\vec{z}_{i,j}$ in transcript are the same as the ones in input to $\mathcal{A}^{\mathrm{PC},i,j}_{\mathrm{prf}}$

For each $z \in \vec{z}_{i,j}$ retrieve corresponding proof $\pi_z^{(\mathrm{eval})}$ and evaluation y_z from $\mathcal{T}$

Return $\left(y_z, \pi_z^{(\mathrm{eval})}\right)_{z \in \vec{z}_{i,j}}$

$\mathsf{EmulateArgTranscript}(\mathsf{srs}, \tilde{\mathsf{st}}, \mathsf{rnd}_{\tilde{\mathcal{P}}}, \vec{\rho}_1, \ldots, \vec{\rho}_{i^*}, \vec{\rho}_{\mathrm{rst}}, q)$

// Notice that by convention we have $\vec{\rho}_{\mathrm{rst}} = q = \bot$ if $i^* \leq \mathsf{k}(\lambda, |\mathsf{i}|)$

If $\vec{\rho}_{\mathrm{rst}} = q = \bot$ run an interaction with $\tilde{\mathcal{P}}$ till round i^* (included)

Else run a full interaction

(in both cases run $\tilde{\mathcal{P}}$ (resp. $\mathcal{V}$) w/ state/randomness $(\tilde{\mathsf{st}}, \mathsf{rnd}_{\tilde{\mathcal{P}}})$ (resp. $(\vec{\rho}_1, \ldots, \vec{\rho}_{i^*}, q, \vec{\rho}_{\mathrm{rst}})$))

Return the transcript from the interaction $\mathcal{T}$

Fig. 14: Auxiliary algorithm definitions for Fig. 13. We assume each $\mathcal{A}^{\mathrm{PC},i,j} = \left(\mathcal{A}^{\mathrm{PC},i,j}_{\mathrm{com}}, \mathcal{A}^{\mathrm{PC},i,j}_{\mathrm{prf}}\right)$ has embedded the parameter $|\mathsf{i}|$.

Lemma 8. *In the proof of Theorem 2 the quantity* $\Pr[\mathsf{P}^* \not\checkmark \wedge \tilde{\mathcal{P}} \checkmark]$ *in Eq.* (9) *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:

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

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

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

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

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

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:

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

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 $\pi^{(\mathrm{eval})}$ 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 $\pi^{(\mathrm{eval})}$ 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. □

I.4 Proof of Theorem 3

It is immediate to see that Fig. 4 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 [31]. We observe that after the prime is sampled the protocol in Fig. 4 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.[18]

We now show how we can invoke results from the proof of Theorem 5.1 from [31], 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 [31]. 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 soundess 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.

[18] 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 [15].

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. 4) would be the same of that of a verifier having access to $\tilde{w}_q$. This proves the claim above and concludes the proof. □

I.5 Proof of Theorem 4

Consider the compiler in Fig. 3 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[19]. □

I.6 Proof of Theorem 5

The core efficiency properties of the construction are argued in [31]; 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 [31]: The proof of Lemma 7.6 in [31] essentially argues that the construction in Fig. 5 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 [31] argues for soundness of the protocol as an argument for $\mathcal{R}^{\mathrm{poly}}$; this translates directly to delayed-input soundness for mod-AHPs. □

[19] A formal version of these last steps is analogous to the ones in the proof of Theorem 2, to which we refer the reader.

J Additional Definitions for mod-PCs

J.1 Weak Evaluation binding

Definition 22 (Weak Evaluation Binding). *For any PPT adversary $\mathcal{A}$, $D, M, \lambda \in \mathbb{N}$, $d \leq D$, $\mu \leq M$:*

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

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

J.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 23.

Definition 23 ((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 $\lambda, D, M \in \mathbb{N}$ and PPT $\mathcal{A} = (\mathcal{A}_{com}, \mathcal{A}_{prf})$ 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:*

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

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

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

K.1 Augmented model

In Fig. 15 augment the model of mod-AHP presented in Definition 10 with an additional round[20] 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.

[20] 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 8.

– **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 $\mathcal{I}(1^\lambda, \mathsf{i})$[21]. 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

K.2 Prime-Agnostic Polynomial Commitments

A prime-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 Appendix K.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 24. *A prime-agnostic polynomial commitment consists of a tuple* $\mathsf{PC}_{agn} = (\mathsf{Setup}, \mathsf{Com}, \mathsf{ProveEval}, \mathsf{VfyEval})$ *such that:*

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

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

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

Knowledge soundness (with knowledge error ϵ). For any $\lambda, D, M \in \mathbb{Z}$ and PPT $\mathcal{A} = (\mathcal{A}_{\text{com}}, \mathcal{A}_{\text{prf}})$ 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:

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

where above the extractor has access to the random tape of the adversary[22].

Remark 5 (Existing constructions). We remark that field-agnostic constructions such as Brakedown [24] and Orion [37] satisfy our definition.

Constructions from mod-PCs with Strong Extractability It is rather straightforward to produce a prime-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 23). Showing security of the resulting construction is trivial.

Theorem 12. *If there exists a mod-PC with negligible strong knowledge error (Definition 23) then there exists a prime-agnostic polynomial commitment with the same efficiency with negligible knowledge error.*

From the fact that our construction satisfies Definition 23 (see Remark 5) we have the following corollary.

Corollary 3. *There exists a prime-agnostic polynomial commitment with negligible knowledge error secure in the GGUO.*

K.3 Extending the compiler

In Fig. 16 we present an extended version of the compiler in Fig. 3 to apply to the augmented mod-AHPs defined in Fig. 15. The approach we use is straightforward and consists of applying a prime-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 [11,16,1]. As a consequence the proof of security of Theorem 13 also follows directly from a minor variant of the proofs of security from these works and the one we present for Theorem 2.

Theorem 13. *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 soundess error, let PC_{agn} be a prime-agnostic PC_{agn} with negligible knowledge soundness error then the construction in Fig. 16 is an interactive argument with preprocessing (Definition 2) for $\mathcal{R}$.*

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

Setup$(1^\lambda, N)$**:** ... $\mathsf{pp}_{\mathsf{agn}} \leftarrow \mathsf{PC}_{\mathrm{agn}}.\mathsf{Setup}(1^\lambda, D, M)$ 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. 3.

We observe that, in general, the other results we have on mod-AHPs, including those in Section 8.1, 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 8).

Theorem 14 (Delayed-Input Soundness ⇒ Sparse mod-PC). *Assume: (a) a mod-PC* modPC *with weak evaluation binding; (b) a prime-agnostic polynomial commitment* PC_{agn} *with negligible knowledge soundness (for dense polynomials); (c) a (possibly augmented) mod-AHP* modAHP *for* $\mathcal{R}^{poly}$ *that: (i) is efficient for sparse polynomial evaluation (Definition 19); (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 2).*

Proof. The proof is almost completely the proof for Theorem 4. The fact that we have additional oracle polynomials does not change the essence of the proof above. Instead of applying the compiler from Fig. 3, we apply its extended variant in Fig. 16. The remaining observations follow mutatis mutandis. □