DewTwo: a transparent PCS with quasi-linear prover, logarithmic verifier and 4.5KB proofs from falsifiable assumptions

Benedikt Bünz
bb@nyu.edu
New York University

Tushar Mopuri
tmopuri@upenn.edu
University of Pennsylvania

Alireza Shirzad
alrshir@upenn.edu
University of Pennsylvania

Sriram Sridhar
srirams@berkeley.edu
University of California, Berkeley

Abstract

We construct the first polynomial commitment scheme (PCS) that has a transparent setup, quasi-linear prover time, $\log N$ verifier time, and $\log\log N$ proof size, for multilinear polynomials of size N. Concretely, we have the smallest proof size amongst transparent PCS, with proof size less than 4.5KB for $N \leq 2^{30}$. We prove that our scheme is secure entirely under falsifiable assumptions about groups of unknown order. The scheme significantly improves on the prior work of Dew (PKC 2023), which has super-cubic prover time and relies on the Generic Group Model (a non-falsifiable assumption). Along the way, we make several contributions that are of independent interest: PoKEMath, a protocol for efficiently proving that an arbitrary predicate over committed integer vectors holds; SIPA, a bulletproofs-style inner product argument in groups of unknown order; we also distill out what prior work required from the Generic Group Model and frame this as a falsifiable assumption.

Contents

1 Introduction

Succinct arguments (SAs)[1]enable an untrusted prover to convince an efficient verifier that it performed a computation correctly. Since their introduction [GMR85], there has been a Cambrian explosion in constructions [Kil92; GKR08; GGPR13; BCTV14; Gro16; BBHR19; GWC19; CBBZ23; ACFY24]. In parallel with these constructions, new and exciting applications have emerged. Today, succinct arguments secure billions of dollars of cryptocurrencies, are used to improve the scalability [LXZSZ23] as well as the privacy of cryptographic payments [Ben+14], are used to build advanced signature schemes [KCLM22], enable electronic voting [SDW08], and certify the authenticity of photographs [NT16].

A key tool in the construction of SAs is polynomial commitment schemes (PCS). A PCS allows a prover to commit to a multilinear polynomial $p \in \mathbb{F}[X_1, \ldots, X_\mu]$[2] of size $N = 2^\mu$ that $p(\mathbf{y}) = z$, for any evaluation point $\mathbf{y} \in \mathbb{F}^\mu$. There are many proof systems for NP (e.g., for arithmetic circuits) in which the prover sends polynomial oracles to the verifier, and the verifier queries these at a few evaluation points and accepts or rejects based on these evaluations [MBKM19; GWC19; CHMMVW20; BBHR18; Set20; CBBZ23; STW23]. These so-called polynomial interactive oracle proofs (poly-IOPs) [BFS20; CHMMVW20] are efficient in terms of communication and verifier efficiency, modulo the sending and evaluation of these polynomial oracles. Fortunately, these poly-IOPs can be compiled with a PCS to form an efficient, succinct argument. Given the efficiency of existing poly-IOPs, which are often information-theoretically secure, the efficiency metrics, as well as the cryptographic assumptions of the resulting SA almost entirely depend on the efficiency and the assumptions of the underlying PCS. This motivates designing polynomial commitments with near-optimal properties. There are several important evaluation criteria to optimize for in a PCS:

Transparent setup. Some schemes, e.g. KZG [KZG10] rely on a trusted setup: a trusted party is required to run the setup algorithm. If this process is subverted, then the PCS will be insecure. Especially in permissionless blockchains, it is unclear who should run this trusted setup. Therefore, the ideal scheme does *not* rely on a trusted setup but instead has a *transparent* setup, which can be run in a publicly verifiable manner.

Efficient prover. The PCS prover should be able to commit to the polynomial and generate the evaluation proof efficiently. The PCS commitment and evaluation time are lower bounds to the SA's prover's runtime. More efficient provers enable larger statements to be proven, which is particularly important when SAs are used to outsource computation. Efficient schemes have linear or quasi-linear (in N, the size of the polynomial) prover runtimes.

Efficient verifier. The verifier should be able to verify the PCS evaluation proofs efficiently. Ideally, the PCS verifier runs in logarithmic time in the size of the polynomials committed to in an SA, which corresponds to the size of the underlying computation being proven. Schemes with efficient verification are useful for outsourcing computation to weak machines, e.g., smartphones or smart contacts.

Small proof size. The proof size of the evaluation proof, as well as the size of the commitment to a polynomial, should be both asymptotically and practically small. This is particularly the case when the proof gets distributed to many parties, as is the case in many blockchain applications.

[1]We will primarily discuss protocols as interactive arguments. However, both DewTwo and all relevant related work have public-coin verifiers and can heuristically be turned into non-interactive SNARKs. Multiple of the discussed applications require non-interactivity.

[2]Some polynomial commitments are designed for univariate polynomials. We focus on multilinear PCS's, which can be used to commit to and evaluate univariate polynomials and vice versa [ZXZS20].

Provable security from falsifiable assumptions. A fundamental goal of any cryptographic primitive is to have provable security under reasonable cryptographic assumptions. An important property of such assumptions is that they are efficiently falsifiable, i.e. one could design an algorithm that efficiently shows that the assumption can be broken. Unfortunately, many PCS's can only be proven secure under non-falsifiable assumptions.[3]

1.1 Contributions

We design DewTwo, the first PCS for multilinear polynomials with a transparent setup (our scheme relies on groups of unknown order [DGS22]) that simultaneously has quasi-linear prover time, logarithmic verification time, and proofs that are both asymptotically (of size $O(\log\log(N))$ for polynomial size N) and practically (of size 4.5KB for $N \leq 2^{30}$) small.

We prove that our scheme is secure under falsifiable assumptions about groups of unknown order (GUO), while all prior work based on GUO were proven secure in the generic group model (a non-falsifiable assumption). To boost confidence in our falsifiable assumptions, we show that they all hold in the generic group model.

DewTwo improves upon, and in the case of the former builds upon, prior work: Dew [AGLMS23], and Behemoth [SB23]. These works had asymptotically constant proof size but (1) were only secure in the generic group model, (2) for all reasonable parameters have concretely *larger* proofs than DewTwo, and (3) most importantly, have *cubic prover time*. We circumvent these limitations using multiple innovations, all of which are of independent interest:

Falsifiable assumptions PoKE, which is used in Dew and Behemoth, was previously proven secure in the generic group model [BBF19]. Following an idea from [LPS24], we distill what is necessary from the generic group model into a falsifiable assumption: the modular consistency assumption. The assumption intuitively states that a prover must answer certain PoKE queries consistent with a previously committed, bounded-size integer. We prove that the assumption holds in the generic group model and that it implies the adaptive root assumption [Wes19].

PoKEMath We develop an efficient protocol that can prove *arbitrary* predicates $f(\boldsymbol{x})$ over a committed vector of integers $\boldsymbol{x} \in \mathbb{Z}^N$. Importantly, the proof size and verifier time of the protocol is independent of how large the integers are. PoKEMath generalizes the PoKE protocol from [BBF19], and is both extremely general and practically efficient. We further extend PoKEMath to PoKEDEx which can handle range proofs, alongside the arbitrary predicate. That is, a PoKEDEx argument shows that $x_i \in [a_i, b_i]$ for each $i \in [N]$ *and* that $f(\boldsymbol{x}) = 0$.

Efficient square root decomposition Dew, Behemoth, and DewTwo all rely on range proofs as a key component. Dew and Behemoth prove that an integer $x \in [a, b]$ by computing the square decomposition of a related integer $y = f(x, a, b)$ into the sum of 4 square roots. Unfortunately, the time taken to compute such a decomposition is at least *cubic* in the bit-length of the integer, and for us, the integer y encodes the entire polynomial, i.e., has $\Theta(N)$ bits. We show that the number can instead be decomposed into $\log(N)$ many squares in $O(M(N)\log^2(N)) = O(N\log^3(N))$ time. Here, $M(N)$ is the complexity of multiplying two N-bit numbers, which is quasilinear in N for large N [HH21].

[3]There exists a key impossibility result for proving *non-interactive* succinct arguments are secure under falsifiable assumptions [GW11]. We consider our assumption in the interactive, pre-fiat-shamir variant of the scheme.

Self inner product arguments As an important sub-protocol of PoKEDEx, we construct a new 'self inner product argument' (SIPA) for groups of unknown order. SIPA allows a prover to convince a verifier that $\langle \boldsymbol{x}, \boldsymbol{x} \rangle = v \mod \ell$, for a committed vector of integers $\boldsymbol{x} \in \mathbb{Z}^k$ and some public prime ℓ, with just $O(\log(k))$ communication. Our protocol is in the spirit of classic IPA techniques [BCCGP16; BBBPWM18], and is used to show that integers y of the aforementioned form are indeed the sum of $\log(N)$ many squares. Thus, our overall proof size reduces to $O(\log\log(N))$.

DewTwo We put everything together by designing a polynomial commitment scheme DewTwo using PoKEDEx. The scheme is concretely efficient and has small constants. The overall proof size is bounded by

$$2\lceil \log\lceil \log 5 + 2(7(N-2)\log(N)\lambda + \log(N) + \lambda(\log(N)+1))\rceil\rceil + 4$$

group of unknown order elements and $12\lambda\lceil \log\lceil \log 5 + 2(7(N-2)\log(N)\lambda + \log(N) + \lambda(\log(N) + 1))\rceil\rceil + 22\lambda$ bits of integers. For polynomials of size $N \leq 2^{30}$, the proof contains at most 15 group elements, requiring 192 bytes per group element (128-bit security level for hyper-elliptic curves from [DGS22]). This can be upper bounded by 4.5KB. Our proof sizes are significantly smaller than that of Dew or Behemoth, which require 66 and 47 group elements (12 and 9 KB), respectively. Despite, or perhaps because of, this efficiency, our scheme is arguably simpler than prior works.

1.2 Applications

In this section, we list some unique benefits and applications of DewTwo, even amongst other polynomial commitment schemes.

Field flexibility. A unique aspect of the DewTwo PCS is that the prover commits to the polynomial in integer representation, and the evaluation field is only chosen at evaluation time. In fact, it is possible to evaluate the polynomial modulo multiple integers verifiably (not just primes). This stands in contrast to elliptic curve-based schemes, where the polynomial's field is intrinsically tied to the order of the curve. For example, it is impossible to commit to polynomials over a prime field smaller than 2λ-bits using an elliptic curve, as otherwise, the curve cannot be secure. DewTwo, on the other hand, can be used to evaluate polynomials modulo arbitrarily small integers.

Transparent proof compression. One promising application for DewTwo is the compression of large hash-based SNARK proofs that are to be posted on a blockchain. Most zk-rollups use hash-based proof systems, like FRI [BBHR18] and STIR [ACFY24] due to their fast prover times and transparent setup. Unfortunately, hash-based proofs have proof sizes that are about 100 KB for even a single PCS evaluation proof (See Table1.3). In order to compress these proofs, most deployments use an elliptic curve-based (EC-based) proof system (e.g., one using KZG [KZG10]) to reduce the size of the final proof that is posted on chain. This is done via what is called SNARK composition, where the EC-based proofs are used to prove the correctness of the hash-based SNARK. Unfortunately, these EC-based proof systems generally require a trusted setup. Using DewTwo, one could avoid the need for a trusted setup while still enjoying the benefits of small proof sizes and efficient verification.

1.3 Related Work

In Table 1.3, we compare DewTwo to various relevant polynomial commitment schemes in literature. Note that DewTwo has the smallest proof size of any polynomial commitment with a transparent setup, modulo Bulletproofs [BBBPWM18]. Bulletproofs, however, has linear verification time, and thus does not suffice for many applications, including for constructing verifiable computation schemes. Unlike its closest comparisons Dew [AGLMS23] and Behemoth [SB23], DewTwo only has quasi-linear prover time. Additionally, although DewTwo has asymptotically larger proof sizes than these schemes, for any reasonable parameter choice our concrete proof size is significantly smaller. Compared to Dory [Lee21] and STIR [ACFY24], DewTwo has 9x and 25x smaller proofs, respectively.

Scheme	$\mathcal{P}$ time	$\mathcal{V}$ time	$\lvert\pi\rvert$ (Assympt.) $\lvert\pi\rvert$ (Concrete $N = 2^{30}$)	Falsif.	Trans.
KZG [KZG10]	$N\ \mathbb{F}$ $N\ \mathbb{G}_{\mathsf{EC}}$	1 P	$1\ \mathbb{G}_{\mathsf{EC}}$ 32 bytes	Yes	No
STIR + Redshift [ACFY24; KPV22]	$N\log(N)\ \mathbb{F}$ $N\mathbb{H}$	$\lambda\log\log(N)\log(N)\mathbb{H}$ $\lambda^2\ \mathbb{F}$	$\lambda\log(N)\log\log(N)\mathbb{H}$ 107 KB	Yes	Yes
WHIR [ACFY24; KPV22]	$N\log(N)\ \mathbb{F}$ NH	$\lambda\log\log(N)\log(N)\ \mathbb{H}$ $\lambda\log\log(N)\log(N)\ \mathbb{F}$	$\lambda\log(N)\log\log(N)\mathbb{H}$ 107 KB	Yes	Yes
Greyhound [ACFY24; KPV22]	$\lambda N\ \mathbb{Z}_q$	$\lambda\sqrt{N}\ \mathbb{Z}_q$	$\lambda\log\log(N)\ \mathbb{Z}_q$ 53 KB	Yes	Yes
Bulletproofs [BBBPWM18]	$N\ \mathbb{G}_{\mathsf{EC}}$	$N\ \mathbb{G}_{\mathsf{EC}}$	$2\log(N)\ \mathbb{G}_{\mathsf{EC}}$ 1.5 KB	Yes	Yes
Dory [Lee21]	$N\ \mathbb{G}_{\mathsf{EC}}$	$\log(N)\ \mathbb{G}_T$	$6\log(N)\ \mathbb{G}_T$ 37 KB	Yes	Yes
Dew [AGLMS23]	$N\ \mathbb{G}_{\mathsf{GUO}}$ $N^2M(N)\ \mathbb{B}$	$\log(N)\ \mathbb{F}$ $O(1)\ \mathbb{G}_{\mathsf{GUO}}$	$O(1)\ \mathbb{G}_{\mathsf{GUO}}$ 12 KB	No	Yes
Behemoth [SB23]	$N\ \mathbb{G}_{\mathsf{GUO}}$ $N^2M(N)\ \mathbb{B}$	$\log(N)\ \mathbb{F}$ $O(1)\ \mathbb{G}_{\mathsf{GUO}}$	$O(1)\ \mathbb{G}_{\mathsf{GUO}}$ 9 KB	No	Yes
DewTwo This work (Theorem 5.5)	$N\log^4(N)\mathbb{B}$ $N\log^2 N\ \mathbb{G}_{\mathsf{GUO}}$	$\log(N)\ \mathbb{F}$ $\log(N)\ \mathbb{G}_{\mathsf{GUO}}$	$\log\log(N)\ \mathbb{G}_{\mathsf{GUO}}$ 4.5 KB	Yes	Yes

Table 1: Comparison of several polynomial commitment schemes on various (asymptotic) metrics. We omit the $O(\cdot)$ notation for brevity and denote by $\mathbb{F}$ a field operation, by $\mathbb{H}$ a hash operation, by $\mathbb{B}$ a bit operation, by $\mathbb{G}_{\mathsf{EC}}$ a group operation in elliptic-curves, by $\mathbb{G}_T$ a group operation in the pairing target group, by $\mathbb{G}_{\mathsf{GUO}}$ a group operation in groups of unknown order, by $\mathbb{Z}_q$ an operation on integers modulo q, and by P an elliptic curve pairing operation.

Groups of unknown order. DewTwo requires a group of unknown order. Two candidate GUOs with a transparent setup are class groups [BW88] and hyper-elliptic curves [DGS22]. A key challenge with these transparent groups is that some groups will have smooth or semi-smooth order, which makes them vulnerable to generic attacks. Unfortunately, it is impossible to test whether the order of a group is smooth or semi-smooth (outside of running the attack up to a smoothness parameter). Dobson et al. [DGS22] suggest selecting the group large enough such that with probability $1 - 2^{-k}$, the fastest generic attack on a randomly sampled group runs in time 2^{λ}. We set our parameters in the comparison to $k = 40$ and $\lambda = 128$. For these parameters, the group must have about 2^{1536} elements. For hyper-elliptic curves with such group order, an element can be represented using 192 bytes [DGS22]. We suggest that this is sufficiently secure but further study in these transparent groups of unknown order is warranted.

Proofs of exponentiation. Our $\mathsf{PoKEMath}$ and $\mathsf{PoKEDEx}$ protocols are based on PoKE [BBF19] which in turn is based on the efficient VDF [BBBF18] protocol from Wesolowski [Wes19]. Wesolowski's protocol showed that for some $y \in \mathbb{G}$, $y = 2^T \cdot x$ with communication and verification independent of T. Fascinatingly, our constructions show that similar protocols can be used to prove much more complicated relations while maintaining the constant overhead in proof size.

2 Preliminaries

2.1 Notation

General notation. We write both groups and fields additively. We use $\mathbb{Z}$ to denote the set of integers. $\mathsf{Primes}[\lambda, 2\lambda]$ is the set of all prime numbers in the range $[2^{\lambda}, 2^{2\lambda}]$.

Given a group $\mathbb{G}$, we denote group elements using uppercase letters $G \in \mathbb{G}$ and n-dimensional vector of group elements by $\mathbf{G} \in \mathbb{G}^n$, with G_i denoting the i-th element of the vector. Vectors are indexed as $\mathbf{G} = [G_i]_{i=1}^n = (G_1, G_2, \dots, G_n)$.

For a field $\mathbb{F}$, we use lowercase letters to denote an element $x \in \mathbb{F}$ and $\boldsymbol{x} \in \mathbb{F}^n$ to denote an n-dimensional vector of field elements. We denote matrices by $\boldsymbol{M} \in \mathbb{F}^{n \times n}$, with $\boldsymbol{M}_i$ representing the i-th row of the matrix. We use similar notation for integers.

We use the function $\mathsf{MSM}(m, n)$ as the asymptotic complexity of multi-scalar multiplication of size m with n bit scalars. We note that the state of the art multi-scalar multiplication algorithms use variants of the Pippenger algorithm and have complexity $\mathsf{MSM}(m, n) = O(mn/\log m)$ [Wes19].

Integer casting. For a prime field $\mathbb{F}_q$ and an element $x \in \mathbb{F}_q$, we denote by $\tilde{x}$ the casting of x to an integer in the range $[0, q-1]$. Similarly, for a vector of field elements $\boldsymbol{x} \in \mathbb{F}_q^n$, we denote by $\tilde{\boldsymbol{x}}$ the vector of integers $[\tilde{x}_i]_{i=1}^n$. Additionally, for a polynomial $p(X) = \sum_{i=0}^{n-1} p_i X^i \in \mathbb{F}_q[X]$, we denote by $\tilde{p}(y)$ the evaluation of the cast integer polynomial at $y \in \mathbb{Z}$.

Vector operations. Given a vector $\boldsymbol{x} = [x_i]_{i=1}^n$, it can be partitioned into its left and right halves, denoted $\boldsymbol{x}_L = [x_i]_{i=1}^{n/2}$ and $\boldsymbol{x}_R = [x_{n/2+i}]_{i=1}^{n/2}$ respectively. We use $\bar{\boldsymbol{x}}$ to denote the flipped vector, $\overline{\boldsymbol{x}} := [x_n, x_{n-1}, \dots, x_1]$. We denote by $\boldsymbol{x}|_{[1:m]}$ where $m \le n$ the trimmed vector $[x_i]_{i=1}^m$.

Binary representation. Let n be a positive integer, and let i be an integer such that $0 \le i < 2^n$. We denote by $\mathsf{bin}(i) : \mathbb{Z} \to \{0,1\}^n$, the n -bit binary representation of i, arranged with the most significant bit (MSB) first. For $1 \le j \le n$, let $\mathsf{bin}(i, j)$ denote the j-th bit of this representation, where $\mathsf{bin}(i, 1)$ corresponds to the MSB and $\mathsf{bin}_n(i, n)$ to the least significant bit (LSB).

Integer operations. Throughout the paper, We use the function $\mathsf{M}(n)$ as the asymptotic complexity of multiplying two n-bit integers. We note that the state-of-the-art integer multiplication algorithms [HH21] have complexity $\mathsf{M}(n) = O(n \log n)$. It can also be shown that the complexity of integer division is $O(\mathsf{M}(n))$ [Rei89; CA69].

Ternary relations. In this work, we consider ternary relations $\mathscr{R}$ consisting of tuples of the form $(\mathsf{pp}, \mathbb{x}, \mathbb{w})$, where pp are some public parameters, $\mathbb{x}$ is an instance, and $\mathbb{w}$ is a witness. For a particular choice of pp, the corresponding NP relation $\mathscr{R}^{\mathsf{pp}}$ is defined to be $\mathscr{R}^{\mathsf{pp}} := \{(\mathbb{x}, \mathbb{w}) : (\mathsf{pp}, \mathbb{x}, \mathbb{w}) \in \mathscr{R}\}$.

Inner products. Given two field vectors $\boldsymbol{x} \in \mathbb{F}^n$ and $\boldsymbol{y} \in \mathbb{F}^n$, we use the notation $\langle \boldsymbol{x}, \boldsymbol{y} \rangle := \sum_{i=1}^n x_i \cdot y_i$ for their inner product. We similarly use $\langle \boldsymbol{x}, \mathbf{Y} \rangle := \sum_{i=1}^n x_i \cdot Y_i$ to denote the inner product of a field vector $\boldsymbol{x} \in \mathbb{F}^n$ and a group vector $\mathbf{Y} \in \mathbb{G}^n$.

2.2 Multilinear polynomials

A multivariate polynomial is said to be *multilinear* if the individual degree of any variable X_i in any term is at most 1. We use $\mathbb{F}^{\leq 1}[X_1, \ldots, X_\mu]$ to denote the space of all μ-variate multilinear polynomials. A multilinear polynomial $p(X_1, \ldots, X_\mu) = \sum_{i=0}^{2^\mu - 1} p_i \prod_{j=1}^{\mu} X_j^{\mathsf{bin}(i,j)}$ of size $N = 2^\mu$ is uniquely defined by its N coefficients, which we denote $\boldsymbol{p} := [p_i]_{i=0}^{2^\mu-1}$. Notice that we make an exception to our vector notation while indexing vectors corresponding to multilinear polynomials.

2.3 Arguments of knowledge

An (adaptive) argument of knowledge for a ternary relation $\mathscr{R}$, consisting of public parameter, instance, witness tuples of the form $(\mathsf{pp}, \mathbb{x}, \mathbb{w})$, is a tuple of three algorithms $\mathsf{ARG} = (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$. Setup is a probabilistic polynomial-time setup algorithm that given a security parameter 1^λ (in unary) and a bound $N \in \mathbb{N}$ on the size of the instances supported, samples the public parameters pp, which define the NP relation $\mathscr{R}^{\mathsf{pp}}$. $\mathcal{P}$ and $\mathcal{V}$ are probabilistic polynomial-time interactive algorithms that satisfy the following properties:

Completeness. For all $N \in \mathbb{N}$ and all expected poly-time adversaries $\mathcal{A}$, the following holds

$$\Pr\left[(\mathbb{x}, \mathbb{w}) \in \mathscr{R}^{\mathsf{pp}} \implies \langle \mathcal{P}(\mathsf{pp}, \mathbb{x}, \mathbb{w}), \mathcal{V}(\mathsf{pp}, \mathbb{x}) \rangle = 1 \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, N) \\ (\mathbb{x}, \mathbb{w}) \leftarrow \mathcal{A}(\mathsf{pp}) \end{array} \right] = 1$$

Adaptive knowledge soundness. We say ARG is an adaptive knowledge-sound argument, if for all $N \in \mathbb{N}$ and all expected poly-time stateful adversaries $\mathcal{A} := (\mathcal{A}_1, \mathcal{A}_2)$ there exists an expected poly-time extractor Ext such that $\mathsf{Adv}^{\mathsf{Adap\text{-}KS}}_{\mathsf{ARG},\mathsf{Ext},\mathcal{A}}(\lambda) :=$

$$\Pr\left[\begin{array}{c} \langle \mathcal{A}_2(\mathsf{pp}), \mathcal{V}(\mathsf{pp}, \mathbb{x}) \rangle = 1 \\ \wedge \\ (\mathbb{x}, \mathbb{w}^*) \notin \mathscr{R}^{\mathsf{pp}} \end{array} \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, N) \\ \mathbb{x} \leftarrow \mathcal{A}_1(\mathsf{pp}) \\ \mathbb{w}^* \leftarrow \mathsf{Ext}^{\mathcal{A}}(\mathsf{pp}, \mathbb{x}) \end{array} \right] = \mathrm{negl}(\lambda)$$

Succinctness. We say ARG is succinct if the size of the communication transcript between $\mathcal{P}(\mathsf{pp}, \mathbb{x}, \mathbb{w})$ and $\mathcal{V}(\mathsf{pp}, \mathbb{x})$ is $\mathrm{poly}(\log(|\mathbb{w}|), \lambda)$ and the running time of $\mathcal{V}(\mathsf{pp}, \mathbb{x})$ is $\mathrm{poly}(\log(|\mathbb{w}|), |\mathbb{x}|, \lambda)$.

2.3.1 Setup

All the arguments of knowledge in this paper have the same setup algorithm Setup, which for convenience we present in this section. Setup takes as input the security parameter 1^λ and an integer $N \in \mathbb{N}$ and outputs the public parameters $\mathsf{pp} = (N, \mathbb{G}, \mathbf{G})$.

```
Setup(1^λ, N) → pp
_______________________________________________
1. Sample the group $\mathbb{G} \leftarrow \mathsf{GGen}(1^\lambda)$.
2. Sample the vector of generators $\mathbf{G} \leftarrow \mathbb{G}^N$.
3. Output $\mathsf{pp} := (N, \mathbb{G}, \mathbf{G})$.
```

2.4 Commitment schemes

A commitment scheme CS over a message space $\mathcal{M}$ is a tuple of PPT algorithms $\mathsf{CS} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$ where:

1. $\mathsf{CS.Setup}(1^\lambda, \mathcal{M}) \to \mathsf{pp}$: On input a security parameter 1^λ (unary) and a description of the message space generates the public parameters pp.
2. $\mathsf{CS.Commit}(\mathsf{pp}; m) \to (\mathsf{cm}; \mathsf{r})$: On input the public parameters pp and a message $m \in \mathcal{M}$, outputs a commitment cm and an opening hint r.
3. $\mathsf{CS.Open}(\mathsf{pp}, \mathsf{cm}, m, \mathsf{r}) \to \{0,1\}$: On input the public parameter pp, a commitment $\mathsf{CS.Commit}$, a message $\mathcal{M}$ and an opening hint r, verifies the opening of cm to m using the hint r.

Binding. A commitment scheme is *binding* if for all PPT adversaries $\mathcal{A}$,

$$\Pr\left[\begin{array}{c|c} \mathsf{CS.Open}(\mathsf{pp}, \mathsf{cm}, m, \mathsf{r}) = 1 & \\ \wedge & \mathsf{pp} \leftarrow \mathsf{CS.Setup}(1^\lambda) \\ \mathsf{CS.Open}(\mathsf{pp}, \mathsf{cm}, m', \mathsf{r}') = 1 & (\mathsf{cm}, m, m', \mathsf{r}, \mathsf{r}') \leftarrow \mathcal{A}(\mathsf{pp}) \\ \wedge & \\ m \neq m' & \end{array}\right] = \mathrm{negl}(\lambda)$$

2.5 Polynomial commitment schemes

Following the approach in [BFS19], we extend the commitment scheme syntax to polynomial commitment schemes. A polynomial commitment scheme is a tuple of protocols $\mathsf{PCS} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Eval})$ where $(\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$ is a binding commitment scheme for the message space $\mathbb{F}_q[X_1, \ldots, X_\mu]$ of μ-variate multilinear polynomials over some prime field $\mathbb{F}_q$, and

- $\mathsf{Eval}(\mathsf{pp}, \mathsf{cm}, z, \mathbf{y}, \mu; p(\mathbf{X})) \to b \in \{0,1\}$: is a succinct interactive public-coin argument between a PPT prover $\mathcal{P}$ and verifier $\mathcal{V}$. Both $\mathcal{P}$ and $\mathcal{V}$ have as input a commitment cm, evaluation value $z \in \mathbb{F}_q$, evaluation point $\mathbf{y} \in \mathbb{F}_q$, and a number of variables μ. The prover additionally knows the opening of cm to a secret multilinear polynomial $p(\mathbf{X}) \in \mathbb{F}_q[X_1, \ldots, X_\mu]$. The protocol convinces the verifier that $p(\mathbf{y}) = z$.

Extractability. Any prover that succeeds in the Eval protocol must possess knowledge of a polynomial $p \in \mathbb{F}_q[X_1, \ldots, X_\mu]$ such that $p(\mathbf{y}) = z$ and cm is a commitment to p. We formalize this notion as an argument of knowledge for the following relation:

Definition 2.1 (PCS indexed relation)**.** *Given* $\mathsf{pp} = (N, \mathbb{G}, \mathbf{G}) \leftarrow \mathsf{Setup}(1^\lambda, N)$*, we define the NP relation* $\mathscr{R}^{\mathsf{pp}}_{\mathsf{PCS}}$ *to be the set of tuples*

$$\begin{pmatrix} \mathbb{x}, \\ \mathbb{w} \end{pmatrix} = \begin{pmatrix} (q, \mu, \mathsf{cm}, \mathbf{y}, z, \mathcal{H}), \\ p(\mathbf{X}), \tilde{\mathbf{p}} \end{pmatrix}$$

where $q \in \mathbb{N}$ is a prime positive integer, $\mu \in \mathbb{N}$ is the number of variables, $\mathsf{cm} \in \mathbb{G}$ *is a commitment, $\mathbf{y} \in \mathbb{F}_q^\mu$ is the evaluation point, $z \in \mathbb{F}_q$ is the evaluation, $p(\mathbf{X}) \in \mathbb{F}_q^{\leq 1}[X_1, \ldots, X_\mu]$ is a multilinear polynomial, and $\mathcal{H}$ is the hint space such that:*

$$z = p(\mathbf{y}) \mod q \qquad \textit{and} \qquad \mathsf{PCS.Open}(\mathsf{pp}, \mathsf{cm}, p(\mathbf{X}), \tilde{\mathbf{p}}) = 1.$$

Weak vs Strong Extractability. Depending on whether adversaries are permitted to adaptively select the evaluation point $\mathbf{y}$, we distinguish between two notions of extractability: weak and strong extractability.

Definition 2.2 (Strong Extractability)**.** *A polynomial commitment scheme* $\mathsf{PCS} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Eval})$ *has strong extractability if* Eval *is a succinct adaptive argument of knowledge for the relation* $\mathscr{R}_{\mathsf{PCS}}$.

Before defining weak extractability, we introduce semi-adaptive arguments of knowledge, inspired by [BISW17], which considers adaptive and non-adaptive settings. Our work focuses on adaptive and semi-adaptive settings, with the latter bridging adaptive and non-adaptive notions. All arguments in this paper are adaptive unless stated otherwise.

Semi-Adaptive arguments of knowledge. The definition of a semi-adaptive argument of knowledge ARG for a ternary relation $\mathscr{R}$ is identical to that of an adaptive argument of knowledge, except that we relax the notion of adaptive knowledge soundness to semi-adaptive knowledge soundness. Given a partitioning of the instance $\mathbb{x} = (\mathbb{x}_1, \mathbb{x}_2, \mathbb{x}_3)$, we say ARG satisfies semi-adaptive knowledge soundness if for all $N \in \mathbb{N}$, every polynomial-time public-coin sampler $\mathcal{C}$, and all expected poly-time adversaries $\mathcal{A} := (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$, there exists an expected poly-time extractor Ext such that $\mathsf{Adv}^{\mathsf{Semi\text{-}Adap\text{-}KS}}_{\mathsf{ARG},\mathsf{Ext},\mathcal{C},\mathcal{A}}(\lambda) :=$

$$\Pr\left[\begin{array}{c|c} \langle \mathcal{A}_3(\mathsf{pp}), \mathcal{V}(\mathsf{pp}, \mathbb{x})\rangle = 1 & \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, N) \\ \mathbb{x}_1 \leftarrow \mathcal{A}_1(\mathsf{pp}) \\ \mathbb{x}_2 \leftarrow \mathcal{C} \\ \mathbb{x}_3 \leftarrow \mathcal{A}_2(\mathsf{pp}, \mathbb{x}_1, \mathbb{x}_2) \end{array} \\ \wedge & \\ (\mathbb{x}, \mathbb{w}^*) \notin \mathscr{R}^{\mathsf{pp}} & \begin{array}{l} \hline \mathbb{x} := (\mathbb{x}_1, \mathbb{x}_2, \mathbb{x}_3) \\ \mathbb{w}^* \leftarrow \mathsf{Ext}^{\mathcal{A}}(\mathsf{pp}, \mathbb{x}) \end{array} \end{array}\right] = \operatorname{negl}(\lambda)$$

Remark 2.3. The difference between adaptive and semi-adaptive knowledge soundness lies in how the instance is determined. In the semi-adaptive case, part of the instance is sampled by a public randomness, and the adversary can adaptively select the rest. This is particularly useful in extractability definitions of polynomial commitment schemes (see Definitions 2.2 and 2.4), where the evaluation point is chosen publicly, but the adversary determines the polynomial's value at that point.

Definition 2.4 (Weak Extractability)**.** *A polynomial commitment scheme* $\mathsf{PCS} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Eval})$ *has weak extractability if* Eval *is a succinct semi-adaptive argument of knowledge for the relation* $\mathscr{R}_{\mathsf{PCS}}$ *with an instance partition* $\mathbb{x}_1 = (q, \mu, \mathsf{cm}), \mathbb{x}_2 = \mathbf{y}, \mathbb{x}_3 = z$ *(see Section 2.3).*

2.6 Reductions of knowledge

In this section we state the definition of reductions of knowledge from [KP23], along with some relevant theorems. These can be thought of as an extension of arguments of knowledge.

Definition 2.5 (Reduction of knowledge)**.** *Consider two ternary relations* $\mathscr{R}_1$ *and* $\mathscr{R}_2$*, each consisting of tuples of the form* $(\mathsf{pp}, \mathbb{x}, \mathbb{w})$ *where* pp *is the set of public parameters,* $\mathbb{x}$ *is the instance, and* $\mathbb{w}$ *is the witness. We denote by the set of PPT algorithms* $\Pi = (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ *a reduction of knowledge from* $\mathscr{R}_1$ *to* $\mathscr{R}_2$ *if* Setup *is as defined in Section 2.3.1 and* $\mathcal{P}$ *and* $\mathcal{V}$ *are as follows:*

- $\mathcal{P}(\mathsf{pp}, \mathbb{x}_1, \mathbb{w}_1) \rightarrow (\mathbb{x}_2, \mathbb{w}_2)$*: Takes as input public parameters* pp*, and instance-witness pair* $(\mathbb{x}_1, \mathbb{w}_1)$ *and interactively reduces the statement* $(\mathsf{pp}, \mathbb{x}_1, \mathbb{w}_1) \in \mathscr{R}_1$ *to a new statement* $(\mathsf{pp}, \mathbb{x}_2, \mathbb{w}_2) \in \mathscr{R}_2$.
- $\mathcal{V}(\mathsf{pp}, \mathbb{x}_1) \rightarrow \mathbb{x}_2$*: Takes as input public parameters* pp*, and instance* $\mathbb{x}_1$ *associated with* $\mathscr{R}_1$*. Interactively reduces the task of checking* $\mathbb{x}_1$ *to the task of checking a new instance* $\mathbb{x}_2$ *associated with* $\mathscr{R}_2$.

Let $\langle \mathcal{P}(\mathsf{pp}, \mathbb{x}_1, \mathbb{w}_1), \mathcal{V}(\mathsf{pp}, \mathbb{x}_1)\rangle \rightarrow (\mathbb{x}_2, \mathbb{w}_2)$ denote the output of the prover in the interaction between $\mathcal{P}$ and $\mathcal{V}$. The reduction of knowledge $\Pi := (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ must satisfy the following properties:

1. **Completeness:** For any PPT adversary $\mathcal{A}$, given $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, N)$ and $(\mathbb{x}_1, \mathbb{w}_1) \leftarrow \mathcal{A}(\mathsf{pp})$ such that $(\mathsf{pp}, \mathbb{x}_1, \mathbb{w}_1) \in \mathscr{R}_1$, we have that
$$\mathbb{x}_1 = \mathbb{x}_2, \qquad (\mathsf{pp}, \langle \mathcal{P}(\mathsf{pp}, \mathbb{x}_1, \mathbb{w}_1), \mathcal{V}(\mathsf{pp}, \mathbb{x}_1)\rangle) \in \mathscr{R}_2$$

2. **Knowledge soundness:** For all expected poly-time stateful adversary $\mathcal{A} := (\mathcal{A}_1, \mathcal{A}_2)$ there exists an expected poly-time extractor Ext such that $\mathsf{Adv}^{\mathsf{Adap\text{-}KS}}_{\mathsf{ARG},\mathsf{Ext},\mathcal{A}}(\lambda) :=$
$$\Pr\left[\begin{array}{c|l} (\mathsf{pp}, \langle \mathcal{A}_2(\mathsf{pp}), \mathcal{V}(\mathsf{pp}, \mathbb{x}_1)\rangle) \in \mathscr{R}_2 & \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, N) \\ \wedge & \mathbb{x}_1 \leftarrow \mathcal{A}_1(\mathsf{pp}) \\ (\mathsf{pp}, \mathbb{x}_1, \mathbb{w}_1^*) \notin \mathscr{R}_1 & \mathbb{w}_1^* \leftarrow \mathsf{Ext}^{\mathcal{A}}(\mathsf{pp}) \end{array}\right] = \mathrm{negl}(\lambda)$$

3. **Public reducibility:** There exists a deterministic poly-time function φ such that for all expected poly-time stateful adversary $\mathcal{A} := (\mathcal{A}_1, \mathcal{A}_2)$, given $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, N)$, $\mathbb{x}_1 \leftarrow \mathcal{A}_1(\mathsf{pp})$ and $(\mathbb{x}_2, \mathbb{w}_2) \leftarrow \langle \mathcal{A}_2(\mathsf{pp}), \mathcal{V}(\mathsf{pp}, \mathbb{x}_1)\rangle$ with interaction transcript tr, we have that $\varphi(\mathsf{pp}, \mathbb{x}_1, \mathsf{tr}) = \mathbb{x}_2$.

We write $\Pi : \mathscr{R}_1 \rightarrow \mathscr{R}_2$ to denote that protocol Π is a reduction of knowledge from relation $\mathscr{R}_1$ to relation $\mathscr{R}_2$. It is easy to see that an argument of knowledge is a reduction of knowledge that reduces a relation $\mathscr{R}$ to $\mathscr{R}_\top := \{(\mathrm{true}, \bot)\}$.

Tree extraction. We also recall the following lemma from [KP23], inspired by a theorem originally in [BCS21]. This lemma is crucial in proving the security of our protocols.

Lemma 2.6 (Lemma 6 [KP23])**.** *Consider an m-round public-coin interactive protocol* $(\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ *that satisfies the interface described in Section 2.6 and satisfies completeness and public reducibility. Then* $(\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ *is a reduction of knowledge if there exists a PPT extractor* Ext *that, for all instances $\mathbb{x}_1$, outputs a satisfying witness $\mathbb{w}_1^*$ with probability at least $1 - \mathrm{negl}(\lambda)$, given an $(n_1, \ldots, n_m)$-tree of accepting transcripts for $\mathbb{x}_1$ where the verifier's randomness is sampled from space Q such that $|Q| = O(2^\lambda)$, and $\prod_i n_i = \mathrm{poly}(\lambda)$.*

We do not recall the definition of an $(n_1, \ldots, n_m)$-tree of accepting transcripts, as we only use this lemma for 1-round public-coin interactive protocols. An n-tree of accepting transcripts is a list of n instance-witness pairs $[(\mathbb{x}_{2,i}, \mathbb{w}_{2,i})]_{i=1}^n$ output by the interaction $\langle \mathcal{A}_2(\mathsf{pp}, \mathsf{state}), \mathcal{V}(\mathsf{pp}, \mathbb{x}_1)\rangle$ for unique public verifier challenges α_i such that $(\mathsf{pp}, \mathbb{x}_{2,i}, \mathbb{w}_{2,i}) \in \mathscr{R}_2$ for all $i \in [n]$.

Sequential composition. We now recall a theorem that shows that the sequential composition of two reductions of knowledge yields another reduction of knowledge.

Theorem 2.7 (Theorem 5 [KP23])**.** *Consider ternary relations $\mathscr{R}_1$, $\mathscr{R}_2$, and $\mathscr{R}_3$. For reductions of knowledge $\Pi_1 = (\mathsf{Setup}, \mathcal{P}_1, \mathcal{V}_1) : \mathscr{R}_1 \rightarrow \mathscr{R}_2$ and $\Pi_2 = (\mathsf{Setup}, \mathcal{P}_2, \mathcal{V}_2) : \mathscr{R}_2 \rightarrow \mathscr{R}_3$, we have that $\Pi_2 \circ \Pi_1 = (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ is a reduction of knowledge from $\mathscr{R}_1$ to $\mathscr{R}_3$ where*
$$\mathcal{P}(\mathsf{pp}, \mathbb{x}_1, \mathbb{w}_1) := \mathcal{P}_2(\mathsf{pp}, \mathcal{P}_1(\mathsf{pp}, \mathbb{x}_1, \mathbb{w}_1))$$
$$\mathcal{V}(\mathsf{pp}, \mathbb{x}_1) := \mathcal{V}_2(\mathsf{pp}, \mathcal{V}_1(\mathsf{pp}, \mathbb{x}_1))$$

Additionally, [Lee21] (Lemma 4) shows that sequential composition of two arguments that satisfy tree extraction is also tree-extractable.

3 Groups of unknown order

At a high level, groups of unknown order are groups where the order of the group is unknown to the adversary but group operations as well as sampling random group elements can be performed efficiently. These primitives as well as the associated assumptions are mainly useful in the setting when unbounded integers need to be committed to or when relations need to be verified over the integers. These groups have also been used in constructions of accumulators [BM94; BBF19; LLX07], verifiable delay functions [Pie19; Wes19] and range proofs [Lip03; CPP17; CGKR22].

Candidate GUOs. Some candidates for groups of unknown order include RSA groups [RSW96], class groups [Lip12; BW88] and Jacobins of hyperelliptic curves [Bre00]. While RSA groups require a trusted setup to generate the RSA modulus, the other two candidates do not require a trusted setup. Of the latter two, we use hyperelliptic curves as they have the shortest representation (for a given security level) and are believed to satisfy all the assumptions we require. Importantly, class groups do not satisfy the Strong RSA assumption, stated in Assumption 3.6, as it is possible to compute square roots efficiently in these groups.

Security parameter. Throughout the paper, we use 2λ as the security parameter for random challenges in the group of unknown order, where λ is the security parameter for the cryptographic primitives we are constructing. This is due to the existence of sub-exponential algorithms[DGS22] that can break assumptions in these groups such as the adaptive root assumption (see Definition A.1). This requires all PoKE-style protocols to have a challenge space of size $\Theta(2^{2\lambda})$ to be secure against these attacks.

In the next section, we describe the generic group model (GGM), an idealized group model for GUOs. After that, we describe the falsifiable assumptions we require, and show that they all hold in the GGM.

3.1 The generic group model

The Generic Group Model, first defined in [DK02b] and used in prior work such as [BBF19; AGLMS23], has a group sampler GGen that on input a security parameter outputs the description $\mathbb{G}$ of a group with an (unknown) order that is sampled uniformly from a public range $[A, B]$, such that $1/|A|= \mathrm{negl}(\lambda)$ and $1/|B-A|= \mathrm{negl}(\lambda)$. The group description consists of a random injective labeling function $\sigma : \mathbb{Z}_{|\mathbb{G}|} \to 2^{\ell}$, where $2^{\ell} \gg |\mathbb{G}|$, such that the labels $\sigma(\cdot)$ are the only representations of group elements visible to algorithm in the model, known as generic group algorithms.

In particular, a generic group algorithm $\mathcal{A}$ is a PPT algorithm with a list L that contains the labels of all the group elements given to $\mathcal{A}$ as input. Note that it does not get the group order as input. $\mathcal{A}$ is then allowed to query two oracles $\mathcal{O}_1$ and $\mathcal{O}_2$. $\mathcal{O}_1$ samples a random $r \overset{\$}{\leftarrow} \mathbb{Z}_{|\mathbb{G}|}$ and returns $\sigma(r)$ which is appended to L. $\mathcal{O}_2(i, j, \pm)$ takes two labels $i, j \in [L]$ and a sign bit. For the group elements H_i, H_j corresponding to these labels, it returns the label $\sigma(H_i \pm H_j)$, which is also then appended to L. Intuitively, this formulation of a generic algorithm allows adversaries to perform operations on group elements and sample random group elements without knowing the order or structure of the group. The GGM is not a falsifiable assumption.

3.2 Assumptions

Our constructions rely on three standard falsifiable assumptions about GUOs: the discrete logarithm assumption (see Definition 3.1), the strong RSA assumption (see Definition 3.6), and the modular consistency assumption (see Definition 3.4).

These three assumptions are the strongest on which our constructions are based. To simplify the assumption "zoo" and reduce the total number of assumptions to three, we defer a few weaker assumptions used in our proofs to the appendix (see Appendix A) and show that they are implied by these three assumptions. All of our assumptions are shown to hold in the Generic Group Model (see Section 3.1). See Fig. 1 for a summary of the assumptions and their relationships.

New Assumptions. We introduce two new assumptions in this paper: the modular consistency assumption (see Definition 3.4), which is used in the main body, and the multi-rational root assumption (see Appendix A.1.4), which appears in the appendix and is implied by the strong RSA assumption (see Definition 3.6). These two new assumptions are tailored to our setting but are likely to have applications in other contexts as well.

3.2.1 Discrete logarithm assumption

The discrete logarithm assumption states that for any $n \in \mathbb{N}$, no efficient adversary $\mathcal{A}$, given the group $\mathbb{G} \leftarrow \mathsf{GGen}(1^\lambda)$ and n randomly sampled generators $\mathbf{G} \leftarrow \mathbb{G}^n$, succeeds with non-negligible probability at outputting $(\alpha_1, \dots, \alpha_n, \beta_1, \dots, \beta_n) \in \mathbb{Z}^{2n}$ such that $\sum_{i=1}^n \alpha_i \cdot G_i = \sum_{i=1}^n \beta_i \cdot G_i$ and $\alpha_i \neq \beta_i$, for some i. A more formal statement is as follows.

Definition 3.1 (Discrete logarithm assumption [BBF19]). *The discrete logarithm assumption holds for* GGen *if for any* $n \in \mathbb{N}$, *every efficient adversary* $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{DLOG}}_{\mathcal{A},n}(\lambda) \leq \mathrm{negl}(\lambda)$, *where* $\mathsf{Adv}^{\mathsf{DLOG}}_{\mathcal{A},n}(\lambda) :=$

$$\Pr\left[\begin{array}{c|l} \sum_{i=1}^n \alpha_i \cdot G_i = \sum_{i=1}^n \beta_i \cdot G_i & \mathbb{G} \leftarrow \mathsf{GGen}(1^\lambda) \\ \wedge & \mathbf{G} \leftarrow \mathbb{G}^n \\ \exists i \in [n] : \alpha_i \neq \beta_i & (\alpha_1, \dots, \alpha_n, \beta_1, \dots, \beta_n) \leftarrow \mathcal{A}(\mathbb{G}, \mathbf{G}) \end{array}\right]$$

Lemma 3.2. *[BBF19] The Discrete logarithm assumption (see Definition 3.1) is secure in the Generic Group Model (GGM, Section 3.1).*

3.2.2 New assumption: modular consistency assumption

We now present our first new assumption, the modular consistency assumption, which is inspired by ideas in [LPS24]. At a high level, it states that, given n randomly sampled generators $\mathbf{G} \overset{\$}{\leftarrow} \mathbb{G}^n$, the best an adversary $\mathcal{A}$ that runs in T time can do is output a group element U obtained by performing a sequence of T multiplications on pairs of group elements it has received or computed so far. We show in Lemma A.2 that the modular consistency assumption implies the adaptive root assumption (see Definition A.1).

Before we state the assumption we recall the Chinese remainder theorem, which is used routinely in the 'groups of unknown order' setting.

Theorem 3.3 (Chinese remainder theorem). *Let* $\ell_1, \dots, \ell_T$ *be coprime integers and let* $r_1, \dots, r_T \in \mathbb{Z}$, *then there exists a unique* $0 \leq x < \prod_{i=1}^T \ell_i$ *such that* $x = r_i \mod \ell_i$ *and there is an efficient algorithm for computing* $x \leftarrow \mathsf{CRT}([r_i]_{i=1}^T, [\ell_i]_{i=1}^T)$.

Now, we state the modular consistency assumption.

Definition 3.4 (Modular consistency assumption). *Let* $\mathsf{rep} = \mathrm{poly}(\lambda)$ *be the number of bits used to represent the group elements of a group generated by* GGen. *The modular consistency assumption holds for* GGen *if for* *all* $n = \mathrm{poly}(\lambda)$*, all* $T = \mathrm{poly}(\lambda)$*, and for every* $(T \cdot \mathsf{rep})$*-step stateful Turing machine* $\mathcal{A}_1$ *and every PPT adversary* $\mathcal{A}_2$*, we have* $\mathsf{Adv}^{\mathsf{MC}}_{T,\mathcal{A},n}(\lambda) \leq \mathrm{negl}(\lambda)$*, where* $\mathsf{Adv}^{\mathsf{MC}}_{T,\mathcal{A},n}(\lambda) :=$

$$\Pr\left[\begin{array}{c|l} \forall i \in [T] \ : \ \langle \boldsymbol{r}_i, \boldsymbol{G}\rangle + \ell_i \cdot Q_i = U & \mathbb{G} \leftarrow \mathsf{GGen}(1^\lambda) \\ \textit{s.t. } \boldsymbol{G} := (G_1, \ldots, G_n) \in \mathbb{G} & \mathbf{G} \xleftarrow{\$} \mathbb{G}^n \\ \wedge & U \leftarrow \mathcal{A}_1(\mathbb{G}, \mathbf{G}) \\ \langle \boldsymbol{x}, \boldsymbol{G}\rangle \neq U & [\boldsymbol{r}_i, Q_i, \ell_i]_{i=1}^T \leftarrow \mathcal{A}_2 \textit{ s.t. } \boldsymbol{r}_i = [r_{i,j}]_{j\in[n]} \\ & \ell_1, \ldots, \ell_T \textit{ are coprime integers} \\ & \forall j \in [n] : x_j \leftarrow \mathsf{CRT}([r_{i,j}]_{i\in[T]}, [\ell_i]_{i=1}^T) \end{array}\right]$$

Lemma 3.5. *The modular consistency assumption (see Definition 3.4) is secure in the Generic Group Model (GGM, see Section 3.1).*

Proof. We defer the proof to Appendix A.2.

3.2.3 Strong RSA assumption

At a high level, the strong RSA assumption states that no efficient adversary $\mathcal{A}$, with non-negligible probability, can compute a non-trivial ℓ^{th} root ($\ell > 1$) of a randomly sampled group element $G \xleftarrow{\$} \mathbb{G}$. A more formal statement is as follows.

Definition 3.6 (Strong RSA assumption [FO97; BP97; CPP17])**.** *The strong RSA assumption holds for* GGen *if for all efficient* $\mathcal{A}$*,* $\mathsf{Adv}^{\mathsf{SRSA}}_{\mathcal{A}}(\lambda) \leq \mathrm{negl}(\lambda)$*, where*

$$\mathsf{Adv}^{\mathsf{SRSA}}_{\mathcal{A}}(\lambda) := \Pr\left[\begin{array}{c|l} \ell \cdot U = G & \mathbb{G} \leftarrow \mathsf{GGen}(1^\lambda) \\ \wedge & G \xleftarrow{\$} \mathbb{G} \\ \ell > 1 & (U, \ell) \in \mathbb{G} \times \mathbb{N} \leftarrow \mathcal{A}(\mathbb{G}, G) \end{array}\right]$$

Lemma 3.7. *[DK02a] The strong RSA assumption (see Definition 3.6) is secure in the Generic Group Model (GGM, Section 3.1).*

4 Techniques

In this section, we present a technical overview of the sub-protocols employed in our PCS. We start by giving an intuitive but high-level description of each of our sub-protocols.

DewTwo The DewTwo PCS reduces the task of proving the evaluation of a committed multilinear polynomial $p(\mathbf{X}) \in \mathbb{F}_q[X_1, \ldots, X_\mu]$ at a point $\mathbf{z} \in \mathbb{F}_q^\mu$, to the task of committing to a vector of integers $\boldsymbol{u} \in \mathbb{Z}^M$ and proving two statements: (i) some of these integers are within a certain range (ii) and a certain function $f(\boldsymbol{u})$ evaluates to 0. We provide more details in Section 4.1.

PoKEDEx The PoKEDEx protocol allows us to do precisely this: convince a verifier, for a vector $\boldsymbol{u} \in \mathbb{Z}^M$ (succinctly) committed to in a commitment C, that $f(\boldsymbol{u}) = 0$, for some function f, and that $u_i \in [a_i, b_i]$, for each $i \in [\mathcal{I}]$ where $\mathcal{I} \subseteq [M]$ indicates a set of indices being range-checked. In the final version of DewTwo, we use TPoKEDex, an optimized and specialized PoKEDEx.

PoKEMath As a stepping stone to PoKEDEx (and TPoKEDex), but also of independent applicability, we construct the PoKEMath protocol. It allows a prover to convince a verifier holding a commitment C, that $f(\boldsymbol{u}) = 0$, for some public function f and vector $\boldsymbol{u} \in \mathbb{Z}^M$ committed within C. Notice that PoKEMath can directly be used as a PoKEDEx (by having the function f additionally verify the range claims), but the resulting PoKEDEx has a communication complexity that is too large for our PCS. We optimize this naive approach using the following SIPA scheme.

SIPA The self-inner product argument (SIPA) protocol, is a bulletproof-style protocol, that enables a prover to convince a verifier who holds a commitment C to a vector $\boldsymbol{u} \in \mathbb{Z}^M$, that the inner product of $\boldsymbol{u}$ with itself satisfies $\langle \boldsymbol{u}, \boldsymbol{u} \rangle \mod \ell = v$. Importantly, the communication complexity between the prover and verifier is only $O(\log M)$.

4.1 DewTwo

Our multilinear polynomial commitment scheme, DewTwo, builds upon Dew [AGLMS23], a transparent PCS based on groups of unknown order (see Section 3). In this section, we provide a high level overview of how our PCS works.

Vector inner products suffice. We start by observing that one can commit to a multilinear polynomial and argue about its evaluations by instead committing to its vector of coefficients and considering the inner product of this vector with a public vector derived from the evaluation point. Recall that the multilinear polynomial $p(\mathbf{X}) = \sum_{i=0}^{N-1} p_i \prod_{j=1}^{\mu} X_j^{\mathsf{bin}(i,j)} \in \mathbb{F}_q^{\leq 1}[X_1, \ldots, X_\mu]$ can equivalently be represented by its vector of coefficients $\boldsymbol{p} = [p_i]_{i=0}^{N-1} \in \mathbb{F}_q^N$, where $N := 2^\mu$. More formally, there exists a bijective mapping between the space of polynomials $\mathbb{F}_q^{\leq 1}[X_1, \ldots, X_\mu]$ and the corresponding space of vectors $\mathbb{F}_q^N$. Furthermore, given a commitment to the vector $\boldsymbol{p}$, polynomial evaluation $y = p(\mathbf{z})$ at a point $\mathbf{z} \in \mathbb{F}_q^\mu$ can be equivalently expressed as the inner product $y = \langle \boldsymbol{p}, \mathbf{z}^{\mathsf{ext}} \rangle$, where the tensor product $\mathbf{z}^{\mathsf{ext}} \in \mathbb{F}_q^N$ is defined as $z_i^{\mathsf{ext}} := \prod_{j=1}^{\mu} z_j^{\mathsf{bin}(i,j)}$ for all $i \in [0, \ldots, N-1]$. Thus, it suffices to show how to commit to a vector of field elements $\boldsymbol{p} \in \mathbb{F}_q^N$ and prove its inner product with such a (public) vector $\mathbf{z}^{\mathsf{ext}} \in \mathbb{F}_q^N$ derived from an evaluation point.

Committing to vectors. Given a group of unknown order $\mathbb{G}$ and a generator $G \in \mathbb{G}$, to commit to a vector $\boldsymbol{p} \in \mathbb{F}_q^N$, we first encode the vector as an integer a via an injective and *invertible* encoding that maps $\mathbb{F}_q^N \to \mathbb{Z}$.

We then commit to the integer a as $\mathsf{cm} := a \cdot G$. The latter is a binding commitment to integers in the GUO setting (Theorem 5.2), and we show how to do the integer encoding below.

Given a vector $\boldsymbol{p} = [p_i]_{i=0}^{N-1} \in \mathbb{F}_q^N$, consider the univariate integer polynomial $\tilde{p}(X) \in \mathbb{Z}^{<N}[X]$ with $\tilde{\boldsymbol{p}}$ as its coefficient vector. For any integer $\alpha > q$, $\tilde{p}(\alpha) = \sum_{i=0}^{N-1} \tilde{p}_i \alpha^i \in \mathbb{Z}$ is an injective encoding of $\boldsymbol{p}$. We also have that it is invertible: given $\tilde{p}(\alpha)$, one can obtain $\tilde{\boldsymbol{p}}$ by decomposing the integer $\tilde{p}(\alpha)$ into its base-α representation to obtain the integer vector $[\tilde{p}_i]_{i=0}^{N-1}$. It is easy to see that the corresponding field vector $\boldsymbol{p} = [p_i]_{i=0}^{N-1}$ is the pre-image of $\tilde{p}(\alpha)$.

Thus, one can commit to a field vector $\boldsymbol{p} \in \mathbb{F}_q^N$ as $\mathsf{cm} := \tilde{p}(\alpha) \cdot G$.

Inner product proofs. We start with an observation: given two vectors $\boldsymbol{p}, \mathbf{r} \in \mathbb{F}_q^N$, the product $\tilde{p}(\alpha) \cdot \tilde{r}(\alpha)$ can be expressed as follows:

$$\begin{aligned}
\tilde{p}(\alpha) \cdot \tilde{r}(\alpha) &= \left(\tilde{p}_0 + \tilde{p}_1\alpha + \cdots + \tilde{p}_{N-1}\alpha^{N-1}\right) \cdot \left(\tilde{r}_0 + \tilde{r}_1\alpha + \cdots + \tilde{r}_{N-1}\alpha^{N-1}\right) \\
&= \sum_{i=0}^{N-2} \left(\sum_{j+k=i} \tilde{p}_j \tilde{r}_k \right) \alpha^i + \sum_{j=0}^{N-1} \tilde{p}_j \tilde{r}_{N-1-j} \alpha^{N-1} + \sum_{i=N}^{2N-2} \left(\sum_{j+k=i} \tilde{p}_j \tilde{r}_k \right) \alpha^i \\
&:= s + (\tilde{p}_0\tilde{r}_{N-1} + \tilde{p}_1\tilde{r}_{N-2} + \cdots + \tilde{p}_{N-1}\tilde{r}_0) \cdot \alpha^{N-1} + u \cdot \alpha^N
\end{aligned}$$

Importantly, the middle coefficient (corresponding to α^{N-1}) is $\langle \tilde{\boldsymbol{p}}, \bar{\tilde{\mathbf{r}}} \rangle$, the inner product of the integer castings of $\boldsymbol{p}$ and $\bar{\mathbf{r}}$ (the reverse of the vector $\mathbf{r}$). It is easy to see that $\langle \boldsymbol{p}, \bar{\mathbf{r}} \rangle = \langle \tilde{\boldsymbol{p}}, \bar{\tilde{\mathbf{r}}} \rangle \mod q$. One might thus hope that by decomposing the product $\tilde{p}(\alpha) \cdot \tilde{r}(\alpha)$ into its base-α representation and picking out the middle coefficient, we can obtain $\langle \boldsymbol{p}, \bar{\mathbf{r}} \rangle$. However, for this to hold, α must not only be larger than q, but must also be larger than $\max_i \left(\sum_{j+k=i} \tilde{p}_j \tilde{r}_k \right)$. When this constraint holds, one can also verify that the terms s and u satisfy $0 \leq s \leq \alpha^{N-1}$ and $0 \leq u \leq \alpha^{N-1}$.

This gives a potential protocol for proving an inner product claim about the vector committed to in $\mathsf{cm} = \tilde{p}(\alpha) \cdot G$ and a public vector $\mathbf{r}$. Defining $\boldsymbol{v} := \bar{\mathbf{r}}$, the prover can convince the verifier that $\langle \boldsymbol{p}, \mathbf{r} \rangle = \langle \boldsymbol{p}, \bar{\boldsymbol{v}} \rangle = y$ by sending three values s, t, u to the verifier such that the following equation holds:

$$\tilde{p}(\alpha) \cdot \tilde{v}(\alpha) = s + t \cdot \alpha^{N-1} + u \cdot \alpha^N \tag{1}$$

The verifier can then check that the appropriate bounds on s, t, u hold, check that $y = t \mod q$ and compute $\tilde{v}(\alpha)$ on its own[4]. Finally, it checks that:

$$\begin{aligned}
\tilde{v}(\alpha) \cdot \mathsf{cm} &= s \cdot G + (t \cdot \alpha^{N-1}) \cdot G + (u \cdot \alpha^N) \cdot G && (2) \\
\implies (\tilde{p}(\alpha) \cdot \tilde{v}(\alpha)) \cdot G &= (s + t \cdot \alpha^{N-1} + u \cdot \alpha^N) \cdot G && (3)
\end{aligned}$$

This assures the verifier that Eq. (1) holds over the integers, except with negligible probability, under the discrete logarithm assumption.

Roadblocks. Unfortunately, the aforementioned protocol is only sound when the elements of $\tilde{\boldsymbol{p}}$ and $\tilde{\mathbf{r}}$ are indeed in the range $[0, q-1]$. While the verifier can ensure the elements of $\tilde{\mathbf{r}}$ are bounded, if the prover chooses to use integers from outside this range while computing cm, even if the check in Eq. (2) holds, the claimed inner product t may be incorrect. One illustrative concern is that $\langle \tilde{\boldsymbol{p}}, \tilde{\mathbf{r}} \rangle = \sum_{j=0}^{N-1} \tilde{p}_j \tilde{r}_j$ could be

[4]This can be done efficiently when the inner product corresponds to PCS evaluation, as we show later.

larger than α (due to the malicious unbounded integers) and thus 'overflow'. That is, the prover could now send a malicious inner product $t' := \langle \tilde{\boldsymbol{p}}, \tilde{\mathbf{r}} \rangle \mod \alpha \neq \langle \tilde{\boldsymbol{p}}, \tilde{\mathbf{r}} \rangle$ to the verifier, and the following instance of the check in Eq. (2) would still pass, for appropriate 'malicious' values s', u':

$$\tilde{v}(\alpha) \cdot \mathsf{cm} = s' \cdot G + (t' \cdot \alpha^{N-1}) \cdot G + (u' \cdot \alpha^N) \cdot G$$

As an example, let $N := 2$, $q := 5$, $\alpha := 100$, the *out of bound* vector $\tilde{\boldsymbol{p}} := [1, 98]$ and the honest vector $\tilde{\mathbf{r}} := [3, 4]$. Then, $\tilde{p}(100) = 1 + 98 \cdot 100 = 9801$ and $\tilde{r}(100) = 3 + 4 \cdot 100 = 403$. The product $\tilde{p}(100) \cdot \tilde{r}(100) = 3949803$ when expressed in its base-α representation gives $3 + 98 \cdot 100 + 394 \cdot 100^2$. However, the inner product $\langle \boldsymbol{p}, \mathbf{r} \rangle = 1 \cdot 4 + 98 \cdot 3 = 298 \neq 98$. Clearly, the prover could cheat by sending $s' = 3$, $t' = 298 \mod \alpha = 98$ and $u' = 394$ in this instance.

Ensuring bounded elements. At a high level, we show that for a randomly sampled $\tilde{\boldsymbol{\gamma}} \in [0, q-1]^N$, if the middle coefficient of $\tilde{p}(\alpha) \cdot \tilde{\gamma}(\alpha)$ is small, then $\tilde{\boldsymbol{p}}$ is 'bounded' (with a caveat that we explain shortly) with high probability. The verifier can first perform such a check using the above techniques, and then use the guarantee that $\tilde{\boldsymbol{p}}$ is bounded to perform the inner product check with respect to $\tilde{\mathbf{r}}$. This uses the LCSZ lemma from [BF23], which can be thought of as a generalization of the Schwartz-Zippel lemma to the case of composite moduli and multilinear polynomials. In addition, this also generalizes the batch shortness test techniques based on random linear combinations/subset sums used in works like [CKLR21; CGKR22; BBCdGL18] by using only logarithmic randomness. This crucially allows the verifier to be efficient, which is not the case when using a linear number of random elements.

Our result is built on two crucial observations:

- While the overflow in the above example was indeed due to a large element in $\tilde{\boldsymbol{p}}$, it depends on the choice of the vector $\tilde{\mathbf{r}}$ as well, which the verifier can control.
- In the honest case, where $\tilde{\boldsymbol{p}}, \tilde{\mathbf{r}} \in [0, q-1]^N$, it is the case that the middle coefficient of $\tilde{p}(\alpha) \cdot \tilde{\gamma}(\alpha)$, $\langle \tilde{\boldsymbol{p}}, \tilde{\mathbf{r}} \rangle$, is always $< Nq^2$.

Let $\beta = Nq^2$, by setting $\alpha > \beta$[5], we show that the fact that the middle coefficient of $\tilde{p}(\alpha) \cdot \tilde{\gamma}(\alpha)$ is smaller than β, for $\tilde{\boldsymbol{\gamma}} \xleftarrow{\$} [0, q-1]^N$, implies that $\boldsymbol{p}$ is 'bounded'. To illustrate, suppose the prover chooses a single element of $\tilde{\boldsymbol{p}}$, say $\tilde{p}_i$ to be large, and the rest to be zero. The inner product $\langle \tilde{\boldsymbol{p}}, \tilde{\boldsymbol{\gamma}} \rangle = \tilde{p}_i \cdot \tilde{\gamma}_i$ could be large and overflow, causing the middle coefficient to instead be $\tilde{p}_i \cdot \tilde{\gamma}_i \mod \alpha$. The above check implies that $\tilde{p}_i \cdot \tilde{\gamma}_i = s_i \mod \alpha$ for some $s_i < \beta$, which gives that $\tilde{p}_i = \frac{s_i}{\tilde{\gamma}_i} \mod \alpha$. Although we can only guarantee that the prover must stick to $\tilde{p}_i$ has a bounded rational representation modulo α (with a small numerator and denominator), we show that it suffices that the prover cannot arbitrarily use large coefficients and must stick to such bounded rationals.

We show in Theorem 5.5 that the above ideas can be extended to the general case, where arbitrarily many elements of $\tilde{\boldsymbol{p}}$ are non-zero. While the details are too involved for a technical overview, at a high level, we assume WLOG that $\tilde{\boldsymbol{p}} \in \mathbb{Q}^N$. We then show that if the middle coefficient of $\tilde{p}(\alpha) \cdot \tilde{\gamma}(\alpha) < \beta$, for $\tilde{\boldsymbol{\gamma}} \xleftarrow{\$} [0, q-1]^N$, then the elements of $\tilde{\boldsymbol{p}}$ must all be bounded rationals (modulo α). Although we can only guarantee that the prover must stick to bounded rationals and cannot use arbitrarily large integers, Corollary 5 in [BF23] shows that this is sufficient to ensure that the commitment can only be opened to a vector from $\mathbb{F}_q^N$ (see Section 5.1 for details). In our security proof, we further guarantee that if all our checks pass, this vector must indeed satisfy the required inner product constraint.

[5]Looking ahead, we actually require that $\alpha \gg \beta$ due to some additional subtleties regarding the binding property of the commitment scheme.

As an additional contribution, we use the inverse LCSZ lemma from [BF23] to obtain stronger bounds on these rationals than prior work, Dew [AGLMS23]. This allows us to choose a much smaller value for α, significantly improving the concrete efficiency of our scheme.

Verifier efficiency. Recall that the PCS verifier, to be convinced that $p(\mathbf{z}) = y$ for the vector $\boldsymbol{p}$ committed to in cm, needs to check that $\langle \boldsymbol{p}, \mathbf{z}^{\mathsf{ext}} \rangle = y$. This is done using the aforementioned 'inner product commitment scheme', and we would like the verifier to run in time that is sublinear in N, the size of the polynomial. The most expensive verifier steps are (a) computing $\tilde{v}(\alpha)$, where $\boldsymbol{v} := \overline{\mathbf{z}^{\mathsf{ext}}}$, (b) performing the range checks (on s, t and u) and (c) performing the expensive exponentiations $\tilde{v}(\alpha) \cdot \mathsf{cm}$, $s' \cdot G$, $(t' \cdot \alpha^{N-1}) \cdot G$ and $(u' \cdot \alpha^{N}) \cdot G$.

It is plausible that computing $\tilde{v}(\alpha)$ might take $O(N)$ time, as $\boldsymbol{v}$ is a vector of length N. However, we show in Remark 5.4 that this can in fact be done in just $O(\mu) = O(\log N)$ time, by exploiting the fact that $\boldsymbol{v}$ is the tensor product of a succinct evaluation point $\mathbf{z} \in \mathbb{F}^{\mu}$[6].

Reduction to PoKEDex. We capture the range checks, the exponentiations, as well as performing the check in Eq. (1) in a single protocol called PoKEDEx (see Section 4.4). At a high level, PoKEDEx allows the prover to commit to an integer vector $(\tilde{p}(\alpha), s, t, u)$ using a Pedersen commitment, and prove to the verifier that the integers s, t, and u lie in the appropriate ranges *and* that the following equations hold:

$$\begin{aligned} \tilde{p}(\alpha) \cdot \tilde{v}(\alpha) &= s + t \cdot \alpha^{N-1} + u \cdot \alpha^{N}, \quad \text{and} \\ \tilde{p}(\alpha) \cdot \tilde{\gamma}(\alpha) &= s' + t' \cdot \alpha^{N-1} + u' \cdot \alpha^{N} \end{aligned}$$

where $\boldsymbol{\gamma} \xleftarrow{\$} [0, q-1]^{\mu}$, $y = t \mod q$, and $t' < \beta$. We present the full construction in Section 5.2. In the upcoming sections, we provide a technical overview of the PoKEDEx protocol, preceded by several preliminary protocols that are used in its construction.

4.2 Self inner product arguments (SIPA)

The self inner product argument (SIPA), for a *large* prime $\ell > 2^{\lambda}$, allows a prover $\mathcal{P}$ to convince a verifier $\mathcal{V}$ that the inner product $\langle \mathbf{r}, \mathbf{r} \rangle = v \mod \ell$, where $\mathbf{r} \in \mathbb{Z}^{N}$ is a private vector committed to in a Pedersen-like commitment $C := \langle \mathbf{r}, \mathbf{G} \rangle$[7], and $\mathcal{P}$ and $\mathcal{V}$ both share: (1) the commitment key consisting of a vector of group generators $\mathbf{G} \in \mathbb{G}^{N}$, (2) the Pedersen commitment $C \in \mathbb{G}$, and (3) the claimed inner product value $v \in \mathbb{Z}_{\ell}$. What makes SIPA interesting is that the prover can do this by communicating only $O(\log N)$ many elements of $\mathbb{G}$ and $\mathbb{Z}_{\ell}$ to the verifier. We leverage this to reduce the communication complexity of our PoKEDEx protocol.

Definition 1 (informal). *We define the ternary relation $\mathscr{R}_{\mathsf{SIPA}}$ to consist of tuples of the form* $(\mathsf{pp} = (N, \mathbf{G}), \mathbb{x} = (C, \ell, v), \mathbb{w} = \mathbf{r})$ *such that* $C = \langle \mathbf{r}, \mathbf{G} \rangle$ *and* $v = \langle \mathbf{r}, \mathbf{r} \rangle \mod \ell$.

Construction. WLOG we can assume the length of the vector is a power of 2, $N = 2^{n}$ (we can round it up to the closest power of 2). Our SIPA construction follows the typical blueprint used to construct inner product arguments (first introduced in [BCCGP16]): we construct a reduction of knowledge (formalized in [KP23])

$$\mathsf{SIPA.Reduce}(\langle \mathcal{P}((2^{n}, \mathbf{G}), \mathbb{x}, \mathbb{w}), \mathcal{V}((2^{n}, \mathbf{G}), \mathbb{x}) \rangle) \rightarrow ((2^{n-1}, \mathbf{G}'), \mathbb{x}', \mathbb{w}')$$

[6]The verifier also needs to compute $\tilde{\gamma}(\alpha)$ efficiently, and we show that this can be done analogously, by showing that it suffices for $\tilde{\boldsymbol{\gamma}}$ to also be a tensor product of a random vector sampled from $[0, q-1]^{\mu}$. We omit the details here for brevity.

[7]These Pedersen-like commitments are binding in the GUO setting, under the discrete logarithm assumption.

SIPA.Reduce, with only a constant amount of communication between $\mathcal{P}$ and $\mathcal{V}$, reduces the task of checking if $(\mathbb{x}, \mathbb{w}) \in \mathscr{R}_{\mathsf{SIPA}}^{(2^n, \mathbf{G})}$ to the easier task of checking if $(\mathbb{x}', \mathbb{w}') \in \mathscr{R}_{\mathsf{SIPA}}^{(2^{n-1}, \mathbf{G}')}$. Our SIPA protocol iteratively applies the SIPA.Reduce protocol n times to reduce the task of checking if $(\mathbb{x}, \mathbb{w}) \in \mathscr{R}_{\mathsf{SIPA}}^{(2^n, \mathbf{G})}$ to checking if $(\mathbb{x}', \mathbb{w}') \in \mathscr{R}_{\mathsf{SIPA}}^{(1, G')}$. The prover then directly sends the constant sized $\mathbb{w}' = r'$ to the verifier, who can now verify that $(\mathbb{x}', r') \in \mathscr{R}_{\mathsf{SIPA}}^{(1, G')}$. The total communication is $O(n) = O(\log N)$, as required.

TIPA. In the construction of our polynomial commitment scheme, we would like to perform 3 SIPA protocols of size N each, which would have naively required communicating $3k \log N$ group elements, if one SIPA protocol required communicating $k \log N$ group elements. In order to optimize our proof size, we design TIPA that allows us to do this while communicating just $k \log(3N)$ group elements. We refer the reader to Appendix E.1 for the construction of TIPA, but state the corresponding relation $\mathscr{R}_{\mathsf{TIPA}}$ here.

Definition 2 (informal)**.** *We define the ternary relation* $\mathscr{R}_{\mathsf{TIPA}}$ *to consist of tuples of the form* $(\mathsf{pp} = (3 \cdot N, (\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_3)), \mathbb{x} = (C, \ell, (v_1, v_2, v_3)), \mathbb{w} = (\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3))$ *such that* $C = \langle \mathbf{r}_1, \mathbf{G}_1 \rangle + \langle \mathbf{r}_2, \mathbf{G}_2 \rangle + \langle \mathbf{r}_3, \mathbf{G}_3 \rangle$ *and* $v_i = \langle \mathbf{r}_i, \mathbf{r}_i \rangle \mod \ell$*, for all* $i \in [3]$.

4.3 Proof of Knowledge of Exponent Math (PoKEMath)

PoKEMath protocol allows a prover to demonstrate knowledge of a preimage $\boldsymbol{u} \in \mathbb{Z}^m$, committed as $U = \langle \boldsymbol{u}, \boldsymbol{G} \rangle$, such that $f(\boldsymbol{u}) = 0$, where $f : \mathbb{Z}^m \to \mathbb{Z}$ is an arbitrary polynomial-time computable function. This enables the verifier to succinctly verify a property (characterized by f) of the preimage $\boldsymbol{u}$ without needing to compute $f(\boldsymbol{u})$ directly.

Definition 3 (informal)**.** *We define the ternary relation* $\mathscr{R}_{\mathsf{PoKEMath}}$ *to consist of tuples of the form* $(\mathbf{G}, (U, m, f), \boldsymbol{u})$ *such that* $U = \langle \boldsymbol{u}, \mathbf{G} \rangle$ *and* $f(\boldsymbol{u}) = 0$*, where* $f : \mathbb{Z}^m \to \mathbb{Z}$ *is a polynomial-time computable function.*

PoKEMath builds on the Proof of Knowledge of Exponent Representation (PoKERep) [BBF19], which we refer to as PoKE in this paper for simplicity.

Definition 4 (informal)**.** *We define the relation* $\mathscr{R}_{\mathsf{PoKE}}$ *as the set of tuples* $(\mathbf{G}, (U, m), \boldsymbol{u})$ *such that* $U = \langle \boldsymbol{u}, \mathbf{G} \rangle$.

In PoKE, the verifier sends a challenge $\ell \overset{\$}{\leftarrow} \mathsf{Primes}[\lambda, 2\lambda]$ to the prover. For each $i \in [m]$, the prover uses this challenge to decompose u_i as $u_i = q_i \cdot \ell + r_i$, where $q_i \in \mathbb{Z}$ and $r_i \in [0, \ell - 1]$ for every $i \in [m]$. The prover then commits to $\mathbf{q}$ by computing $Q = \langle \mathbf{q}, \mathbf{G} \rangle$ and sends $(Q, \boldsymbol{r})$ to the verifier. The verifier accepts if $U = \ell \cdot Q + \langle \mathbf{r}, \mathbf{G} \rangle$, which would imply that $\langle \boldsymbol{u}, \mathbf{G} \rangle = \langle \ell \cdot \mathbf{q} + \mathbf{r}, \mathbf{G} \rangle$. Soundness follows from the fact that if this check passes for overwhelmingly many challenges, an extractor could use several 'good' challenges, say $[\ell_i]_{i \in [t]}$, and corresponding remainders $[\mathbf{r}_i]_{i \in [t]}$, and solve for $\boldsymbol{u}$ using the Chinese Remainder Theorem.

In PoKEMath, the verifier additionally checks that $f(\boldsymbol{r}) = 0 \mod \ell$. Essentially, PoKEMath reduces the computation of f over a vector of potentially large integers $\boldsymbol{u}$ to a computation over a vector $\boldsymbol{r}$ of the same length, but with values bounded by the size of the challenge ℓ.

4.4 Proof of Knowledge of Exponent, Decomposition, and Expression (PoKEDex)

A desired protocol for the DewTwo construction is PoKEDEx which is an extension of PoKEMath protocol that additionally allows the verifier to check that certain elements of the preimage $\boldsymbol{u}$ lie in a range $[a, b]$.

Definition 5 (informal)**.** *We define the ternary relation $\mathscr{R}_{\mathsf{PoKEDEx}}$ to consist of tuples of the form* $(\mathbf{G}, (U, m, t, (a_i, b_i)_{i=1}^t, f), \boldsymbol{u})$ *such that*

(a) $U = \langle \boldsymbol{u}, \mathbf{G} \rangle$,
(b) $f(\boldsymbol{u}) = 0$, *and*
(c) $\forall i \in [m - t + 1, m] : u_i \in [a_i, b_i]$.

Items (a) and (b) are guaranteed by a $\mathsf{PoKEMath}$. Item (c) can be achieved by further extending the $\mathsf{PoKEMath}$ protocol to contain a *range proof* for u_i being in the range $[a_i, b_i]$, for each $i \in [m - t + 1, m]$.

Prior range proofs. A technique used in prior works [AGLMS23; Gro05; Lip03], constructs a range proof that shows that $v \in [a, b]$ by arguing that both $b - v$ and $v - a$ are positive, via the *Lagrange four square theorem* and a proof that both $b - v$ and $v - a$ can be decomposed into a sum of four squares. While this approach, due to a constant-size decomposition, yields constant-size range proof, the prover's asymptotic complexity is proportional to the complexity of finding the four-square decomposition, which is $\Omega(n^3)$ [PT18].

Our approach. We improve on this approach by

(i) **Improving the square decomposition prover complexity**: We propose a novel algorithm that decomposes a non-negative n-bit integer into a sum of logarithmically many squares, in contrast to the constant number of squares (e.g., 4) used in prior work. Our approach reduces the $\Omega(n^3)$ prover complexity to $O(\mathsf{M}(n) \log^2 n)$ bit operations, where $\mathsf{M}(n)$ is the complexity of n-bit multiplication, which is $O(n \log n)$ [HH21]. The full algorithm is detailed in Section 6.

(ii) **Batching the positivity arguments**: Observe that $v \in [a, b]$ if and only if $(b - v)(v - a) \geq 0$. Therefore, Unlike [AGLMS23] that separately proves positivity of $(b - v)$ and $(v - a)$, we only prove positivity for $(b - v)(v - a)$. Now the prover invokes only *one* square decomposition, obtaining the vector $\boldsymbol{u} = (u_1, \ldots, u_{\log n})$, where $(b - v)(v - a)$ is an n-bit integer, such that:

$$(b - v)(v - a) = u_1^2 + \cdots + u_m^2 = \langle \boldsymbol{u}, \boldsymbol{u} \rangle \tag{4}$$

At a high level, our range proof for $v \in [a, b]$ works as follows: the prover computes $\boldsymbol{u} \in \mathbb{Z}^{\log n}$ as in Eq. (4) and sends $V := v \cdot G_1 + \sum_{i=1}^{\log n} u_i \cdot G_{i+1}$ to the verifier. The prover and verifier then engage in a $\mathsf{PoKEMath}$ protocol to check that $V = \langle (v, \boldsymbol{u}), \mathbf{G} \rangle$ and $f(v, \boldsymbol{u}) = 0$, where

$$f(v, \boldsymbol{u}) := \langle \boldsymbol{u}, \boldsymbol{u} \rangle - (v - a)(b - v) \tag{5}$$

Naively, given a verifier's challenge ℓ, $\mathsf{PoKEMath}$ would require that the prover send a vector $\mathbf{r} = (v, \boldsymbol{u}) \mod \ell$ to the verifier, who then checks that $f(\mathbf{r}) = 0 \mod \ell$. This would involve $O(\log n)$ communication.

However, we notice that the check $f(\mathbf{r}) = 0 \mod \ell$ can be expressed as

$$(v - a)(b - v) = \langle \boldsymbol{u}, \boldsymbol{u} \rangle \mod \ell$$

This is precisely the kind of statement that SIPA can handle. Recall that SIPA has a communication complexity that is logarithmic in the length of the input instance, which in this case is $O(\log n)$. Thus, our construction of $\mathsf{PoKEDEx}$ uses SIPA to check the above equation, which allows us to reduce the communication complexity of the range proof to $O(\log \log n)$. The full construction is presented in Section 7.3.

5 DewTwo

In this section, we provide a detailed presentation of DewTwo, a transparent multilinear polynomial commitment scheme (PCS) designed for the groups of unknown order setting. We improve upon prior work, Dew [AGLMS23] and Behemoth [SB23] in the prover time, proof size, parameter specification and underlying assumptions (see Section 1.3 for a full comparison).

First, in Section 5.1, we recall a commitment scheme for multilinear polynomials implicit in prior work. Then, in Section 5.2, we equip this commitment scheme with an evaluation protocol, presenting DewTwo as a full-fledged polynomial commitment scheme.

5.1 Polynomial commitment from integer encoding

Prior work, DARK [BFS19] and Dew [AGLMS23], shows how to commit to a univariate polynomial using an integer encoding of the polynomial. We modify this slightly to the multilinear setting, and present the construction in a self-contained manner in this section.

Definition 5.1 (Polynomial integer encoding)**.** *Let $q \in \mathbb{N}$ be a prime integer, $\mu \in \mathbb{N}$ the number of variables, and $p(\mathbf{X}) \in \mathbb{F}_q[X_1, \dots, X_\mu]$ a multilinear polynomial of size $N = 2^\mu$ (see Section 2.2). Denote by $\mathbf{p} \in \mathbb{F}_q^N$ the coefficient vector corresponding to p, and let $\tilde{\mathbf{p}} \in \mathbb{Z}^N$ represent this vector cast to integers, where $\tilde{\mathsf{p}}_i \in [0, q-1]$ for all $i \in [N]$. The integer encoding of p at $\alpha \in \mathbb{Z}$ is defined as a map $f : \mathbb{F}_q[X_1, \dots, X_\mu] \to \mathbb{Z}$ and is denoted by:*

$$f(p(\mathbf{X})) = \widetilde{p}(\alpha) = \sum_{i=0}^{N} \tilde{\mathsf{p}}_i \alpha^i$$

Computing the integer encoding of a polynomial is equivalent to evaluating a univariate polynomial at an integer α which can be done via the Horner's rule [Knu98, Section 4.6.4] in $O(N \cdot \mathsf{M}(\log \alpha))$ bit operations.

Polynomial integer decoding. We define $\mathsf{repr}_\alpha(x) \to (y_0, y_1, \dots, y_n)$ as the algorithm that decodes an integer x into its coefficient vector $(y_0, y_1, \dots, y_n)$ in base α. In other words, the algorithm computes the *representation* of x in base α.

```
$\mathsf{repr}_\alpha(x) \to (y_0, y_1, \dots, y_n)$

1. Initialize $i = 0$.
2. While $x > 0$:
    (a) Compute $y_i = x \mod \alpha$.
    (b) Update $x = \lfloor x/\alpha \rfloor$.
    (c) Increment $i = i + 1$.
3. Return the coefficients $y_0, y_1, \dots, y_n$.
```

To decode an integer into a polynomial over a finite field, we first apply the polynomial integer decoding algorithm $\mathsf{repr}_\alpha(\cdot)$, which retrieves the coefficient vector of the polynomial in base α. Next, we cast this coefficient vector to field elements to reconstruct the polynomial.

However, It is important to note that, the encoding-decoding process succeeds only if α is significantly larger (as large as $q^{7\mu}$) than the coefficients of the polynomial. Consequently, $|\widetilde{p}(\alpha)| = \Omega(N)$. This implies that the integer encoding cannot serve directly as a succinct commitment to the polynomial. Instead, following the approach of DARK [BFS19] and Dew [AGLMS23], we employ the exponentiation of the integer encoding to commit to the polynomial. The construction is as follows:

Construction 1 (Commitment scheme)**.** *Let $q \in \mathbb{N}$ be a prime integer, $\mu \in \mathbb{N}$ the number of variables, and $N = 2^\mu$. The commitment scheme $\mathsf{CS} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$ for the message space $\mathcal{M} = \mathbb{F}_q[X_1, \ldots, X_\mu]$, characterized by the tuple (q, μ), is constructed as follows:*

- $\mathsf{CS.Setup}(1^\lambda, (q, \mu)) \rightarrow \mathsf{pp}$*: On input of the unary security parameter 1^λ, the message space characterized by the prime order of the field $q \in \mathbb{N}$, satisfying $\lambda < \log q$, and the number of variables $\mu \in \mathbb{N}$, satisfying $4 \leq \mu \leq \lambda/4$, it samples a group $\mathbb{G} \leftarrow \mathsf{GGen}(1^\lambda)$ and a generator $G \leftarrow \mathbb{G}$. It then outputs $\mathsf{pp} = (G, \alpha)$, where $\alpha := q^{7\mu}$.*
- $\mathsf{CS.Commit}(\mathsf{pp}, p) \rightarrow (\mathsf{cm}, \tilde{\mathbf{p}})$*: To commit to a multilinear polynomial $p \in \mathbb{F}_q[X_1, \ldots, X_\mu]$ using the public parameter pp, compute $\mathsf{cm} = \widetilde{p}(\alpha) \cdot G$, where $\widetilde{p}(\alpha)$ is the integer encoding of p, and obtain $\tilde{\mathbf{p}}$ as described in Definition 5.1.*
- $\mathsf{CS.Open}(\mathsf{pp}, \mathsf{cm}, p, \tilde{\mathbf{p}}) \rightarrow 0, 1$*: To verify an opening, given the public parameters pp, the commitment cm, the polynomial p (the message), and the integer evaluation vector $\tilde{\mathbf{p}}$ (the hint vector), perform the following checks:*
  - $\mathsf{cm} = \sum_{i=0}^{N-1} \tilde{\mathsf{p}}_i \alpha^i$.
  - $p \in \mathbb{F}_q[X_1, \ldots, X_\mu]$ *with a coefficient vector* $\mathbf{p}$ *such that* $\mathbf{p} = \tilde{\mathbf{p}} \mod q$.
  - $\tilde{\mathbf{p}} \in \mathcal{H}^N$ *where* $\mathcal{H} = \left\{\frac{a}{b} \in \mathbb{Q} : 0 \leq |\frac{a}{b}| \leq (Nq^{2\mu+1} + 1), 0 < b \leq N^{8\mu} \cdot (2\mu)^\lambda\right\}$

  and output 1 if all checks pass.

Theorem 5.2. *The commitment scheme $\mathsf{CS} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$ described in Construction 1 is binding under the modular consistency assumption (see Definition 3.4).*

Proof. The proof is deferred to Appendix B.1. □

Remark 5.3. In Construction 1, the hint space $\mathcal{H}$ is defined to ensure that coefficients of the polynomial are bounded rationals – this is required to ensure binding of the commitment scheme (see Corollary 5 in [BF23]).

5.2 The DewTwo protocol

We extend the commitment scheme introduced in Construction 1 to a polynomial commitment scheme by adding an evaluation protocol to obtain DewTwo. First, we build a polynomial commitment scheme with weak extractability (see Definition 2.4), meaning that its only extractable for random evaluation points. Then we extend it to a polynomial commitment scheme with strong extractability (see Definition 2.2), meaning that it is extractable for arbitrary evaluation points.

5.2.1 Weak-DewTwo: A PCS with weak extractability

Weak-DewTwo inherits the same setup, commitment and opening procedures as Construction 1. Here, we present the evaluation protocol for Weak-DewTwo, denoted by Weak-Eval which has weak extractability (see

Definition 2.4). We emphasize that in the prominent application of polynomial commitment schemes, namely compiling a PIOP to a SNARK, the evaluation point is indeed chosen randomly and hence Weak-DewTwo suffices.

Before presenting the Weak-DewTwo protocol, we define some helper algorithms:

Helper Method: CoeffSplit. Similar to the function of the same name in Dew [AGLMS23], this method intuitively splits the product ab into three parts (based on its base-α representation): a left part s, containing all terms with degree less than i; a middle term t, containing the i-th degree term; and a right part u, containing all terms with degree greater than i.

```
CoeffSplit$(i, a, b, \alpha) \to (s, t, u)$

Parse: $i \in \mathbb{N}, a, b, s, t, u, \alpha \in \mathbb{Z}$
1. Compute $c = ab$ and let $\boldsymbol{c} = (c_0, \ldots, c_m)$ be the vector representing $c$ in base $\alpha$ where $m := \lceil \log_\alpha c \rceil$.
2. Output $(s, t, u)$ where
```

$$
\begin{aligned}
c &= \sum_{j=0}^{m} c_j \alpha^j \\
&= \sum_{j=0}^{i-1} c_j \alpha^j + c_i \alpha^i + \sum_{j=i+1}^{m} \vec{c}_j \alpha^j \\
&= s + t\alpha^i + u\alpha^{i+1}
\end{aligned}
$$

Helper method: MonomialExpand. The MonomialExpand algorithm expands a μ-variate evaluation point $\boldsymbol{y} \in \mathbb{F}_q^\mu$ into a N-dimensional vector $\boldsymbol{r} \in \mathbb{F}_q^N$ representing all possible monomials of a multilinear polynomial evaluated at this point.

```
MonomialExpand$(\boldsymbol{y}) \to \boldsymbol{r}$

1. For all $i \in \{0, \ldots, N-1\}$: compute $r_i = \prod_{j=0}^{\mu-1} y_j^{i_j}$ where $i_j$ is the $j$-th bit of the binary representation of $i$ and the product is over integers.
2. Output the monomial expansion $\boldsymbol{r} = [r_i]_{i=0}^{N}$.
```

```
Weak-Eval

$\langle \mathcal{P}(\mathsf{pp}, \mathbb{x}, \mathbb{w}), \mathcal{V}(\mathsf{pp}, \mathbb{x}) \rangle$:

Parse: $\mathbb{x} = (q \in \mathbb{N}, \mu \in \mathbb{N}, \mathsf{cm} \in \mathbb{G}, \mathbf{y} \in \mathbb{F}_q^\mu, z \in \mathbb{F}_q)$, $\mathbb{w} = (p \in \mathbb{F}_q[X_1, \ldots, X_\mu], \tilde{\mathbf{p}} \in \mathbb{Z}^N)$, $\mathsf{pp} := (3 \log \lambda N, \mathbb{G}, \boldsymbol{G})$.

1. $\mathcal{P}$ and $\mathcal{V}$ compute $\mathbf{r} :=$ MonomialExpand$(\mathbf{y})$ and $\mathbf{r}_f := \overline{\mathbf{r}}$.
2. $\mathcal{P}$ and $\mathcal{V}$ compute the integer encoding $\sigma := \widetilde{\mathbf{r}}_f(\alpha)$.
3. $\mathcal{P}$ splits up the integer $\sigma \cdot \widetilde{p}(\alpha)$ by invoking

       $(s, t, u) = $ CoeffSplit$(N-1, \widetilde{p}(\alpha), \sigma, \alpha)$
```

4. $\mathcal{P}$ casts the evaluation $z \in \mathbb{F}_q$ to $\tilde{z} \in \mathbb{Z}$ and computes the quotiont $w \in \mathbb{Z}$ such that $t = \tilde{z} + wq$.
5. $\mathcal{P}$ computes a batch commitment to the integers w, s, t, and u

$$U = w \cdot G_2 + s \cdot G_3 + t \cdot G_4 + u \cdot G_5$$

   and sends U to $\mathcal{V}$.
6. $\mathcal{P}$ and $\mathcal{V}$ invoke the $\mathsf{TPoKEDex}$ (See Argument 6) on the following instance-witness pair

$$\begin{aligned}\mathbb{x}_{\mathsf{TPoKEDex}} &= (\mathsf{cm}, U, (0, Nq^{\mu+1}\alpha^{N-2}), (0, Nq^{\mu+1}), (0, q^{\mu+1}\alpha^{N-2}), g(\cdot)) \\ \mathbb{w}_{\mathsf{TPoKEDex}} &= (\widetilde{p}(\alpha), (w, s, t, u))\end{aligned}$$

   where $g : \mathbb{Z}^5 \to \mathbb{Z}$ is a function that on input $(\widetilde{p}(\alpha), w, s, t, u)$, outputs 0 iff
   (a) $t = z + wq$, *and*
   (b) $\sigma \cdot \widetilde{p}(\alpha) = s + t\alpha^{N-1} + u\alpha^N$.

Remark 5.4 (Verifier efficiency)**.** The verifier does not need to compute σ. Instead, it can just compute the value of $\sigma \mod \ell$ required in $\mathsf{TPoKEDex}$ where $\ell \in \mathsf{Primes}[\lambda, 2\lambda]$ is a random challenge. however, a naive implementation of performing a simple integer encoding after the monomial expansion would, even with the $\mod \ell$, result in a $\Omega(N)$ many $O(\lambda)$ bit-operations, making the verifier non-succinct. To address this, $\mathcal{V}$ leverages the fact that the polynomiak $\widetilde{\mathbf{r}_f}(x)$ has the succinct representation

$$\widetilde{\mathbf{r}_f}(X) = \prod_{i=1}^{\mu} \Big(1 + \mathsf{r}_{\mu-i}\, X^{\,2^{\,i-1}} \Big),$$

and computes $\widetilde{\mathbf{r}_f}(\alpha)$ by computing $\prod_{i=1}^{\mu} \Big(1 + \mathsf{r}_{\mu-i}\, \alpha^{\,2^{\,i-1}} \Big)$. The powers of α can be efficiently computed using repeated squaring, requiring $O(\mu)$ number of $O(\lambda)$-bit multiplications. Additionally, multiplying the powers with r_i and aggregating the results also requires $O(\mu)$ many $O(\lambda)$-bit multiplications. As a result, the total verifier time complexity is just $O(\mu\mathsf{M}(\lambda))$.

Theorem 5.5. *Let* $\mathsf{CS} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$ *be the commitment scheme defined in Construction 1. Then* WEAK-DEWTWO $= (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open},$ WEAK-EVAL$)$ *is a polynomial commitment scheme with weak extractability under the modular consistency (see Definition 3.4), strong RSA (see Definition 3.6), and discrete logarithm (see Definition 3.1) assumptions.*

Furthermore, let $\mu \in \mathbb{N}$ be the number of variables, $N = 2^\mu \in \mathbb{N}$ be the polynomial size, and $\lambda \in \mathbb{N}$ be the security parameter; WEAK-DEWTWO *has the following properties:*

1. *Prover complexity is $O(\lambda N \log^2(\lambda N))$ group operations, and $O(\lambda N \log^4(\lambda N))$ bit operations.*
2. *Verifier complexity is $O(\lambda + \log \lambda N)$ group operations, and $O(\lambda \log \lambda \log N)$ bit operations.*
3. *The proof size is $O(\log\log N\lambda)$ group elements and $O(\lambda \log\log N\lambda)$ bits of integers (Concretely 4.5 KB for $\mu = 30$ and $\lambda = 128$).*
4. *The public parameter size is $O(\log N\lambda)$ group elements.*

Proof. The completeness is straightforward to check. The proof for binding follows from the binding property of the commitment scheme in Section 2.4 which is deferred to Appendix B.2. The discussion of the costs and the proof for weak extractability are also deferred to Appendix B.2.

5.2.2 DewTwo protocol with strong extractability

We introduce DewTwo, a PCS with strong extractability, which ensures security even when the evaluation point is chosen arbitrarily or adversarially. DewTwo invokes Weak-DewTwo twice: once with a random evaluation point and once with the claimed evaluation point. Intuitively, the first invocation (with the random point) ensures that the commitment is well-formed, while the second invocation (with the claimed point) verifies the actual claimed evaluation. This design parallels the approach in [AGLMS23], where the protocol is similarly divided into two sub-protocols, TEST and IPP.

Eval

$\langle \mathcal{P}(\mathsf{pp}, \mathbb{x}, \mathbb{w}), \mathcal{V}(\mathsf{pp}, \mathbb{x})\rangle$:

Parse: $\mathbb{x} = (q \in \mathbb{N}, \mu \in \mathbb{N}, \mathsf{cm} \in \mathbb{G}, \mathbf{y} \in \mathbb{F}_q^{\mu}, z_{\mathbf{y}} \in \mathbb{F}_q)$, $\mathbb{w} = (p \in \mathbb{F}_q[X_1, \ldots, X_\mu], \tilde{\mathbf{p}} \in \mathbb{Z}^N)$, $\mathsf{pp} := (3 \log \lambda N, \mathbb{G}, \boldsymbol{G})$.

1. $\mathcal{V}$ generates a random evaluation point $\mathbf{r} \overset{\$}{\leftarrow} \mathbb{F}_q^{\mu}$ and sends $\mathbf{r}$ to $\mathcal{P}$.
2. $\mathcal{P}$ sends the claimed evaluation $z_{\mathbf{r}}$ to $\mathcal{V}$.
3. $\mathcal{P}$ and $\mathcal{V}$ compute two instances for Weak-DewTwo:

$$\mathbb{x}_{\mathbf{r}} = (q, \mu, \mathsf{cm}, \mathbf{r}, z_{\mathbf{r}}) \text{ and } \mathbb{x}_{\mathbf{y}} = (q, \mu, \mathsf{cm}, \mathbf{y}, z_{\mathbf{y}})$$

4. $\mathcal{V}$ outputs 1 if and only if the two invoked Weak-DewTwo verifiers accept:

$$\textsc{Weak-Eval}(\langle \mathcal{P}(\mathsf{pp}, \mathbb{x}_{\mathbf{r}}, \mathbb{w}), \mathcal{V}(\mathsf{pp}, \mathbb{x}_{\mathbf{r}})\rangle) \quad \wedge \quad \textsc{Weak-Eval}(\langle \mathcal{P}(\mathsf{pp}, \mathbb{x}_{\mathbf{y}}, \mathbb{w}), \mathcal{V}(\mathsf{pp}, \mathbb{x}_{\mathbf{y}})\rangle)$$

Theorem 5.6. *Let* $\mathsf{CS} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$ *be the commitment scheme defined in Construction 1. Then DewTwo* $= (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Eval})$ *is a PCS with strong extractability under the modular consistency (see Definition 3.4), strong RSA (see Definition 3.6), and discrete logarithm (see Definition 3.1) assumptions.*

Furthermore, let $\mu \in \mathbb{N}$ *be the number of variables,* $N = 2^\mu \in \mathbb{N}$ *be the polynomial size, and* $\lambda \in \mathbb{N}$ *be the security parameter; Weak-DewTwo has the following properties:*

1. *Prover complexity is* $O(\lambda N \log^2(\lambda N))$ *group operations, and* $O(\lambda N \log^4(\lambda N))$ *bit operations.*
2. *Verifier complexity is* $O(\lambda + \log \lambda N)$ *group operations, and* $O(\lambda \log \lambda \log N)$ *bit operations.*
3. *The proof size is* $O(\log \log N\lambda)$ *group elements and* $O(\lambda \log \log N\lambda)$ *bits of integers (Concretely* 9 *KB for* $\mu = 30$ *and* $\lambda = 128$*).*
4. *The public parameter size is* $O(\log N\lambda)$ *group elements.*

Proof. The completeness is straightforward to check. The proof for binding follows from the binding property of the commitment scheme in Appendix B.1 which is deferred in Appendix B.2. The proof for strong extractability are also deferred in Appendix B.3.

The costs of DewTwo are effectively double that of Weak-DewTwo since it invokes Weak-DewTwo twice. Hence, the discussion for costs of DewTwo follows from the discussion for Weak-DewTwo (see Appendix B.2). □

6 Fast integer square decomposition

The problem of integer square decomposition concerns how to represent a non-negative n-bit integer x as a sum of k squares, i.e.,

$$x = y_1^2 + y_2^2 + \ldots + y_k^2.$$

Finding these representations is used in proving that a committed integer x is non-negative (since it can be expressed as the sum of k squares). Furthermore, one can show that a committed integer x lies within a given interval $[a, b]$ by proving that both $x - a$ and $b - x$ are non-negative, i.e., the product $(b - x)(x - a)$ can be written as a sum of k squares.

Lagrange's theorem shows that such a representation always exists for $k = 4$, meaning that every non-negative integer can be expressed as a sum of four squares [DSV03]. Moreover, for constructing range proofs, Groth introduced a technique [Gro05] that reduces the size of the representation to $k = 3$. While such a representation always exists for some $k \in \mathbb{N}$, finding one is non-trivial. To the best of our knowledge, all previous works on range proofs that rely on integer square decomposition to establish positivity, such as [AGLMS23; Gro05; LAN01], have used variants of the Rabin-Shallit algorithm [RS86] to find a *constant-size* decomposition. The state-of-the-art Pollack-Trevino algorithm [PT18] finds a constant-size decomposition in an expected $O(n^2/\log n)$ arithmetic operations, or equivalently $\Omega(n^3)$ bit operations.

In this work, we propose a new algorithm $\mathsf{IntegerSquareDecompose}$ that decomposes an n-bit integer into a sum of at most $\log n$ squares, instead of constant number of squares, and runs in quasi-linear time. Specifically, it has a complexity of $O(\mathsf{M}(n) \log^2 n) = \tilde{O}(n)$ bit operations.

6.1 Our algorithm

Our new $\mathsf{IntegerSquareDecompose}$ algorithm is a greedy algorithm that iteratively identifies the largest square that can be subtracted from the input integer, removes it, and continues this process until the integer is reduced to zero.

Integer square roots. The most computationally expensive part of this algorithm is the repeated integer square root calculation. The integer square root of a non-negative integer x is the integer $y = \lfloor \sqrt{x} \rfloor$. As a subprotocol, we use the function

$$\mathsf{SquareRoot}(a, \delta) \rightarrow b$$

that takes in an integer a and an error tolerance δ and outputs a rational b such that $|b - \sqrt{a}| \leq \delta$. The error tolerance δ is set to a value $0.1 < 1$ in our case. $\mathsf{SquareRoot}$ uses the Newton-Raphson method [AH85, Chapter 3], a classical root-finding algorithm, and we present the full construction in Appendix C. We show in Corollary C.4 that $\mathsf{SquareRoot}$ requires $O(\mathsf{M}(n) \log n)$ bit operations for an n-bit integer a and constant δ.

Given $\mathsf{SquareRoot}$ as a black-box. we show how to compute the integer square root of an integer x below.

```
IntegerSquareRoot($x$) → $y$

1. Compute $y \leftarrow \lfloor \mathsf{SquareRoot}(x, 0.1) \rfloor$.
2. If $(y+1)^2 \leq x$: update $y \leftarrow y + 1$.
```

```
3. If $y^2 > x$: update $y \leftarrow y - 1$.
```

Lemma 6.1. *The algorithm* IntegerSquareRoot(x) *outputs* $y = \lfloor\sqrt{x}\rfloor$ *using* $O(\mathsf{M}(n)\log n)$ *bit operations.*

Proof. Corollary C.4 gives that SquareRoot$(x, 0.1)$ requires $O(\mathsf{M}(n)\log n)$ bit operations and outputs b such that $b \in [\sqrt{x} - 0.1, \sqrt{x} + 0.1]$. Thus $\lfloor b\rfloor \in \{\lfloor\sqrt{x}\rfloor - 1, \lfloor\sqrt{x}\rfloor, \lfloor\sqrt{x}\rfloor + 1\}$. This implies that Steps 2 and 3 ensure that the final output $y = \lfloor\sqrt{x}\rfloor$, and the additional number of bit operations required is $O(\mathsf{M}(n))$. □

Integer square decomposition. IntegerSquareDecompose iteratively uses the IntegerSquareRoot algorithm to decompose an n-bit integer x into the sum of $\log n$ squares, as detailed below.

```
IntegerSquareDecompose$(x) \to \boldsymbol{y}$

1. Set $x_{rem} := x$ and $i := 0$.
2. While $x_{rem} > 0$:
3.     Define $y_i :=$ IntegerSquareRoot$(x_{rem})$.
4.     Update $x_{rem} \leftarrow x_{rem} - y_i^2$.
5. Output $\boldsymbol{y}$.
```

Lemma 6.2. *Let* $x \in \mathbb{N}$ *be an* n*-bit non-negative integer. The algorithm* IntegerSquareDecompose(x) *outputs* $\boldsymbol{y} \in \mathbb{Z}^{\log n+5}$ *such that* $x = \sum_{i=1}^{\log n+5} y_i^2$ *and requires* $O(\mathsf{M}(n)\log^2 n)$ *bit operations.*

Proof. At each step, x_{rem} decreases by $\lfloor\sqrt{x_{rem}}\rfloor^2$, and since $\lfloor\sqrt{x_{rem}}\rfloor^2 \geq 1$ for $x_{rem} \geq 1$, x_{rem} strictly decreases, and in at most $k = x$ iterations outputs the vector $\boldsymbol{y} = (y_1, \ldots, y_k)$ that satisfies $x = y_1^2 + y_2^2 + \cdots + y_k^2$ by construction.

For runtime, let x_i be the value of x_{rem} in the i'th iteration. In each iteration x_i decreases by at least $(\sqrt{x_i} - 1)^2$, leading to $x_{i+1} \leq 2\sqrt{x_i} - 1 \implies x_{i+1} < 2\sqrt{x_i}$. Taking the logarithm of both sides:

$$\log x_{i+1} < \log 2 + \frac{1}{2}\log x_i \implies \lceil\log x_{i+1}\rceil \leq \lceil 1 + \frac{1}{2}\log x_i\rceil \implies \lceil\log x_{i+1}\rceil \leq 2 + \frac{1}{2}\lceil\log x_i\rceil$$

Define $n_i := \lceil\log x_i\rceil$ be the bit length of x_i, then this implies that $n_{i+1} \leq 2 + \frac{1}{2}n_i$. Let n be the bit length of x, this implies that $n_2 \leq 2 + \frac{n}{2}$, $n_3 \leq 2 + \frac{2}{2} + \frac{n}{4} = 3 + \frac{n}{4}$, $n_4 \leq 2 + \frac{3}{2} + \frac{n}{8}$, and so on. More formally, it can be shown via induction that $n_k \leq 4 + \frac{n}{2^{k-1}}$. Thus $n_k = 4$ when $k = \log n + 2$, which means after $\log n + 2$ iterations, x_{rem} has bit length 4.

This implies that $x_{rem} \leq 15$, which implies that the number of additional iterations is at most 3 (as there are only 3 squares smaller than 15). Thus the total number of iterations required is at most $\log n + 5$, and each iterations costs $O(\mathsf{M}(n)\log n)$ bit operations as the most expensive part of each iteration is computing IntegerSquareRoot. This implies that the entire algorithm requires $O(\mathsf{M}(n)\log^2 n)$ bit operations. □

7 Toolbox: arguments of knowledge about exponents

In this section, we describe a sequence of arguments of knowledge which serve as a toolbox for constructing our polynomial commitment scheme, but could also be of independent interest.

First, we introduce PoKE (Section 7.1), which was originally proposed in [BBF19]. However, we provide an alternative security proof based on a new falsifiable assumption (see Definition 3.4), rather than relying on the Generic Group Model (GGM). Then, we present our novel extensions of PoKE: $\mathsf{PoKEMath}$ (Section 7.2) and $\mathsf{PoKEDEx}$ (Section 7.3). Finally, we introduce $\mathsf{TPoKEDex}$ (Section 7.4), an optimized version of $\mathsf{PoKEDEx}$ that is specifically tailored for constructing DewTwo.

7.1 Proof of Knowledge of Exponent (PoKE)

PoKE (originally called $\mathsf{PoKERep}$ in [BBF19]) is a proof of knowledge of a representation in terms of public bases $G_1, \ldots, G_m$. Specifically, given a Pedersen commitment U, the prover demonstrates knowledge of an integer vector $\boldsymbol{u} \in \mathbb{Z}^m$ such that $U = u_1 G_1 + \cdots + u_m G_m$. The goal of the protocol is to verify the knowledge of potentially large integers $\boldsymbol{u}$ more succinctly than by transmitting $\boldsymbol{u}$ itself. We formally define the PoKE relation below.

Definition 7.1 (PoKE relation[BBF19])**.** *Given* $\mathsf{pp} = (N, \mathbb{G}, \mathbf{G}) \leftarrow \mathsf{Setup}(1^\lambda, N)$*, we define the proof of knowledge of exponent relation* $\mathscr{R}^{\mathsf{pp}}_{\mathsf{PoKE}}$ *as follows:*

$$\begin{pmatrix} \mathbb{x}, \\ \mathbb{w} \end{pmatrix} = \begin{pmatrix} (U, m), \\ \boldsymbol{u} \end{pmatrix}$$

where $U \in \mathbb{G}$ *is the commitment,* $m \in \mathbb{N}$ *is the size of the witness and* $\boldsymbol{u} \in \mathbb{Z}^m$ *is the witness integer vector such that*

$$m \leq N \qquad \textit{and} \qquad U = \langle \boldsymbol{u}, \boldsymbol{G}|_{[1:m]} \rangle.$$

PoKE [BBF19]

$\langle \mathcal{P}(\mathsf{pp}, \mathbb{x}, \mathbb{w}), \mathcal{V}(\mathsf{pp}, \mathbb{x}) \rangle$:

Parse. : $\mathsf{pp} := (N, \mathbb{G}, \mathbf{G}),\ \mathbb{x}_{\mathsf{PoKE}} := (U, m),\ \mathbb{w}_{\mathsf{PoKE}} := \boldsymbol{u}$

1. If $m > N$, $\mathcal{V}$ rejects. Otherwise, continues.
2. $\mathcal{V}$ samples a random challenge $\mathsf{Primes}[\lambda, 2\lambda]$ and sends it to $\mathcal{P}$.
3. $\mathcal{P}$ computes the vectors $\boldsymbol{r} := [r_i]_{i=1}^m$ and $\boldsymbol{q} := [q_i]_{i=1}^m$ such that it holds that $u_i = l \cdot q_i + r_i$.
4. $\mathcal{P}$ computes $Q := \langle \boldsymbol{q}, \boldsymbol{G}|_{[1:m]} \rangle$ and sends $(Q, \boldsymbol{r})$ to $\mathcal{V}$.
5. $\mathcal{V}$ computes $R := \langle \boldsymbol{r}, \boldsymbol{G}|_{[1:m]} \rangle$ and checks that $U \overset{?}{=} \ell \cdot Q + R$.

Theorem 7.2. PoKE *is a public-coin argument of knowledge (see Section 2.3) for the indexed relation* $\mathscr{R}^{\mathsf{pp}}_{\mathsf{PoKE}}$ *in the standard model under the Modular consistency assumption (see Definition 3.4). Moreover, let each witness component* u_i *have bit-length of at most* $n \gg \lambda$, PoKE *has the following properties:*

- *Prover time is* $\mathsf{MSM}(m, n)$ *group operations and* $O(m\mathsf{M}(n))$ *bit operations.*
- *Verifier time is* $\mathsf{MSM}(m, 2\lambda)$ *group operations.*

- *Proof size is* 1 *group element and* $2m\lambda$ *bits.*

Proof. Completeness and complexity of the argument are straightforward to check. We defer the proof of knowledge-soundness to Appendix D.1.

7.2 Proof of Knowledge of Exponent Math (PoKEMath)

$\mathsf{PoKEMath}$ is an extension of PoKE where, given a Pedersen commitment U, the prover not only demonstrates knowledge of the preimage integer vector $\boldsymbol{u}$, but also proves that $\boldsymbol{u}$ is a root of a specified function f, i.e., $f(\boldsymbol{u}) = 0$. Essentially, this allows the prover to verify any polynomial-time computable property (represented by the function f) of the witness vector $\boldsymbol{u}$ much more succinctly than by explicitly computing $f(\boldsymbol{u})$.

Definition 7.3. *Given* $\mathsf{pp} = (N, \mathbb{G}, \mathbf{G}) \leftarrow \mathsf{Setup}(1^\lambda, N)$*, we define* $\mathscr{R}^{\mathsf{pp}}_{\mathsf{PoKEMath}}$ *as the set of tuples:*

$$\begin{pmatrix} \mathbb{x}, \\ \mathbb{w} \end{pmatrix} = \begin{pmatrix} (U, m, f), \\ \boldsymbol{u} \end{pmatrix}$$

where $U \in \mathbb{G}$ *is a commitment,* $m \in \mathbb{N}$ *is the size of the witness vector,* $f : \mathbb{Z}^m \to \mathbb{Z}$ *is a function and* $\boldsymbol{u} \in \mathbb{Z}^m$ *is the witness integer vector such that*

$$m \leq N, \quad U = \langle \boldsymbol{u}, \boldsymbol{G}|_{[1:m]} \rangle, \quad f(\boldsymbol{u}) = 0$$

$\mathsf{PoKEMath}$

$\langle \mathcal{P}(\mathsf{pp}, \mathbb{x}, \mathbb{w}), \mathcal{V}(\mathsf{pp}, \mathbb{x}) \rangle$:

Parse. $\mathsf{pp} := (N, \mathbb{G}, \mathbf{G}),\ \mathbb{x}_{\mathsf{PoKEMath}} := (U, m, f),\ \mathbb{w}_{\mathsf{PoKEMath}} := \boldsymbol{u}$

1. If $m > N$, $\mathcal{V}$ rejects. Otherwise, continues.
2. $\mathcal{V}$ samples a random challenge $\ell \xleftarrow{\$} \mathsf{Primes}[\lambda, 2\lambda]$ and sends it to $\mathcal{P}$.
3. $\mathcal{P}$ computes the vectors $\boldsymbol{r} := [r_i]_{i=1}^m$ and $\boldsymbol{q} := [q_i]_{i=1}^m$ such that it holds that
$$u_i = \ell \cdot q_i + r_i$$
4. $\mathcal{P}$ computes $Q := \langle \boldsymbol{q}, \boldsymbol{G} \rangle$ and sends $(Q, \boldsymbol{r})$ to $\mathcal{V}$.
5. $\mathcal{V}$ computes $R := \langle \boldsymbol{r}, \boldsymbol{G} \rangle$ and checks that
$$U \overset{?}{=} \ell \cdot Q + R$$
$$f(\boldsymbol{r}) \overset{?}{=} 0 \mod \ell$$

Theorem 7.4. $\mathsf{PoKEMath}$ *is a public-coin argument of knowledge (see Section 2.3) for the ternary relation* $\mathscr{R}^{\mathsf{pp}}_{\mathsf{PoKEMath}}$ *under the modular consistency (see Definition 3.4) and strong RSA (see Definition 3.6) assumptions.*

Moreover, let the function f *be a function computable by a size* s *arithmetic circuit and let each witness component* u_i *have a bit length of at most* n*, then* PoKE *has the following properties:*

- *Prover time is* $\mathsf{MSM}(m, n)$ *group operations and* $O(m\mathsf{M}(n))$ *bit operations.*
- *Verifier time is* $\mathsf{MSM}(m, \lambda)$ *group operations and* $O(s\mathsf{M}(\lambda))$ *bit operations.*

- *Proof size is* 1 *group element plus* $2m\lambda$ *bits.*

Proof. Completeness and complexity of the argument are straightforward to check. We defer the proof of knowledge-soundness to Appendix D.2.

7.3 Proof of Knowledge of Exponent, Decomposition, and Expression (PoKEDEx)

PoKEDEx, is an extension of PoKEMath (see Section 7.2) where some exponents must also be range-constrained. Specifically, given a Pedersen commitment U, the prover demonstrates knowledge of an integer vector $\boldsymbol{u} \in \mathbb{Z}^m$ such that $U = \langle \boldsymbol{u}, \boldsymbol{G}|_{[1:m]} \rangle$, and the last t components of $\boldsymbol{u}$ are constrained to lie within a given range. We formally define the PoKEDEx relation below.

Definition 7.5. *Given* $\mathsf{pp} = (N, \mathbb{G}, \mathbf{G}) \leftarrow \mathsf{Setup}(1^\lambda, N)$*, we define* $\mathscr{R}^{\mathsf{pp}}_{\mathsf{PoKEDEx}}$ *as the set of tuples:*

$$\begin{pmatrix} \mathbb{x}, \\ \mathbb{w} \end{pmatrix} = \begin{pmatrix} (U, m, t, (a_i, b_i)_{i=1}^t, f) \\ \boldsymbol{u} \end{pmatrix}$$

where $U \in \mathbb{G}$ *is the commitment,* $m \in \mathbb{N}$ *is the size of the witness vector,* $t \in \mathbb{N}$ *is the number of range-constrained witness components,* $f : \mathbb{Z}^m \to \mathbb{Z}$ *is a function and* $\boldsymbol{u} \in \mathbb{Z}^m$ *is the witness integer vector. Moreover, let* $(a_i, b_i)_{i=1}^t$ *and each witness component* u_i *be* n*-bit integers. We say* $(\mathbb{x}, \mathbb{w}) \in \mathscr{R}^{\mathsf{pp}}_{\mathsf{PoKEDEx}}$ *if and only if:*

$$m + t\lceil 2n \rceil \le N, \quad U = \langle \boldsymbol{u}, \boldsymbol{G}|_{[1:m]} \rangle, \quad f(\boldsymbol{u}) = 0,$$
$$\forall i \in [t] : u_{m-t+i} \in [a_i, b_i]$$

The construction of PoKEDEx is as follows. For an intuitive explanation of the construction, refer to Section 4.4 of the techniques.

PoKEDEx

$\langle \mathcal{P}(\mathsf{pp}, \mathbb{x}, \mathbb{w}), \mathcal{V}(\mathsf{pp}, \mathbb{x}) \rangle$:

Parse: $\mathsf{pp} = (N, \mathbb{G}, \mathbf{G})$, $\mathbb{x} = (U, m, t, (a_i, b_i)_{i=1}^t, f)$ and $\mathbb{w} = \boldsymbol{u}$.

1. If $m > N$, $\mathcal{V}$ rejects. Otherwise, continues.
2. For each $i \in [m - t + 1, m]$, i.e. ranged witness indices:
3. $\quad\mathcal{P}$ computes the integer square decomposition of the integer $(u_i - a_i)(b_i - u_i)$

$$\boldsymbol{v}_i := \mathsf{IntegerSquareDecompose}((u_i - a_i)(b_i - u_i))$$

   and sends the bit length $m_i := |\boldsymbol{v}_i|$ to $\mathcal{V}$.
4. $\mathcal{P}$ concatenates the decompositions $\boldsymbol{v} := (\boldsymbol{v}_{m-t+1} \;||\; \ldots \;||\; \boldsymbol{v}_m)$ and sends the Pedersen commitment $V := \langle \boldsymbol{v}, \boldsymbol{G}|_{[m+1:m+|\boldsymbol{v}|]} \rangle$ to $\mathcal{V}$.
5. $\mathcal{V}$ samples a random stitching challenge $\beta \xleftarrow{\$} \mathsf{Primes}[\lambda, 2\lambda]$ and sends it to $\mathcal{P}$.
6. $\mathcal{P}$ and $\mathcal{V}$ compute $U' := \beta \cdot U + V$ and $\boldsymbol{G}' := (\beta \cdot \boldsymbol{G}|_{[1:m]} \;||\; \boldsymbol{G}|_{[m+1:m+|\boldsymbol{v}|]})$.
7. $\mathcal{P}$ and $\mathcal{V}$ define $f'(\boldsymbol{u}, \boldsymbol{v})$ to be the function that outputs 0 if and only if:
   (a) $f(\boldsymbol{u}) = 0$, and

(b) $(u_i - a_i)(b_i - u_i) = \langle \boldsymbol{v}_i, \boldsymbol{v}_i \rangle$, for all $i \in [m-t+1, m]$ where $\boldsymbol{v}_i$ is the i-th sub-vector of $\boldsymbol{v}$ of length m_i.

8. $\mathcal{P}$ and $\mathcal{V}$ compute the PoKEMath public parameters as $\mathsf{pp}_{\mathsf{PoKEMath}} := (N, \mathbb{G}, \boldsymbol{G}')$.
9. $\mathcal{P}$ and $\mathcal{V}$ compute the PoKEMath instance as $\mathbb{x}_{\mathsf{PoKEMath}} := (f', U', m + |\boldsymbol{v}|)$.
10. $\mathcal{P}$ computes the PoKEMath witness as $\mathbb{w}_{\mathsf{PoKEMath}} := (\boldsymbol{u}, \boldsymbol{v})$.
11. $\mathcal{P}$ and $\mathcal{V}$ run $\mathsf{PoKEMath}(\langle \mathcal{P}(\mathsf{pp}_{\mathsf{PoKEMath}}, \mathbb{x}_{\mathsf{PoKEMath}}, \mathbb{w}_{\mathsf{PoKEMath}}), \mathcal{V}(\mathsf{pp}_{\mathsf{PoKEMath}}, \mathbb{x}_{\mathsf{PoKEMath}}) \rangle)$.

Theorem 7.6. PoKEDEx *is a public-coin argument of knowledge for the indexed relation* $\mathscr{R}^{\mathsf{pp}}_{\mathsf{PoKEDEx}}$ *under the modular consistency (Definition 3.4) and strong RSA (see Definition 3.6) assumptions.*

Moreover, let $(a_i, b_i)_{i=1}^t$ *and each witness component* u_i *be* n*-bit integers, and* f *be a function computable by a size* s *arithmetic circuit.* PoKEDEx *has the following properties:*

- *Prover time is* $O(\mathsf{MSM}(m + t\log n, n))$ *group operations and* $O(\mathsf{M}(n)(m + t\log n))$ *bit operations.*
- *Verifier time is* $O(\mathsf{MSM}(m + t\log n, \lambda))$ *group operations and* $O(\mathsf{M}(\lambda)(s + t\log n))$ *bit operations.*
- *Proof size is 2 group elements plus* $O(\lambda(m + t\log(n)) + t\log\log(n))$ *bits.*

Proof. Completeness of the argument is straightforward to check. We defer the complexity analysis and proof of knowledge-soundness to Appendix D.3.

7.4 Triple PoKEDEx (TPoKEDEx)

In this section, we introduce TPoKEDex, an optimized version of PoKEDEx tailored to meet the requirements of DewTwo. Specifically, TPoKEDex is invoked once in Weak-DewTwo (see Argument 1) and twice in DewTwo (see Argument 2). The proof length in PoKEDEx consists of two components: group elements sent and bits sent. The optimization in TPoKEDex focuses on reducing the bits sent by replacing the range proof inside PoKEMath with TIPA, a Bulletproof-style range proof. This exponentially reduces the proof size from $O(n)$ to $O(\log\log n)$.

Definition 7.7. *Given* $\mathsf{pp} = (N, \mathbb{G}, \mathbf{G}) \leftarrow \mathsf{Setup}(1^\lambda, N)$*, we define the triple knowledge of exponent ranged math relation denoted by* TPoKEDex *as the set of tuples:*

$$\begin{pmatrix} \mathbb{x}, \\ \mathbb{w} \end{pmatrix} = \begin{pmatrix} (C, U, (a_i, b_i)_{i=2}^4, f) \\ (c, \boldsymbol{u}) \end{pmatrix}$$

where $(C, U) \in \mathbb{G}^2$ *are commitments,* $f : \mathbb{Z}^5 \to \mathbb{Z}$ *is a function, and* $(c, \boldsymbol{u}) \in \mathbb{Z}^5$ *are witness components such that* $c \in \mathbb{Z}$, $\boldsymbol{u} \in \mathbb{Z}^4$*. Moreover, let* $(a_i, b_i)_{i=2}^4$ *and each witness component* u_i *or* c *be* n*-bit integers. We say* $(\mathbb{x}, \mathbb{w}) \in \mathscr{R}^{\mathsf{pp}}_{\mathsf{TPoKEDex}}$ *if and only if:*

$$C = c \cdot G_1, \quad U = u_1 \cdot G_2 + u_2 \cdot G_3 + u_3 \cdot G_4 + u_4 \cdot G_5,$$
$$u_2 \in [a_2, b_2], \quad u_3 \in [a_3, b_3], \quad u_4 \in [a_4, b_4],$$
$$5 + 3\lceil 2n \rceil \leq N, \quad f(c, u_1, u_2, u_3, u_4) = 0$$

TPoKEDex

$\langle \mathcal{P}(\mathsf{pp}, \mathbb{x}, \mathbb{w}), \mathcal{V}(\mathsf{pp}, \mathbb{x})\rangle$:

Parse: $\mathsf{pp} = (N, \mathbb{G}, \mathbf{G})$, $\mathbb{x} = (C, U, (a_i, b_i)_{i=2}^4, f)$ and $\mathbb{w} = (c, \boldsymbol{u})$.

1. If $m > N$, $\mathcal{V}$ rejects. Otherwise, continues.

Stitching procedure

2. $\mathcal{V}$ samples a random stitching challenge $\beta \xleftarrow{\$} \mathsf{Primes}[\lambda, 2\lambda]$ and sends it to $\mathcal{P}$.
3. $\mathcal{P}$ and $\mathcal{V}$ compute the stitched generator vector and the stitched commitment

$$\boldsymbol{G}' := (\beta G_1, G_2, G_3, G_4, G_5), \quad U' := \beta C + U$$

Unrolling PoKEMath

4. $\mathcal{V}$ samples a random challenge $\ell \xleftarrow{\$} \mathsf{Primes}[\lambda, 2\lambda]$ and sends it to $\mathcal{P}$.
5. $\mathcal{P}$ computes the remainder vector $\boldsymbol{r} := [r_i]_{i=1}^5$ and the quotient vector $\boldsymbol{q} := [q_i]_{i=1}^5$ such that:

$$c = \ell \cdot q_1 + r_1$$
$$u_i = \ell \cdot q_i + r_i \quad \forall i \in \{2, 3, 4, 5\}$$

6. $\mathcal{P}$ computes a commitment to the quotient vector $\boldsymbol{q}$, i.e. $Q := \langle \boldsymbol{q}, \boldsymbol{G}'\rangle$, and sends $(Q, \boldsymbol{r})$ to $\mathcal{V}$.
7. $\mathcal{V}$ computes a commitment to the remainder vector $\boldsymbol{r}$, i.e. $R := \langle \boldsymbol{r}, \boldsymbol{G}'\rangle$, and checks that

$$U' \stackrel{?}{=} \ell \cdot Q + R$$
$$f(\boldsymbol{r}) \stackrel{?}{=} 0 \mod \ell$$

Invoking TIPA

8. $\mathcal{P}$ computes the integer decomposition vectors (see Section 6) $\boldsymbol{v}_3, \boldsymbol{v}_4, \boldsymbol{v}_5$ where

$$\boldsymbol{v}_i := \mathsf{IntegerSquareDecompose}((u_i - a_i)(b_i - u_i))$$

9. $\mathcal{P}$ computes $\gamma := \lceil \log(\max\{\boldsymbol{v}_3, \boldsymbol{v}_4, \boldsymbol{v}_5\})\rceil$ to $\mathcal{V}$ and pads enough zeros separately to $\boldsymbol{v}_3$, $\boldsymbol{v}_4$, and $\boldsymbol{v}_5$ to make them of equal length 2^γ.
10. $\mathcal{P}$ defines the concatenated decomposition vector $\boldsymbol{v} := (\boldsymbol{v}_3 \ || \ \boldsymbol{v}_4 \ || \ \boldsymbol{v}_5)$ of total length $3 \times 2^\gamma$.
11. $\mathcal{P}$ computes the Pedersen commitment $V := \langle \boldsymbol{v}, \boldsymbol{G}|_{[6:5+|\boldsymbol{v}|]}\rangle$ and sends V to $\mathcal{V}$.
12. $\mathcal{P}$ and $\mathcal{V}$ computes TIPA instance and public parameters (see Definition 8.6):

$$\mathsf{pp}_{\mathsf{TIPA}} := (3 \cdot 2^\gamma, \mathbb{G}, \boldsymbol{G}|_{[6:5+|\boldsymbol{v}|]})$$
$$\mathbb{x}_{\mathsf{TIPA}} := (V, \ell, (r_3, r_4, r_5))$$

13. $\mathcal{P}$ computes TIPA witness (see Definition 8.6) as

$$\mathbb{w}_{\mathsf{TIPA}} := ((\boldsymbol{v}_3, \boldsymbol{v}_4, \boldsymbol{v}_5))$$

14. $\mathcal{P}$ and $\mathcal{V}$ invoke TIPA (see Argument 8) and $\mathcal{V}$ outputs

$$\mathsf{TIPA}(\langle \mathcal{P}(\mathsf{pp}_{\mathsf{TIPA}}, \mathbb{x}_{\mathsf{TIPA}}, \mathbb{w}_{\mathsf{TIPA}}), \mathcal{V}(\mathsf{pp}_{\mathsf{TIPA}}, \mathbb{x}_{\mathsf{TIPA}})\rangle)$$

Remark 7.8. The $\mathsf{PoKEMath}$ argument (see Argument 4) is unrolled within the $\mathsf{TPoKEDex}$ argument (see Argument 6) rather than being invoked as a black-box. This approach provides white-box access to the remainder vector $\boldsymbol{r} \in \mathbb{Z}^5$, which is subsequently used in the TIPA argument (see Argument 8).

Theorem 7.9. $\mathsf{TPoKEDex}$ *(Argument 6) is a public-coin argument of knowledge (see Section 2.3) for the indexed relation $\mathscr{R}^{\mathsf{pp}}_{\mathsf{TPoKEDex}}$ in the standard model under the Modular consistency (see Definition 3.4), and strong RSA (see Definition 3.6), and Discrete log assumption (see Definition 3.1).*

Moreover, let $(a_i, b_i)_{i=1}^3$ and witness components c and u_i-s be n-bit integers, and f be a function computable by a size s arithmetic circuit. $\mathsf{TPoKEDex}$ *has the following properties:*

- *Prover time is $O(\mathsf{MSM}(\log n, n) + \log n)$ group operations and $O(\mathsf{M}(n)\log^2 n)$ bit operations.*
- *Verifier time is $O(\lambda + \log n)$ group operations and $O(s\mathsf{M}(\lambda))$ bit operations.*
- *Proof size is upper bounded by $O(\log\log n)$ group elements and $O(\lambda \log\log n)$ bits.*

Proof. Completeness of the argument is straightforward to check. We defer the complexity analysis and proof of knowledge-soundness to Appendix D.4.

8 Self inner product arguments

To optimize our PoKEDEx protocol, we construct a bulletproofs [BBBPWM18] style ‘self inner product argument’ (SIPA) in groups of unknown order. In particular, we design an argument for the following relation:

Definition 8.1 (SIPA relation)**.** *Given* $\mathsf{pp} = (N, \mathbb{G}, \mathbf{G}) \leftarrow \mathsf{Setup}(1^\lambda, N)$*, we define the NP relation* $\mathscr{R}^{\mathsf{pp}}_{\mathsf{SIPA}}$ *to be the set of tuples*

$$\begin{pmatrix} \mathbb{x}, \\ \mathbb{w} \end{pmatrix} = \begin{pmatrix} (C, \ell, v), \\ \mathbf{r} \end{pmatrix}$$

where $C \in \mathbb{G}$*,* $\ell \in \mathbb{N}$ *is a prime larger than* 2^λ*,* $v \in \mathbb{Z}_\ell$*, and* $\mathbf{r} \in \mathbb{Z}^N$ *such that*

$$C = \langle \mathbf{r}, \mathbf{G} \rangle \qquad \text{and} \qquad v = \langle \mathbf{r}, \mathbf{r} \rangle \mod \ell = \sum_{i=1}^{N} r_i^2 \mod \ell$$

Remark 8.2. Like the protocols in Section 7, in our use cases of $\mathscr{R}_{\mathsf{SIPA}}$, we often only require an $\mathbf{r} \in \mathbb{Z}^m$, where $m \leq N$. However, since handling the case where $m \neq N$ is a straightforward extension of the current argument, we omit this nuance for ease of exposition.

8.1 SIPA reductions

In this section we present a reduction of knowledge that reduces an instance of the SIPA relation into another instance of half the length using a random chalenge from the verifier. Intuitively, given $\mathsf{pp} = (2^n, \mathbb{G}, \mathbf{G})$, we want to reduce the problem of checking if a tuple $(\mathbb{x}, \mathbb{w}) \in \mathscr{R}^{\mathsf{pp}}_{\mathsf{SIPA}}$ to the easier problem of checking if the tuple $(\mathbb{x}', \mathbb{w}') \in \mathscr{R}^{\mathsf{pp}'}_{\mathsf{SIPA}}$, where $\mathsf{pp}' := (2^{n-1}, \mathbb{G}, \mathbf{G}')$.

As a preliminary we define the following function fold, that takes as input a vector $\mathbf{G} \in \mathbb{G}^{2^n}$, an integer $i \in \mathbb{N}$ and challenges $\boldsymbol{\alpha} \in \mathbb{Z}^i$, and ‘folds’ $\mathbf{G}$ in half i times with respect to the challenges $\boldsymbol{\alpha}$. For $i = 0$, we define $\mathsf{fold}(\mathbf{G}, 0, \perp) := \mathbf{G}$. We also recall the notation $\mathbf{G}_L$ and $\mathbf{G}_R$ that denote the left and right halves of $\mathbf{G}$ respectively.

```
fold(G, i, α) → G'
--------------------------------
1. Set $\mathbf{G}' := \mathbf{G}$.
2. For each $j \in [i]$:
3.     Fold $\mathbf{G}' \leftarrow \alpha_j \cdot \mathbf{G}'_L + \mathbf{G}'_R$.
4. Output $\mathbf{G}'$.
```

To conform to the definition of a reduction of knowledge, we now define the following sequence of relations, with one relation for each $i \in \{0, 1, \ldots, n\}$.

Definition 8.3 (SIPA-i relation)**.** *Given* $\mathsf{pp} = (2^n, \mathbb{G}, \mathbf{G}) \leftarrow \mathsf{Setup}(1^\lambda, 2^n)$*, we define the NP relation* $\mathscr{R}^{\mathsf{pp}}_{\mathsf{SIPA}\text{-}i}$ *to be the set of tuples*

$$\begin{pmatrix} \mathbb{x}, \\ \mathbb{w} \end{pmatrix} = \begin{pmatrix} (C, \ell, v, \boldsymbol{\alpha}), \\ \mathbf{r} \end{pmatrix}$$

where $C \in \mathbb{G}$*,* $\ell \in \mathbb{N}$ *is a prime larger than* 2^λ*,* $v \in \mathbb{Z}_\ell$*,* $\boldsymbol{\alpha} \in [0, \ell)^i$ *and* $\mathbf{r} \in \mathbb{Z}^{2^{n-i}}$ *such that*

$$\mathbf{G}' := \mathsf{fold}(\mathbf{G}, i, \boldsymbol{\alpha}), \qquad C = \langle \mathbf{r}, \mathbf{G}' \rangle, \qquad v = \langle \mathbf{r}, \mathbf{r} \rangle \mod \ell$$

Note that $\mathscr{R}_{\mathsf{SIPA}\text{-}0}$ is precisely $\mathscr{R}_{\mathsf{SIPA}}$. We now present a reduction of knowledge $\mathsf{SIPA.Reduce}_i$ from $\mathscr{R}_{\mathsf{SIPA}\text{-}i}$ to $\mathscr{R}_{\mathsf{SIPA}\text{-}(i+1)}$.

```
SIPA.Reduce_i

⟨P(pp, x, w), V(pp, x)⟩:

Parse: pp = (2^n, G, G), x = (C, ℓ, v, α = [α_j]_{j=1}^i) and w = r.

1. P and V compute $\mathbf{G}' := \mathsf{fold}(\mathbf{G}, i, \boldsymbol{\alpha})$.
2. P computes $C_L := \langle \mathbf{r}_L, \mathbf{G}'_R \rangle$ and $C_R := \langle \mathbf{r}_R, \mathbf{G}'_L \rangle$.
3. P computes $v_L := \langle \mathbf{r}_L, \mathbf{r}_L \rangle \mod \ell$ and $v_C := \langle \mathbf{r}_L, \mathbf{r}_R \rangle \mod \ell$.
4. P sends $C_L, C_R, v_L, v_C$ to V.
5. V samples $\alpha_{i+1} \xleftarrow{\$} [0, \ell)$ and sends it to P.
6. P sets $\mathbf{r}' := \mathbf{r}_L + \alpha_{i+1} \cdot \mathbf{r}_R$.
7. P and V set
       $C' := C_L + \alpha_{i+1} \cdot C + \alpha_{i+1}^2 \cdot C_R,$
       $v_R := v - v_L \mod \ell,$
       $v' := v_L + 2 \cdot \alpha_{i+1} \cdot v_C + \alpha_{i+1}^2 \cdot v_R \mod \ell$
8. Define $\mathbb{x}' := (C', \ell, v', [\alpha_j]_{j=1}^{i+1})$ and $\mathbb{w}' := \mathbf{r}'$.
9. P receives output $(\mathbb{x}', \mathbb{w}')$ and V receives output $\mathbb{x}'$.
```

Lemma 8.4. *For $\mathcal{G} :=$ Setup and the $\mathcal{P}$, $\mathcal{V}$ above, define $\mathsf{SIPA.Reduce}_i := (\mathcal{G}, \mathcal{P}, \mathcal{V})$. If the discrete log assumption (Definition 3.1) and the strong RSA (Definition 3.6) hold, $\mathsf{SIPA.Reduce}_i$ is a reduction of knowledge from $\mathscr{R}_{\mathsf{SIPA}\text{-}i}$ to $\mathscr{R}_{\mathsf{SIPA}\text{-}(i+1)}$. The prover time is $O(2^n)$ group operations, the verifier time is $O(2^n)$ group operations, and the communication complexity is $2|\mathbb{G}|+2|\mathbb{Z}_\ell|$.*

We defer the proof of this lemma to Appendix E.2.1.

8.2 The full SIPA protocol

We now present the full argument for $\mathscr{R}_{\mathsf{SIPA}}^{\mathsf{pp}}$ for a given $\mathsf{pp} = (2^n, \mathbb{G}, \mathbf{G})$, which applies SIPA.Reduce iteratively to shrink the size of the instance to length 1, and then directly checks the instance of length 1 using a PoKE (which is slightly more efficient than the prover sending the witness in the clear to the verifier).

```
SIPA

⟨P(pp, x, w), V(pp, x)⟩:

Parse: pp = (2^n, G, G), x = (C, ℓ, v) and w = r.

1. Define $\mathbb{x}_0 := \mathbb{x}$ and $\mathbb{w}_0 := \mathbb{w}$.
2. For $i$ in $[0, \ldots, n-1]$:
3.     $(\mathbb{x}_{i+1}, \mathbb{w}_{i+1}) \leftarrow \mathsf{SIPA.Reduce}_i(\langle \mathcal{P}(\mathsf{pp}, \mathbb{x}_i, \mathbb{w}_i), \mathcal{V}(\mathsf{pp}, \mathbb{x}_i) \rangle)$.
4. Parse $\mathbb{x}_n = (C', \ell, v', \boldsymbol{\alpha})$ and $\mathbb{w}_n = r'$.
5. P sends $r'$ to V.
6. V accepts if
       $C' = r' \cdot \mathsf{fold}(\mathbf{G}, n, \boldsymbol{\alpha})$, and
       $v' = r' \cdot r' \mod \ell$
```

Theorem 8.5. *If the discrete log assumption (Definition 3.1) and the strong RSA assumption (Definition 3.6) hold, then $\mathsf{SIPA} := (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$, with $\mathcal{P}$ and $\mathcal{V}$ as above, is an argument of knowledge for $\mathscr{R}_{\mathsf{SIPA}}$. The prover time is $O(2^n)$ group operations, the verifier time is $O(2^n)$ group operations, and the communication complexity is $2n|\mathbb{G}|+2n|\mathbb{Z}_\ell|+|r'|$.*

Proof. For each $i \in [0, \ldots, n-1]$, given that the aforementioned assumptions hold, Lemma 8.4 shows that $\mathsf{SIPA.Reduce}_i$ is a reduction of knowledge from $\mathscr{R}_{\mathsf{SIPA}\text{-}i}$ to $\mathscr{R}_{\mathsf{SIPA}\text{-}(i+1)}$.

Since $\mathcal{V}$ directly checks that $(\mathbb{x}_n, \mathbb{w}_n) \in \mathscr{R}_{\mathsf{SIPA}\text{-}(n)}$, which is the trivial argument of knowledge for $\mathscr{R}_{\mathsf{SIPA}\text{-}(n)}$. Theorem 2.7 implies that SIPA is an argument of knowledge for $\mathscr{R}_{\mathsf{SIPA}}$.

The efficiency claims follows from the efficiency claims of $\mathsf{SIPA.Reduce}_i$, and the fact that the instance sizes keep going down by a factor of 2 for each $i \in [0, 1, \ldots, n-1]$, resulting in instance sizes $2^n, 2^{n-1}, \ldots, 2, 1$. $\square$

Communication complexity optimization. In our use case, and potentially in other applications, the integer r' might be too large for the prover to send directly to the verifier. In this case the final check can be performed using PoKEMath, with the prover using another randomly sampled generator, say H (H can also be included in pp), to commit to the quotient q such that $r' \cdot r' = v' + q \cdot \ell$. The prover sends the commitment $Q := q \cdot H$ to the verifier, and the verifier sends a random challenge $\beta \xleftarrow{\$} [0, 2^\lambda]$ to the prover. The prover and verifier then compute the combined commitment $\beta \cdot C' + Q$ to the vector (r', q) under the generators $(\beta \cdot \mathsf{fold}(\mathbf{G}, n, \boldsymbol{\alpha}), H)$. The prover can now use PoKEMath to convince the verifier that $f(r', q) := r' \cdot r' - v' - q \cdot \ell = 0$, which requires communicating 1 group element and 2 elements of $\mathbb{Z}_{\ell'}$, where $\ell' \in \mathsf{Primes}[\lambda, 2\lambda]$ is the PoKEMath challenge. This optimization does not affect prover or verifier efficiency asymptotically, and reduces the communication complexity to at most

$$(2n+1)|\mathbb{G}| + 2n|\mathbb{Z}_\ell| + \mathsf{cost}(\mathsf{PoKEMath}) \leq (2n+2)|\mathbb{G}| + 2n|\mathbb{Z}_\ell| + 4\lambda.$$

The random challenge β ensures that there is no overflow from the terms committed to using $\mathsf{fold}(\mathbf{G}, n, \boldsymbol{\alpha})$ and H. Commitments are 'stitched' together in this manner in both prior work (PoKE2 from [BBF19]), as well as in our PoKEDEx construction (Section 7.3).

8.3 Three SIPAs in parallel

In the construction of our polynomial commitment scheme, we would like to perform 3 SIPA protocols. In order to optimize our proof size, it would help to design a more efficient way to do this than running 3 separate SIPA protocols. Towards this end, we define the TIPA (Triple Inner Product Argument) relation as follows:

Definition 8.6 (TIPA relation). *Given* $\mathsf{pp} = (3N, \mathbb{G}, (\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_3)) \leftarrow \mathsf{Setup}(1^\lambda, 3N)$*, we define the NP relation* $\mathscr{R}^{\mathsf{pp}}_{\mathsf{TIPA}}$ *to be the set of tuples*

$$\begin{pmatrix} \mathbb{x}, \\ \mathbb{w} \end{pmatrix} = \begin{pmatrix} (C, \ell, (v_1, v_2, v_3)), \\ (\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3) \end{pmatrix}$$

where $(\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_3) \in \mathbb{G}^{3N}$*,* $C \in \mathbb{G}$*,* $\ell \in \mathbb{N}$ *is a prime larger than* 2^λ*,* $v_1, v_2, v_3 \in \mathbb{Z}_\ell$*, and* $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3 \in \mathbb{Z}^N$ *such that*

$$C = \langle \mathbf{r}_1, \mathbf{G}_1 \rangle + \langle \mathbf{r}_2, \mathbf{G}_2 \rangle + \langle \mathbf{r}_3, \mathbf{G}_3 \rangle, \quad \textit{and} \quad v_i = \langle \mathbf{r}_i, \mathbf{r}_i \rangle \mod \ell \quad \forall\, i \in [3]$$

Our TIPA protocol, which is an argument of knowledge for $\mathscr{R}_{\mathsf{TIPA}}$, proceeds similarly to the SIPA protocol, by iteratively reducing the size of the instances by half, and finally directly checking the obtained instance of constant size. We defer the construction and proofs to Appendix E.1, but state our final theorem below.

Theorem 8.7. *If the discrete log assumption (see Definition 3.1) and the strong RSA assumption (see Definition 3.6) hold, then* TIPA *is an argument of knowledge for* $\mathscr{R}_{\mathsf{TIPA}}$*. The prover time is* $O(2^n)$ *group operations, the verifier time is* $O(2^n)$ *group operations, and the communication complexity is* $2n|\mathbb{G}| + 6n|\mathbb{Z}_\ell| + 12\lambda$.

References

[ACFY24] G. Arnon, A. Chiesa, G. Fenzi, and E. Yogev. “STIR: Reed-Solomon Proximity Testing with Fewer Queries”. In: *CRYPTO 2024, Part X*. Ed. by L. Reyzin and D. Stebila. Vol. 14929. LNCS. Springer, Cham, Aug. 2024, pp. 380–413. DOI: 10.1007/978-3-031-68403-6_12.

[AGLMS23] A. Arun, C. Ganesh, S. V. Lokam, T. Mopuri, and S. Sridhar. “Dew: A Transparent Constant-Sized Polynomial Commitment Scheme”. In: *PKC 2023, Part II*. Ed. by A. Boldyreva and V. Kolesnikov. Vol. 13941. LNCS. Springer, Cham, May 2023, pp. 542–571. DOI: 10.1007/978-3-031-31371-4_19.

[AH85] K. E. Atkinson and W. Han. *Elementary numerical analysis*. Wiley New York, 1985.

[BBBF18] D. Boneh, J. Bonneau, B. Bünz, and B. Fisch. “Verifiable Delay Functions”. In: *CRYPTO 2018, Part I*. Ed. by H. Shacham and A. Boldyreva. Vol. 10991. LNCS. Springer, Cham, Aug. 2018, pp. 757–788. DOI: 10.1007/978-3-319-96884-1_25.

[BBBPWM18] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. “Bulletproofs: Short Proofs for Confidential Transactions and More”. In: *2018 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2018, pp. 315–334. DOI: 10.1109/SP.2018.00020.

[BBCdGL18] C. Baum, J. Bootle, A. Cerulli, R. del Pino, J. Groth, and V. Lyubashevsky. “Sub-linear Lattice-Based Zero-Knowledge Arguments for Arithmetic Circuits”. In: *CRYPTO 2018, Part II*. Ed. by H. Shacham and A. Boldyreva. Vol. 10992. LNCS. Springer, Cham, Aug. 2018, pp. 669–699. DOI: 10.1007/978-3-319-96881-0_23.

[BBF19] D. Boneh, B. Bünz, and B. Fisch. “Batching Techniques for Accumulators with Applications to IOPs and Stateless Blockchains”. In: *CRYPTO 2019, Part I*. Ed. by A. Boldyreva and D. Micciancio. Vol. 11692. LNCS. Springer, Cham, Aug. 2019, pp. 561–586. DOI: 10.1007/978-3-030-26948-7_20.

[BBHR18] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev. “Fast Reed-Solomon Interactive Oracle Proofs of Proximity”. In: *ICALP 2018*. Ed. by I. Chatzigiannakis, C. Kaklamanis, D. Marx, and D. Sannella. Vol. 107. LIPIcs. Schloss Dagstuhl, July 2018, 14:1–14:17. DOI: 10.4230/LIPIcs.ICALP.2018.14.

[BBHR19] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev. “Scalable Zero Knowledge with No Trusted Setup”. In: *CRYPTO 2019, Part III*. Ed. by A. Boldyreva and D. Micciancio. Vol. 11694. LNCS. Springer, Cham, Aug. 2019, pp. 701–732. DOI: 10.1007/978-3-030-26954-8_23.

[BCCGP16] J. Bootle, A. Cerulli, P. Chaidos, J. Groth, and C. Petit. “Efficient Zero-Knowledge Arguments for Arithmetic Circuits in the Discrete Log Setting”. In: *EUROCRYPT 2016, Part II*. Ed. by M. Fischlin and J.-S. Coron. Vol. 9666. LNCS. Springer, Berlin, Heidelberg, May 2016, pp. 327–357. DOI: 10.1007/978-3-662-49896-5_12.

[BCS21] J. Bootle, A. Chiesa, and K. Sotiraki. “Sumcheck Arguments and Their Applications”. In: *CRYPTO 2021, Part I*. Ed. by T. Malkin and C. Peikert. Vol. 12825. LNCS. Virtual Event: Springer, Cham, Aug. 2021, pp. 742–773. DOI: 10.1007/978-3-030-84242-0_26.

[BCTV14] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza. “Scalable Zero Knowledge via Cycles of Elliptic Curves”. In: *CRYPTO 2014, Part II*. Ed. by J. A. Garay and R. Gennaro. Vol. 8617. LNCS. Springer, Berlin, Heidelberg, Aug. 2014, pp. 276–294. DOI: 10.1007/978-3-662-44381-1_16.

[Ben+14] E. Ben-Sasson et al. “Zerocash: Decentralized Anonymous Payments from Bitcoin”. In: *2014 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2014, pp. 459–474. DOI: 10.1109/SP.2014.36.

[BF10] R. L. Burden and J. D. Faires. *Numerical analysis*. 2010.

[BF23] B. Bünz and B. Fisch. “Multilinear Schwartz-Zippel Mod N and Lattice-Based Succinct Arguments”. In: *TCC 2023, Part III*. Ed. by G. N. Rothblum and H. Wee. Vol. 14371. LNCS. Springer, Cham, 2023, pp. 394–423. DOI: 10.1007/978-3-031-48621-0_14.

[BFS19] B. Bünz, B. Fisch, and A. Szepieniec. *Transparent SNARKs from DARK Compilers*. Cryptology ePrint Archive, Report 2019/1229. 2019. URL: https://eprint.iacr.org/2019/1229.

[BFS20] B. Bünz, B. Fisch, and A. Szepieniec. “Transparent SNARKs from DARK Compilers”. In: *EUROCRYPT 2020, Part I*. Ed. by A. Canteaut and Y. Ishai. Vol. 12105. LNCS. Springer, Cham, May 2020, pp. 677–706. DOI: 10.1007/978-3-030-45721-1_24.

[BISW17] D. Boneh, Y. Ishai, A. Sahai, and D. J. Wu. *Lattice-Based SNARGs and Their Application to More Efficient Obfuscation*. Cryptology ePrint Archive, Report 2017/240. 2017. URL: https://eprint.iacr.org/2017/240.

[BM94] J. Benaloh and M. de Mare. "One-Way Accumulators: A Decentralized Alternative to Digital Signatures". In: *Advances in Cryptology — EUROCRYPT '93*. Ed. by T. Helleseth. Berlin, Heidelberg: Springer Berlin Heidelberg, 1994, pp. 274–285. ISBN: 978-3-540-48285-7.

[BP97] N. Bari and B. Pfitzmann. "Collision-Free Accumulators and Fail-Stop Signature Schemes Without Trees". In: *EUROCRYPT'97*. Ed. by W. Fumy. Vol. 1233. LNCS. Springer, Berlin, Heidelberg, May 1997, pp. 480–494. DOI: 10.1007/3-540-69053-0_33.

[Bre00] R. P. Brent. "Public Key Cryptography with a Group of Unknown Order". In: 2000. URL: https://api.semanticscholar.org/CorpusID:14178879.

[BW88] J. Buchmann and H. C. Williams. "A Key-Exchange System Based on Imaginary Quadratic Fields". In: *Journal of Cryptology* 1.2 (June 1988), pp. 107–118. DOI: 10.1007/BF02351719.

[CA69] S. A. Cook and S. O. Aanderaa. "On the minimum computation time of functions". In: *Transactions of the American Mathematical Society* 142 (1969), pp. 291–314.

[CBBZ23] B. Chen, B. Bünz, D. Boneh, and Z. Zhang. "HyperPlonk: Plonk with Linear-Time Prover and High-Degree Custom Gates". In: *EUROCRYPT 2023, Part II*. Ed. by C. Hazay and M. Stam. Vol. 14005. LNCS. Springer, Cham, Apr. 2023, pp. 499–530. DOI: 10.1007/978-3-031-30617-4_17.

[CGKR22] G. Couteau, D. Goudarzi, M. Klooß, and M. Reichle. "Sharp: Short Relaxed Range Proofs". In: *ACM CCS 2022*. Ed. by H. Yin, A. Stavrou, C. Cremers, and E. Shi. ACM Press, Nov. 2022, pp. 609–622. DOI: 10.1145/3548606.3560628.

[CHMMVW20] A. Chiesa, Y. Hu, M. Maller, P. Mishra, P. Vesely, and N. P. Ward. "Marlin: Preprocessing zkSNARKs with Universal and Updatable SRS". In: *EUROCRYPT 2020, Part I*. Ed. by A. Canteaut and Y. Ishai. Vol. 12105. LNCS. Springer, Cham, May 2020, pp. 738–768. DOI: 10.1007/978-3-030-45721-1_26.

[CKLR21] G. Couteau, M. Klooß, H. Lin, and M. Reichle. "Efficient Range Proofs with Transparent Setup from Bounded Integer Commitments". In: *EUROCRYPT 2021, Part III*. Ed. by A. Canteaut and F.-X. Standaert. Vol. 12698. LNCS. Springer, Cham, Oct. 2021, pp. 247–277. DOI: 10.1007/978-3-030-77883-5_9.

[CPP17] G. Couteau, T. Peters, and D. Pointcheval. "Removing the Strong RSA Assumption from Arguments over the Integers". In: *EUROCRYPT 2017, Part II*. Ed. by J.-S. Coron and J. B. Nielsen. Vol. 10211. LNCS. Springer, Cham, 2017, pp. 321–350. DOI: 10.1007/978-3-319-56614-6_11.

[DGS22] S. Dobson, S. Galbraith, and B. Smith. "Trustless unknown-order groups". In: *Mathematical Cryptology* 1.2 (2022), 25–39. URL: https://journals.flvc.org/mathcryptology/article/view/130579.

[DK02a] I. Damgard and M. Koprowski. *Generic Lower Bounds for Root Extraction and Signature Schemes in General Groups*. Cryptology ePrint Archive, Report 2002/013. 2002. URL: https://eprint.iacr.org/2002/013.

[DK02b] I. Damgård and M. Koprowski. "Generic Lower Bounds for Root Extraction and Signature Schemes in General Groups". In: *EUROCRYPT 2002*. Ed. by L. R. Knudsen. Vol. 2332. LNCS. Springer, Berlin, Heidelberg, 2002, pp. 256–271. DOI: 10.1007/3-540-46035-7_17.

[DSV03] G. P. Davidoff, P. Sarnak, and A. Valette. *Elementary number theory, group theory, and Ramanujan graphs*. Vol. 55. 1. Cambridge university press Cambridge, 2003.

[FO97] E. Fujisaki and T. Okamoto. "Statistical Zero Knowledge Protocols to Prove Modular Polynomial Relations". In: *CRYPTO'97*. Ed. by B. S. Kaliski Jr. Vol. 1294. LNCS. Springer, Berlin, Heidelberg, Aug. 1997, pp. 16–30. DOI: 10.1007/BFb0052225.

[GG03] J. V. Z. Gathen and J. Gerhard. *Modern Computer Algebra*. 2nd ed. USA: Cambridge University Press, 2003. ISBN: 0521826462.

[GGPR13] R. Gennaro, C. Gentry, B. Parno, and M. Raykova. "Quadratic Span Programs and Succinct NIZKs without PCPs". In: *EUROCRYPT 2013*. Ed. by T. Johansson and P. Q. Nguyen. Vol. 7881. LNCS. Springer, Berlin, Heidelberg, May 2013, pp. 626–645. DOI: 10.1007/978-3-642-38348-9_37.

[GKR08] S. Goldwasser, Y. T. Kalai, and G. N. Rothblum. “Delegating computation: interactive proofs for muggles”. In: *40th ACM STOC*. Ed. by R. E. Ladner and C. Dwork. ACM Press, May 2008, pp. 113–122. DOI: 10.1145/1374376.1374396.

[GMR85] S. Goldwasser, S. Micali, and C. Rackoff. “The Knowledge Complexity of Interactive Proof-Systems (Extended Abstract)”. In: *17th ACM STOC*. ACM Press, May 1985, pp. 291–304. DOI: 10.1145/22145.22178.

[Gro05] J. Groth. “Non-interactive Zero-Knowledge Arguments for Voting”. In: *ACNS 05International Conference on Applied Cryptography and Network Security*. Ed. by J. Ioannidis, A. Keromytis, and M. Yung. Vol. 3531. LNCS. Springer, Berlin, Heidelberg, June 2005, pp. 467–482. DOI: 10.1007/11496137_32.

[Gro16] J. Groth. “On the Size of Pairing-Based Non-interactive Arguments”. In: *EUROCRYPT 2016, Part II*. Ed. by M. Fischlin and J.-S. Coron. Vol. 9666. LNCS. Springer, Berlin, Heidelberg, May 2016, pp. 305–326. DOI: 10.1007/978-3-662-49896-5_11.

[GW11] C. Gentry and D. Wichs. “Separating succinct non-interactive arguments from all falsifiable assumptions”. In: *43rd ACM STOC*. Ed. by L. Fortnow and S. P. Vadhan. ACM Press, June 2011, pp. 99–108. DOI: 10.1145/1993636.1993651.

[GWC19] A. Gabizon, Z. J. Williamson, and O. Ciobotaru. *PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge*. Cryptology ePrint Archive, Report 2019/953. 2019. URL: https://eprint.iacr.org/2019/953.

[HH21] D. Harvey and J. van der Hoeven. “Integer multiplication in time $O(n \log n)$”. In: *Annals of Mathematics* 193.2 (2021), pp. 563 –617. DOI: 10.4007/annals.2021.193.2.4. URL: https://doi.org/10.4007/annals.2021.193.2.4.

[KC09] D. R. Kincaid and E. W. Cheney. *Numerical analysis: mathematics of scientific computing*. Vol. 2. American Mathematical Soc., 2009.

[KCLM22] I. Khaburzaniya, K. Chalkias, K. Lewi, and H. Malvai. “Aggregating and Thresholdizing Hash-based Signatures using STARKs”. In: *ASIACCS 22*. Ed. by Y. Suga, K. Sakurai, X. Ding, and K. Sako. ACM Press, 2022, pp. 393–407. DOI: 10.1145/3488932.3524128.

[Kil92] J. Kilian. “A Note on Efficient Zero-Knowledge Proofs and Arguments (Extended Abstract)”. In: *24th ACM STOC*. ACM Press, May 1992, pp. 723–732. DOI: 10.1145/129712.129782.

[KM03] S. Kwek and K. Mehlhorn. “Optimal search for rationals”. In: *Information Processing Letters* 86.1 (2003), pp. 23–26. ISSN: 0020-0190. DOI: https://doi.org/10.1016/S0020-0190(02)00455-6. URL: https://www.sciencedirect.com/science/article/pii/S0020019002004556.

[Knu98] D. E. Knuth. *The art of computer programming, Volume II: Seminumerical Algorithms, 3rd Edition*. Addison-Wesley, 1998. ISBN: 0201896842. URL: https://www.worldcat.org/oclc/312898417.

[KP23] A. Kothapalli and B. Parno. “Algebraic Reductions of Knowledge”. In: *CRYPTO 2023, Part IV*. Ed. by H. Handschuh and A. Lysyanskaya. Vol. 14084. LNCS. Springer, Cham, Aug. 2023, pp. 669–701. DOI: 10.1007/978-3-031-38551-3_21.

[KPV22] A. A. Kattis, K. Panarin, and A. Vlasov. “RedShift: Transparent SNARKs from List Polynomial Commitments”. In: *ACM CCS 2022*. Ed. by H. Yin, A. Stavrou, C. Cremers, and E. Shi. ACM Press, Nov. 2022, pp. 1725–1737. DOI: 10.1145/3548606.3560657.

[KZG10] A. Kate, G. M. Zaverucha, and I. Goldberg. “Constant-Size Commitments to Polynomials and Their Applications”. In: *ASIACRYPT 2010*. Ed. by M. Abe. Vol. 6477. LNCS. Springer, Berlin, Heidelberg, Dec. 2010, pp. 177–194. DOI: 10.1007/978-3-642-17373-8_11.

[LAN01] H. Lipmaa, N. Asokan, and V. Niemi. *Secure Vickrey Auctions without Threshold Trust*. Cryptology ePrint Archive, Report 2001/095. 2001. URL: https://eprint.iacr.org/2001/095.

[Lee21] J. Lee. “Dory: Efficient, Transparent Arguments for Generalised Inner Products and Polynomial Commitments”. In: *TCC 2021, Part II*. Ed. by K. Nissim and B. Waters. Vol. 13043. LNCS. Springer, Cham, Nov. 2021, pp. 1–34. DOI: 10.1007/978-3-030-90453-1_1.

[Lip03] H. Lipmaa. “On Diophantine Complexity and Statistical Zero-Knowledge Arguments”. In: *ASIACRYPT 2003*. Ed. by C.-S. Laih. Vol. 2894. LNCS. Springer, Berlin, Heidelberg, 2003, pp. 398–415. DOI: 10.1007/978-3-540-40061-5_26.

[Lip12] H. Lipmaa. "Secure Accumulators from Euclidean Rings without Trusted Setup". In: *ACNS 12International Conference on Applied Cryptography and Network Security*. Ed. by F. Bao, P. Samarati, and J. Zhou. Vol. 7341. LNCS. Springer, Berlin, Heidelberg, June 2012, pp. 224–240. DOI: 10.1007/978-3-642-31284-7_14.

[LLX07] J. Li, N. Li, and R. Xue. "Universal Accumulators with Efficient Nonmembership Proofs". In: *ACNS 07International Conference on Applied Cryptography and Network Security*. Ed. by J. Katz and M. Yung. Vol. 4521. LNCS. Springer, Berlin, Heidelberg, June 2007, pp. 253–269. DOI: 10.1007/978-3-540-72738-5_17.

[LPS24] H. Lipmaa, R. Parisella, and J. Siim. "Constant-Size zk-SNARKs in ROM from Falsifiable Assumptions". In: *EUROCRYPT 2024, Part VI*. Ed. by M. Joye and G. Leander. Vol. 14656. LNCS. Springer, Cham, May 2024, pp. 34–64. DOI: 10.1007/978-3-031-58751-1_2.

[LXZSZ23] T. Liu, T. Xie, J. Zhang, D. Song, and Y. Zhang. *Pianist: Scalable zkRollups via Fully Distributed Zero-Knowledge Proofs*. Cryptology ePrint Archive, Report 2023/1271. 2023. URL: https://eprint.iacr.org/2023/1271.

[MBKM19] M. Maller, S. Bowe, M. Kohlweiss, and S. Meiklejohn. "Sonic: Zero-Knowledge SNARKs from Linear-Size Universal and Updatable Structured Reference Strings". In: *ACM CCS 2019*. Ed. by L. Cavallaro, J. Kinder, X. Wang, and J. Katz. ACM Press, Nov. 2019, pp. 2111–2128. DOI: 10.1145/3319535.3339817.

[NT16] A. Naveh and E. Tromer. "PhotoProof: Cryptographic Image Authentication for Any Set of Permissible Transformations". In: *2016 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2016, pp. 255–271. DOI: 10.1109/SP.2016.23.

[Pie19] K. Pietrzak. "Simple Verifiable Delay Functions". In: *ITCS 2019*. Ed. by A. Blum. Vol. 124. LIPIcs, Jan. 2019, 60:1–60:15. DOI: 10.4230/LIPIcs.ITCS.2019.60.

[PT18] P. Pollack and E. Treviño. "Finding the Four Squares in Lagrange's Theorem." In: *Integers* 18.A15 (2018), pp. 7–17.

[Rei89] J. H. Reif. "Optimal size integer division circuits". In: *Proceedings of the twenty-first annual ACM symposium on Theory of computing*. 1989, pp. 264–273.

[RS86] M. O. Rabin and J. O. Shallit. "Randomized algorithms in number theory". In: *Communications on Pure and Applied Mathematics* 39.S1 (1986), S239–S256.

[RSW96] R. L. Rivest, A. Shamir, and D. A. Wagner. *Time-lock Puzzles and Timed-release Crypto*. Tech. rep. USA, 1996.

[SB23] I. A. Seres and P. Burcsi. *Behemoth: transparent polynomial commitment scheme with constant opening proof size and verifier time*. Cryptology ePrint Archive, Report 2023/670. 2023. URL: https://eprint.iacr.org/2023/670.

[SDW08] D. Sandler, K. Derr, and D. S. Wallach. "VoteBox: A Tamper-evident, Verifiable Electronic Voting System". In: *USENIX Security 2008*. Ed. by P. C. van Oorschot. USENIX Association, 2008, pp. 349–364.

[Set20] S. Setty. "Spartan: Efficient and General-Purpose zkSNARKs Without Trusted Setup". In: *CRYPTO 2020, Part III*. Ed. by D. Micciancio and T. Ristenpart. Vol. 12172. LNCS. Springer, Cham, Aug. 2020, pp. 704–737. DOI: 10.1007/978-3-030-56877-1_25.

[STW23] S. Setty, J. Thaler, and R. Wahby. *Customizable constraint systems for succinct arguments*. Cryptology ePrint Archive, Report 2023/552. 2023. URL: https://eprint.iacr.org/2023/552.

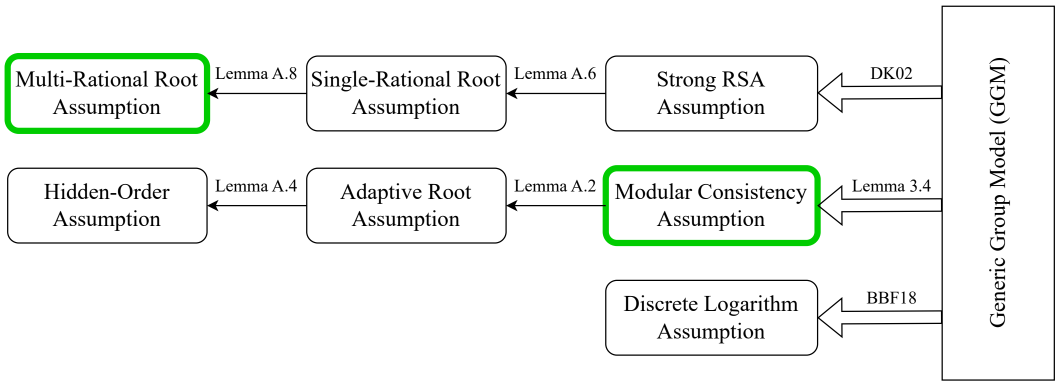
[Wes19] B. Wesolowski. "Efficient Verifiable Delay Functions". In: *EUROCRYPT 2019, Part III*. Ed. by Y. Ishai and V. Rijmen. Vol. 11478. LNCS. Springer, Cham, May 2019, pp. 379–407. DOI: 10.1007/978-3-030-17659-4_13.

[ZXZS20] J. Zhang, T. Xie, Y. Zhang, and D. Song. "Transparent Polynomial Delegation and Its Applications to Zero Knowledge Proof". In: *2020 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2020, pp. 859–876. DOI: 10.1109/SP40000.2020.00052.

A Deferred discussions on assumptions

A.1 Implied assumptions

In Section 3.2 and consequently in the main body, we focus only on the strongest assumptions underlying our work. However, to prove the security of certain constructions, it is more convenient to work with weaker assumptions that are implied by these stronger ones. Here, we present these weaker assumptions and demonstrate how they follow from the stronger ones.



Multi-Rational Root Assumption
Lemma A.8
Single-Rational Root Assumption
Lemma A.6
Strong RSA Assumption
DK02
Hidden-Order Assumption
Lemma A.4
Adaptive Root Assumption
Lemma A.2
Modular Consistency Assumption
Lemma 3.4
Discrete Logarithm Assumption
BBF18
Generic Group Model (GGM)

Figure 1: Assumptions used in this work. New assumptions are highlighted

A.1.1 Adaptive root assumption

At a high level, the adaptive root assumption says that no efficient adversary, with non-negligible probability, can output a random prime ℓ^{th} root of a chosen group element W. A more formal statement is as follows.

Definition A.1 (Adaptive root assumption [Wes19])**.** *We say that the adaptive root assumption holds for* GGen *if for all efficient stateful adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*,* $\mathsf{Adv}^{\mathsf{AR}}_{\mathcal{A}}(\lambda) \leq \text{negl}(\lambda)$*, where*

$$\mathsf{Adv}^{\mathsf{AR}}_{\mathcal{A}}(\lambda) := \Pr\left[\ell \cdot X = Y \neq 0 \;\middle|\; \begin{array}{l} \mathbb{G} \leftarrow \mathsf{GGen}(1^\lambda) \\ Y \leftarrow \mathcal{A}_1(\mathbb{G}) \\ \ell \leftarrow \mathsf{Primes}[\lambda, 2\lambda] \\ X \leftarrow \mathcal{A}_2(\ell) \end{array} \right]$$

Lemma A.2. *The modular consistency assumption implies the adaptive root assumption, meaning it is strictly stronger than the adaptive root assumption.*

Proof. Suppose there exists an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that breaks the adaptive root assumption with non-negligible probability. We will show that we can construct an adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ that breaks the modular consistency assumption.

$\mathcal{B}_1(\mathbb{G}, \boldsymbol{G}) \to U$

1. Run $\mathcal{A}_1$ on input $\mathbb{G}$ to get Y and the internal adaptive root state AR.state.
2. Output $U := Y$ and pass AR.state to $\mathcal{B}_2$.

By construction $\mathcal{B}_1$ runs in at most $\text{poly}(\lambda)$ steps. We can define T in terms of the running time of $\mathcal{A}_1$.

```
B_2 → [r_i, Q_i, ℓ_i]_{i=1}^T
1. Prase the internal state as AR.state.
2. Initialize an empty list of size $T$.
3. For $i \in [T]$:
4.     Sample $\ell_i \xleftarrow{\$} \mathsf{Primes}[\lambda, 2\lambda]$.
5.     Set $X_i \leftarrow \mathcal{A}_2(\ell)$ on internal state AR.state.
6.     Append $(0, X_i, \ell_i)$ to the list.
7. Output the list.
```

Clearly, $\mathcal{B}_2$ also runs in polynomial time. By a counting argument, notice that for a non-negligible fraction of Y output by $\mathcal{A}_1$, there must be a non-negligible number of ℓ_i such that $\mathcal{A}_2$ outputs a valid X_i. If this were not the case, then the probability of success of the adversary $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{AR}}(\lambda)$ would be negligible.

Now, we need to actually check that the modular consistency assumption is broken. Since all remainders are set to 0, the CRT solution $x = \mathbf{0}$. By construction, each X_i is such that $\ell_i \cdot X_i = Y$. Since $Y \neq 0$, we have that $\langle \mathbf{0}, \mathbf{G} \rangle = 0 \neq Y$. Thus, $\mathcal{B}$ breaks the modular consistency assumption. □

A.1.2 Hidden order assumption

The hidden order assumption states that an efficient adversary cannot compute a multiple of the order of a given random group element. A more formal statement of the assumption is as follows.

Definition A.3 (Hidden order assumption)**.** *For a group of unknown order $\mathbb{G} \leftarrow \mathsf{GGen}(1^\lambda)$, the Order assumption holds if for any adversary $\mathcal{A}$:*

$$\Pr\left[W \neq 0 \wedge x \cdot W = 0 \;\middle|\; \begin{array}{l} \mathbb{G} \leftarrow \mathsf{GGen}(1^\lambda) \\ \mathcal{A}(\mathbb{G}) \to (W, x) \\ \textit{where } |x| \leq 2^{\text{poly}(\lambda)} \textit{ and } W \in \mathbb{G} \end{array} \right]$$

Lemma A.4. *[BBF19, Lemma 2] The adaptive root assumption implies the hidden order assumption, meaning it is strictly stronger than the hidden order assumption.*

A.1.3 Single rational root assumption

Definition A.5 (Single rational root assumption)**.** GGen *satisfies the single rational root assumption if for all efficient $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{SRR}}(\lambda) \leq \text{negl}(\lambda)$, where:*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{SRR}}(\lambda) := \Pr\left[\begin{array}{c} m \cdot U = r \cdot G \\ \wedge \\ m \textit{ and } r \textit{ are coprime} \\ \wedge \\ m \neq 1 \end{array} \;\middle|\; \begin{array}{l} \mathbb{G} \leftarrow \mathsf{GGen}(1^\lambda) \\ G \xleftarrow{\$} \mathbb{G} \\ (U, m, r) \in \mathbb{G} \times \mathbb{N} \times \mathbb{N} \leftarrow \mathcal{A}(\mathbb{G}, G) \end{array} \right]$$

We now prove the following lemma.

Lemma A.6. *The strong RSA assumption implies the single rational root assumption, meaning it is strictly stronger than the single rational root assumption.*

Proof. We will prove the contrapositive: given an adversary $\mathcal{A}$ that breaks the single rational root assumption, we show that we can construct an adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ that breaks the strong RSA assumption. The adversary $\mathcal{B}_1$ works as follows:

```
$\mathcal{B}(\mathbb{G}, G) \to (U, \ell)$
---------------------------------------------
1. Obtain $(U', m, r) \leftarrow \mathcal{A}(\mathbb{G}, G)$.
2. Compute the Bezout coefficients $a, b$ such that
       $am + br = 1$
3. Set $U := b \cdot U' + a \cdot G$ and $\ell := m$.
4. Output $(U, \ell)$.
```

The adversary $\mathcal{A}(\mathbb{G}, G)$ with non-negligible probability outputs U', m, r such that $m \neq 1$ and m and r are co-prime and $m \cdot U' = r \cdot G$. $\mathcal{B}$ then essentially uses Shamir's trick: for $U := b \cdot U' + a \cdot G$, clearly

$$\begin{aligned} m \cdot U &= m \cdot (b \cdot U' + a \cdot G) \\ &= b \cdot m \cdot U' + a \cdot m \cdot G \\ &= b \cdot r \cdot G + a \cdot m \cdot G \\ &= (am + br) \cdot G = 1 \cdot G \end{aligned}$$

Since $\mathcal{B}$ succeeds in outputting such a tuple with non-negligible probability, it breaks the strong RSA assumption. $\square$

A.1.4 Multi-rational root assumption

We now state our second new assumption, the multi-rational root assumption, which is implied by the strong RSA assumption. At a high level, it states that for any $n \in \mathbb{N}$, no efficient adversary $\mathcal{A}$, except with negligible probability, can find a non-trivial 'rational root' of a vector of randomly sampled generators $\mathbf{G} \xleftarrow{\$} \mathbb{G}^n$. A group element U is said to be an $(m, \mathbf{r})$'th rational root of $\mathbf{G}$ if $m \cdot U = \langle \mathbf{r}, \mathbf{G} \rangle$ and m and r_i are coprime, for some $i \in [n]$. It is additionally said to be non-trivial if $m \neq 1$.

Definition A.7 (Multi-rational root assumption). GGen *satisfies the multi-rational root assumption if, for all* $n \in \mathbb{N} = \mathrm{poly}(\lambda)$ *and all efficient* $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{MRR}}_{\mathcal{A},n}(\lambda) \leq \mathrm{negl}(\lambda)$, *where* $\mathsf{Adv}^{\mathsf{MRR}}_{\mathcal{A},n}(\lambda) :=$

$$\Pr\left[\begin{array}{c|l} m \cdot U = \langle \mathbf{r}, \mathbf{G} \rangle & \\ \wedge & \mathbb{G} \leftarrow \mathsf{GGen}(1^\lambda) \\ \exists i \in [N] : m \textit{ and } r_i \textit{ are coprime} & \mathbf{G} \xleftarrow{\$} \mathbb{G}^n \\ \wedge & (U, m, \mathbf{r}) \in \mathbb{G} \times \mathbb{N} \times \mathbb{N}^n \leftarrow \mathcal{A}(\mathbb{G}, \mathbf{G}) \\ m \neq 1 & \end{array}\right]$$

We now state our main lemma about the multi-rational root assumption, but defer the proof to Appendix A.1.4.

Lemma A.8. *The strong RSA assumption implies the multi-rational root assumption, meaning it is strictly stronger than the multi-rational root assumption.*

Proof. Given an adversary $\mathcal{A}$ that breaks the multi-rational root assumption for some particular $n = \mathrm{poly}(\lambda)$, we show that we can construct an adversary $\mathcal{B}$ that breaks the single rational root assumption. This along with Lemma A.6 implies that the strong RSA assumption implies the strong rational root assumption. The adversary $\mathcal{B}$ (parametrized by a bound B that we fix later) works as follows:

```
$\mathcal{B}(\mathbb{G}, G) \to (U, m, r)$

1. For $i \in [n]$:
2.     Sample $\alpha_i \xleftarrow{\$} [0, B)$.
3.     Set $G_i := \alpha_i \cdot G$.
4. Obtain $(U', m', \mathbf{r}') \leftarrow \mathcal{A}(\mathbb{G}, \mathbf{G})$.
5. Compute coprime $m$ and $r$ such that:
       $\frac{r}{m} = \sum_{i=1}^{n} \frac{\alpha_i r'_i}{m'}$
6. Output $(U := U', m, r)$.
```

The adversary $\mathcal{A}(\mathbb{G}, \mathbf{G})$ with non-negligible probability outputs $U', m', \mathbf{r}'$ such that $m \neq 1$, $r_i < m$ for all $i \in [n]$, m is co-prime with at least one r_i, say for $i = i'$, and

$$m' \cdot U' = \langle \mathbf{r}', \mathbf{G} \rangle$$

Replacing G_i with $\alpha_i \cdot G$ for each $i \in [n]$, we get:

$$m' \cdot U' = \left(\sum_{i=1}^{n} r'_i \cdot \alpha_i \right) \cdot G$$

Let $a := \gcd(\sum_{i=1}^{n} \alpha_i r'_i, m')$. Since r and m are coprime numbers such that $\frac{r}{m} = \frac{\sum_{i=1}^{n} \alpha_i r'_i}{m'}$, if it were not the case that $m \cdot U' = r \cdot G$, then

$$\begin{aligned} & m \cdot U' \neq r \cdot G \\ \Longrightarrow\ & a \cdot m \cdot U' \neq a \cdot r \cdot G \\ \Longrightarrow\ & m' \cdot U' \neq \sum_{i=1}^{n} \alpha_i r'_i \cdot G \quad \text{(a contradiction)} \end{aligned}$$

Thus it must be the case that $m \cdot U' = r \cdot G$.

Probability of $\mathcal{B}$ succeeding over choice of $\boldsymbol{\alpha}$. If $m \neq 1$, then $\mathcal{B}$ has succeeded in breaking the single rational root assumption. That is, if $\gcd(m, \sum_{i=1}^{n} r_i \cdot \alpha_i) < m$, $\mathcal{B}$ succeeds. Or equivalently, if $\sum_{i=1}^{n} \alpha_i \cdot r_i \neq 0 \mod m$.

Thus, it is enough to upper bound the probability that $\sum_{i=1}^{n} \alpha_i \cdot r_i = 0 \mod m$, over the choice of $\boldsymbol{\alpha} := [\alpha_i]_{i=1}^{n}$. Since $m \neq 1$ it has a prime factor p. Since for any $x = 0 \mod m$, it must also be that $x = 0 \mod p$, the aforementioned probability is upper bounded by

$$\Pr_{\boldsymbol{\alpha} \xleftarrow{\$} [0,B)^n} [\sum_{i=1}^{n} \alpha_i \cdot r_i = 0 \mod p]$$

We know that there exists an $r_{i'}$ that is coprime with m, which means it must also be coprime with p. For any fixed value of $\boldsymbol{\alpha}' := [\alpha_i]_{i \in [n], i \neq i'}$, let $s_{\boldsymbol{\alpha}'} := r_{i'}^{-1} \cdot (\sum_{i \in [n], i \neq i'} \alpha_i \cdot r_i) \mod p$. The above probability now is equal to:

$$\Pr_{\boldsymbol{\alpha} \xleftarrow{\$} [0,B)^n} [\alpha_{i'} = -s_{\boldsymbol{\alpha}'} \mod p] = \Pr_{\alpha_{i'} \xleftarrow{\$} [0,B)} [\alpha_{i'} = c \mod p]$$

The equality follows from the observation that the probability on the left is precisely the average over all choices of $\boldsymbol{\alpha}'$ of the probability on the right, with $k = -s_{\boldsymbol{\alpha}'}$.

Since $\alpha_{i'}$ is sampled from the space $[0, B)$, if $p|B$, then $\alpha_{i'}$ is uniform $\mod p$ (each possible value mod p appears the same number of times). More generally, let $t := B \mod p$. Then if $\alpha_{i'} \xleftarrow{\$} [0, B-t)$, again it looks uniform, but if $\alpha_{i'} \xleftarrow{\$} [B-t, B)$, the probability of it being $c \mod p$ is at most $1/t$ (at most one value that is equal to $-k \mod p$). Thus, we have:

$$\begin{aligned}
\Pr_{\boldsymbol{\alpha} \xleftarrow{\$} [0,B)^n} [\alpha_{i'} = -s_{\boldsymbol{\alpha}'} \mod p] &= \Pr_{\alpha_{i'} \xleftarrow{\$} [0,B)} [\alpha_{i'} = c \mod p] \\
&= \left(1 - \frac{t}{B}\right) \Pr_{\alpha_{i'} \xleftarrow{\$} [0,B-t)} [\alpha_{i'} = c \mod p] \\
&\quad + \left(\frac{t}{B}\right) \Pr_{\alpha_{i'} \xleftarrow{\$} [B-t,B)} [\alpha_{i'} = c \mod p] \\
&\leq \left(1 - \frac{t}{B}\right) \frac{1}{p} + \left(\frac{t}{B}\right) \frac{1}{t} \\
&\leq \frac{1}{p} + \left(\frac{1}{B} - \frac{t}{Bp}\right) \leq \frac{1}{p} + \frac{1}{B}
\end{aligned}$$

Thus, the probability that the m output by $\mathcal{B}$ remains greater than 1, given that $\mathcal{A}$ succeeds in breaking the single rational root assumption is lower bounded by $1 - \Pr_{\boldsymbol{\alpha} \xleftarrow{\$} [0,B)^n} [\alpha_{i'} = -s_{\boldsymbol{\alpha}'} \mod p]$. We now proceed to analyze the probability $\mathcal{A}$ succeeds in breaking the single rational root assumption, given as inputs $\mathbf{G} := [\alpha_i \cdot G]_{i \in [n]}$.

Probability of $\mathcal{A}$ succeeding. Recall that $\mathcal{A}(\mathbb{G}, \mathbf{G})$ succeeds with non-negligible probability if $\mathbf{G}$ is indeed randomly sampled from $\mathbb{G}^n$. Thus, we will just lower bound the probability over the choice of $\boldsymbol{\alpha}$ that $\mathbf{G}$ seems randomly sampled.

The argument is similar to the earlier one. Let B' be an upper bound on the size of the group. If we sample α_i from $[0, B)$. The bad case is that α_i is not uniform $\mod |\mathbb{G}|$. For $t := B \mod |\mathbb{G}|$, this can only be the case if $\alpha_i \in [B-t, B)$. Note that $t < |\mathbb{G}| < B' < B$. So as long as each $\alpha_i \xleftarrow{\$} [0, B-t]$, $\mathbf{G}$ appears randomly sampled from $\mathbb{G}^n$. This happens with probability $(1 - t/B)^n \geq (1 - B'/B)^n$. We can set $B := nB'$, and use the inequality $\left(1 - \frac{1}{x}\right)^x \geq \frac{1}{4}$ for $x \geq 2$. Thus, for at least a $1/4$ fraction of choices of $\boldsymbol{\alpha}$, $\mathbf{G}$ is truly random.

Let the adversary $\mathcal{A}(\mathbb{G}, G)$ succeed with non-negligible probability p when $\mathbf{G} \xleftarrow{\$} \mathbb{G}^n$, then it succeeds with non-negligible probability at least $\frac{p}{4}$ over the randomness of $\mathcal{A}$ and the choice of $\boldsymbol{\alpha}$.

Overall probability of $\mathcal{B}$ succeeding. Recall that the probability that $\mathcal{B}$ succeeds given that $\mathcal{A}$ succeeds is lower bounded by

$$\begin{aligned}
\Pr_{\boldsymbol{\alpha} \xleftarrow{\$} [0,B)^n} [\mathcal{B} \text{ succeeds} | \mathcal{A} \text{ succeeds}] &\geq 1 - \Pr_{\boldsymbol{\alpha} \xleftarrow{\$} [0,B)^n} [\alpha_{i'} = -s_{\boldsymbol{\alpha}'} \mod p] \\
&\geq 1 - \frac{1}{p} - \frac{1}{B}
\end{aligned}$$

For $B = nB' = O(1/2^\lambda)$, this can be lower bounded by a constant c'. Thus, the overall probability of $\mathcal{B}$ succeeding is lower bounded by

$$\Pr_{\boldsymbol{\alpha} \xleftarrow{\$} [0,B)^n} [\mathcal{B} \text{ succeeds} | \mathcal{A} \text{ succeeds}] \Pr[\mathcal{A} \text{ succeeds}] \geq \frac{c'p}{4}$$

Where the latter probability is over the choice of $\boldsymbol{\alpha}$ as well as the randomness of $\mathcal{A}$. Thus, $\mathcal{A}$ succeeds with non-negligible probability for the choice $B := nB'$, where B' is the upper bound on the size of the group implicit in GGen.

□

A.2 Modular consistency security in GGM

Here, we prove the Lemma 3.5.

Proof. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be a generic adversary that plays the modular consistency game. We will show that $\mathcal{A}$ succeeds with at most negligible probability.
Since $\mathcal{A}_1$ is a $T \cdot \mathsf{rep}$-step Turing machine, the output element U of $\mathcal{A}_1$ can be represented as $\sum_{i=1}^{N} \alpha_i \cdot G_i$ with each $\alpha_i \leq 2^{T+2}$. This holds since a $T \cdot \mathsf{rep}$-step Turing machine can make at most T queries to $\mathcal{O}_2$ (as it needs at least rep steps to read the output of each query). Here, N is the number of queries made to $\mathcal{O}_1$.

WLOG, let the generators sampled in the game be $\boldsymbol{G} = (G_1, \ldots, G_n)$. Then, suppose we have that for all $i \in [T]$, $\langle \boldsymbol{r}_i, \boldsymbol{G} \rangle + \ell_i Q_i = U$. This implies that

$$\sum_{j=1}^{n} r_{i,j} \cdot G_j + \sum_{j=1}^{N} \ell_i \cdot \beta_{i,j} \cdot G_j = \sum_{j=1}^{N} \alpha_j \cdot G_j$$
$$\implies \sum_{j=1}^{n} (r_{i,j} + \ell_i \cdot \beta_{i,j}) \cdot G_j + \sum_{j=n+1}^{N} \ell_i \cdot \beta_{i,j} \cdot G_j = \sum_{j=1}^{N} \alpha_j \cdot G_j$$

Unless the event DLOG happens (which can happen with at most negligible probability), we first have that $\ell_i \cdot \beta_{i,j} = \alpha_j$ for all $i \in [T]$ and $j > n$. In particular, this implies that $\ell_i | \alpha_j$ for all $i \in [T]$ and $j > n$. However, since the ℓ_i are all coprime, this can only happen if $\alpha_j > \prod_{i \in [T]} \ell_i > 2^{T+2}$, which is a contradiction or if $\alpha_j = 0$ for all $j > n$.

Similarly, we also have with overwhelming probability that $r_{i,j} + \ell_i \cdot \beta_{i,j} = \alpha_j$ for all $i \in [T]$ and $j \in [n]$. In particular, for any j, this implies that $r_{i,j} = \alpha_j \mod \ell_i$ for all $i \in [T]$. Using the Chinese Remainder Theorem, there is a unique solution of for α_j modulo $\prod_{i \in [T]} \ell_i$. However, this product is larger than the bound on α_j – hence it must hold that $x_j = \alpha_j$ for all $j \in [n]$. This implies that $\langle \boldsymbol{x}, \boldsymbol{G} \rangle = U$ with overwhelming probability.

□

B Deferred DewTwo Proofs

In this section we present the rest of our deferred proofs.

B.1 Proof of binding for Construction 1

The binding property (see Section 2.4) of the DewTwo protocol relies on two factors: (a) the modular consistency assumption, and (b) the properties of the hint space. Here, instead of using the modular consistency assumption, we use the hidden order assumption which is implied by the adaptive root assumption [BBF19, Lemma 2]; Hence, implied by the modular consistency Lemma A.2.

Before proving the binding property of the DewTwo protocol, we state a lemma that will be used in the proof.

Lemma B.1. *Let $x \neq y$ and $x, y \in \mathcal{H} = \{\frac{a}{b} \in \mathbb{Q} : |\frac{a}{b}| \leq A, 0 < b \leq B\}$. Then we have*

$$\frac{1}{B^2} \leq |x - y| \leq 2A$$

Proof of Lemma B.1. Let $x = \frac{x_a}{x_b}$, $y = \frac{y_a}{y_b}$, and $x_a, x_b, y_a, y_b \in \mathbb{Z}$. The lower-bound is derived as follows:

$$|x - y| = \left|\frac{x_a}{x_b} - \frac{y_a}{y_b}\right| = \left|\frac{x_a y_b - y_a x_b}{x_b y_b}\right| \geq \frac{1}{x_b y_b} \geq \frac{1}{B^2}$$

The upper-bound is derived as follows:

$$|x - y| \leq \max \mathcal{H} - \min \mathcal{H} = A - (-A) = 2A$$

□

Proof. If there exists a PPT adversary $\mathcal{A}$ that breaks the binding property of CS with non-negligible probability $\epsilon(\lambda)$, i.e.

$$\Pr\left[\begin{array}{c|c} \mathsf{CS.Open}(\mathsf{pp}, \mathsf{cm}, p, \tilde{\mathbf{p}}) = 1 & \\ \wedge & \mathsf{pp} \leftarrow \mathsf{CS.Setup}(1^\lambda) \\ \mathsf{CS.Open}(\mathsf{pp}, \mathsf{cm}, p', \tilde{\mathbf{p}}') = 1 & (\mathsf{cm}, p, \mathbf{p}', \tilde{\mathbf{p}}, \tilde{\mathbf{p}}') \leftarrow \mathcal{A}(\mathsf{pp}) \\ \wedge & \\ \mathbf{p} \neq \mathbf{p}' & \end{array}\right] \geq \epsilon(\lambda)$$

CS.Open outputs 1 for both pairs of $(p, \tilde{\mathbf{p}})$ and $(p', \tilde{\mathbf{p}}')$; therefore we have:

1. Both opening hints lie in the correct hint space, i.e. $\tilde{\mathbf{p}}, \tilde{\mathbf{p}}' \in \mathcal{H}^N$ where

$$\mathcal{H} = \left\{\frac{a}{b} \in \mathbb{Q} : |\frac{a}{b}| \leq A, 0 < b \leq B\right\}$$

   where $A = Nq^{2\mu+1} + 1$ and $B = N^{8\mu} \cdot (2\mu)^\lambda$.
2. Both of the hints lead to the same commitment:

$$\left(\sum_{i=0}^{N-1} \tilde{\mathsf{p}}_i \alpha^i\right) \cdot G = \left(\sum_{i=0}^{N-1} \tilde{\mathsf{p}}'_i \alpha^i\right) \cdot G \rightarrow \left(\sum_{i=0}^{N-1} (\tilde{\mathsf{p}}_i - \tilde{\mathsf{p}}'_i)\alpha^i\right) \cdot G = 0$$

Now there are two possibilities for the exponent $\sum_{i=0}^{N-1} (\tilde{\mathsf{p}}_i - \tilde{\mathsf{p}}'_i)\alpha^i$, either it's zero or non-zero.

- Case 1 (Non-Zero exponent event): $\sum_{i=0}^{N-1} (\tilde{\mathsf{p}}_i - \tilde{\mathsf{p}}'_i)\alpha^i \neq 0$, In this case we use $\mathcal{A}$ to break the hidden order assumption (see Definition A.3). Specifically, $\mathcal{A}_{ord}$, the adversary for the hidden order assumption, takes on input $(\mathbb{G}, G, \boldsymbol{G})$, invokes $\mathcal{A}$ to get $(\mathsf{cm}, p, p', \tilde{\mathbf{p}}, \tilde{\mathbf{p}}')$, and outputs $(W = G, x = \sum_{i=0}^{N-1} (\tilde{\mathsf{p}}_i - \tilde{\mathsf{p}}'_i)\alpha^i)$.

- Case 2 (Zero exponent event): $\sum_{i=0}^{N-1}(\tilde{\mathsf{p}}_i - \tilde{\mathsf{p}}'_i)\alpha^i = 0$, we show that this case is impossible. WLOG, let j be the largest index such that $\tilde{\mathsf{p}}_j \neq \tilde{\mathsf{p}}'_j$. This implies that $\sum_{i=0}^{j}(\tilde{\mathsf{p}}_i - \tilde{\mathsf{p}}'_i)\alpha^i = 0$ or equivalently

$$(\tilde{\mathsf{p}}_j - \tilde{\mathsf{p}}'_j)\alpha^j = \sum_{i=0}^{j-1}(\tilde{\mathsf{p}}_i - \tilde{\mathsf{p}}'_i)\alpha^i$$

By taking the absolute value of both sides and replacing the upper-bound of the Lemma B.1 in the right-hand side, we get

$$\left|(\tilde{\mathsf{p}}_j - \tilde{\mathsf{p}}'_j)\alpha^j\right| = \left|\sum_{i=0}^{j-1}(\tilde{\mathsf{p}}_i - \tilde{\mathsf{p}}'_i)\alpha^i\right| \leq 2A\sum_{i=0}^{j-1}\alpha^i$$

Also, in a geometric series where $\alpha = q^{7\mu} > 2$, we have $\sum_{i=0}^{j-1}\alpha^i = \alpha^{j-1}/(\alpha-1) \leq \alpha^{j-1}$; hence, we simplify the above equation by:

$$\left|(\tilde{\mathsf{p}}_j - \tilde{\mathsf{p}}'_j)\alpha^j\right| \leq 2A\alpha^{j-1}$$

Now by using the lower-bound of the Lemma B.1 we get

$$\frac{\alpha^j}{B^2} \leq \left|(\tilde{\mathsf{p}}_j - \tilde{\mathsf{p}}'_j)\alpha^j\right| \leq 2A\alpha^{j-1}$$

Now, If we could show that $\alpha^j/B^2 > 2A\alpha^{j-1}$, or equivalently, $\alpha > 2AB^2$, we get a contradiction and we're done with the proof. Basically, this inequality is a guideline to choose the parameters of the scheme. By plugging the A and B, we get

$$\alpha \geq 2AB^2 = 2(Nq^{2\mu+1}+1)\cdot(N^{8\mu}.(2\mu)^{\lambda})^2$$

By plugging the parameters mentioned in the construction, i.e. $\alpha = q^{7\mu}$, $N^4 \leq q$, $\mu \leq \lambda/4$ and $3 \leq \lambda < \log q$, we get

$$q^{7\mu} \geq 2(q^{2\mu+1.25}+1)q^{4\mu}\cdot q^{2+\log\mu}$$

We claim that the above equality holds for $\mu \geq 4$. This is because if $\mu \geq 4$, then $q^{\mu} \geq 4\,q^{3+\log\mu}$. Hence,

$$q^{7\mu} \;=\; q^{6\mu}\,q^{\mu} \;\geq\; q^{6\mu}\,(4\,q^{3+\log\mu}) \;=\; 4\,q^{6\mu+3+\log\mu} \;\geq\; 2(q^{2\mu+1}+1)\,q^{4\mu}\,q^{2+\log\mu},$$

where the last step uses $q^{2\mu+1}+1 \;\leq\; 2\,q^{2\mu+1}$.

Therefore, by the hidden order assumption which is implied by the adaptive root assumption, and by analyzing the above two cases, we have:

$$\begin{aligned}\Pr[\mathcal{A}\text{ success}] &= \Pr[\mathcal{A}\text{ success}|\text{zero exponent}]\Pr[\text{zero exponent}]\\ &\quad+\Pr[\mathcal{A}\text{success}|\text{non-zero exponent}]\Pr[\text{non-zero exponent}]\\ &= 0+\Pr[\mathcal{A}_{ord}\text{ success}]\\ &\leq \text{negl}(\lambda)\end{aligned}$$

□

B.2 Security of Weak-DewTwo

In this section, we present the deferred proof of the DewTwo protocol presented in Section 5.2.

B.2.1 Costs of Weak-Eval

Cost of CoeffSplit (Algorithm 5.2.1). Let a and b be n-bit integers, CoeffSplit first multiplies a and b in $O(\mathsf{M}(n))$, then invokes a constant number of integer divisions of size $O(n)$ bits; Hence, the complexity of CoeffSplit is $O(\mathsf{M}(n))$.

Cost of MonomialExpand (Algorithm 5.2.1). It computes a vector of size N, where each entry is computed in at most $O(\log \mu) = O(\log \log N)$ field multiplications. A q bit field multiplication costs at most $\mathsf{M}(\log(q)) = O(\lambda \log \lambda)$. Then the total complexity of MonomialExpand is $O(N\lambda \log \lambda \log \log N)$.

Prover time. The prover time consists of (i) $O(\mu)$ field operations for Steps 1 and 2 (see Remark 5.4) or equivalently $O(\mu \log q) = O(\mu\lambda)$ bit operations (ii) $O(\log \alpha^N) = O(\lambda N \log N)$ bit operations for invoking CoeffSplit (iii) $O(\mathsf{MSM}(\log n_{\mathsf{TPoKEDex}}, n_{\mathsf{TPoKEDex}}) + \log n_{\mathsf{TPoKEDex}})$ group operations and $O(\mathsf{M}(n_{\mathsf{TPoKEDex}}) \log^2 n_{\mathsf{TPoKEDex}})$ bit operations for invoking TPoKEDex. By plugging $n_{\mathsf{TPoKEDex}} = O(\log(Nq^{\mu+1}\alpha^{N-2})) = O(\lambda N \log N)$, we get

$$\begin{aligned} O(\mathsf{MSM}(\log(\lambda N \log N), \lambda N \log N) + \log(\lambda N \log N)) &= \frac{(\lambda N \log N)\log(\lambda N \log N)}{\log\log(\lambda N \log N)} + \log(\lambda N \log N) \\ &= O(\lambda N \log^2(\lambda N)) \end{aligned}$$

group operations and

$$\begin{aligned} O(\mathsf{M}(\lambda N \log N) \log^2(\lambda N \log N)) &= \lambda N \log N \log^3(\lambda N \log N) \\ &= O(\lambda N \log^4(\lambda N)) \end{aligned}$$

bit operations.

Verifier time. The verifier time consists of (i) $O(\mu)$ field operations, or equivalently $O(\log q) = O(\lambda)$ bit operations for Steps 1 and 2 (ii) $O(\lambda + \log n_{\mathsf{TPoKEDex}})$ group operations and $O(s_{\mathsf{TPoKEDex}}\mathsf{M}(\lambda))$ bit operations for invoking TPoKEDex. By plugging $n_{\mathsf{TPoKEDex}} = O(\lambda N \log N)$ and $s_{\mathsf{TPoKEDex}} = \log N$ we get $O(\lambda + \log \lambda N)$ group operations and $O(\log N\mathsf{M}(\lambda))$ bit operations. By summing up the above, we get $O(\lambda + \log \lambda N)$ group operations, and $O(\lambda \log \lambda \log N)$ bit operations.

Proof size. The proof consists of 1 group element in Step 5 and $2\lceil \log\lceil \log 2n_{\mathsf{TPoKEDex}} + 5\rceil\rceil + 3$ group elements and $12\lambda\lceil \log\lceil \log 2n_{\mathsf{TPoKEDex}} + 5\rceil\rceil + 22\lambda$ bits. For TPoKEDex invocation in Step 6. By the following upper bound for n_{TPoKEDex}:

$$\begin{aligned} n_{\mathsf{TPoKEDex}} &\le \log(Nq^{\mu+1}\alpha^{N-2}) \\ &= \log(N) + (\mu+1)\log(q) + (N-2)\log(\alpha) \\ &< \mu + \lambda(\mu+1) + (2^\mu - 2)7\mu\lambda \end{aligned} \tag{6}$$

asymptotically, we get $n_{\mathsf{TPoKEDex}} = O(\lambda N \log N)$ which leads to a total proof size of $O(\log\log \lambda N)$ group elements and $O(\lambda \log \log \lambda N)$ bits. Concretely, using Eq. (6), for $\mu = 30$ and $\lambda = 128$, we get $n_{\mathsf{TPoKEDex}} = 2.8 \times 10^{13}$ which gives us 16 group elements, i.e. 3KB plus 1.5 KB bit elements which leads to a proof size of $4.5KB$.

B.2.2 Semi-Adaptive Knowledge Soundness of Weak-Eval

Lemma B.2 (Corollary 1 from [BF23]). *For all n such that*

$$\log n \ge 8\mu^2 + \log_2(2\mu) \cdot \lambda$$

we have that for any μ-linear polynomial f that is coprime with n (all coefficients of f are coprime with n),

$$\Pr_{\mathbf{X} \leftarrow [0,m)^\mu}[f(\mathbf{X}) \equiv 0 \mod n] \le 2^{-\lambda} + \frac{\mu}{m}$$

The above lemma implies following corollaries that we will use in our proof.

Corollary B.3 (Local zero to global zero)**.** *Let f be a μ-linear polynomial coprime with n such that $8\mu^2 + \log_2(2\mu) \cdot \lambda \leq \log n$ and for a random $\boldsymbol{y} \leftarrow [0, m)^\mu$, $f(\boldsymbol{y}) \equiv 0 \mod n$; then f is the zero polynomial mod n, or equivalently:*

$$\Pr[f(\boldsymbol{y}) = 0 \text{ for all inputs } \boldsymbol{y} \in [0, m)^\mu\ ] \geq 1 - 2^{-\lambda} - \frac{\mu}{m}$$

Corollary B.4 (Modulus upper bound)**.** *Let f be a μ-linear polynomial that is coprime with n and not identically zero. If $f(\boldsymbol{y}) \equiv 0 \mod n$ for a random point $\boldsymbol{y} \leftarrow [0, m)^\mu$, then it must hold that*

$$n < N^{8\mu} + 2\mu^\lambda.$$

We also use the following lemma in the proof:

Lemma B.5 (Lemma 1 from [BF23]: Evaluation bound)**.** *For any μ-linear polynomial f and $m \geq 2$:*

$$\Pr_{x \leftarrow [0,m)^\mu}\left[|f(x)| \leq \frac{1}{m^\mu} \cdot \|f\|_\infty\right] \leq \frac{3\mu}{m}$$

where $\|f\|_\infty$ denotes the maximum over the absolute values of all coefficients of f

Lemma B.6 (Theorem 1 from [KM03])**.** *Let x be an arbitrary number in $\mathcal{Q}_M = \{\frac{p}{q} : p, q \in \{1, \ldots, M\}\}$. Suppose we are given an oracle that takes an input y and answers query of the form "Is $x \leqslant y$?". Then we can identify the number x in $\Theta(\log M)$ time and space by making at most $2\log_2 M + O(1)$ queries to the oracle.*

Now, we prove the knowledge-soundness of the Weak-Eval protocol.

Proof. Fix an arbitrary expected polynomial time stateful adversary pair $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ for the semi-adaptive knowledge-soundness game of Weak-Eval.

Building an extractor for the reduction of knowledge. The first step of the proof is to define an extractor. By construction of Weak-Eval, using $\mathcal{A}$, we can define a pair of stateful adversaries $\mathsf{TPoKEDex}.\mathcal{A} = (\mathsf{TPoKEDex}.\mathcal{A}_1, \mathsf{TPoKEDex}.\mathcal{A}_2)$ for the adaptive soundness game of TPoKEDex. $\mathsf{TPoKEDex}.\mathcal{A}_1$ is described below:

```
TPoKEDex.$\mathcal{A}_1(\mathsf{pp})$

1. Invoke the Weak-Eval adversary $\mathcal{A}_1(\mathsf{pp})$ to get $\mathbb{x}_1 := (q, \mu, \mathsf{cm})$.
2. Invoke the public-coin challenger $\mathcal{C}(\mathsf{pp}, \mathbb{x}_1)$ to get $\mathbb{x}_2 := \mathbf{y}$.
3. Invoke the Weak-Eval adversary $\mathcal{A}_2(\mathsf{pp}, \mathbb{x}_1)$ to get $\mathbb{x}_2 := z$.
4. Run the Steps 1 to 6 of Weak-Eval to compute $\mathbb{x}_{\mathsf{TPoKEDex}}$. Output $\mathbb{x}_{\mathsf{TPoKEDex}}$.
```

We define $\mathsf{TPoKEDex}.\mathcal{A}_2$ to be the TPoKEDex sub-protocol of $\mathcal{A}_2$. Now, the adaptive knowledge-soundness of TPoKEDex implies that there exists an extractor TPoKEDex.Ext that extracts the witness $\mathbb{w}_{\mathsf{TPoKEDex}}$ from with a non-negligible probability $\epsilon_{\mathsf{TPoKEDex}}(\lambda)$. We use this extractor to construct an extractor Ext for Weak-Eval. Intuitively, the extractor Ext runs the TPoKEDex.Ext to extract the integer c used to construct the commitment cm and represents c in base α. Then, extracts a rational representation of the form $c_i = (m_i\alpha + n_i)/k_i$ for each coefficient c_i.

```
Ext$(\mathsf{pp}, \mathbb{x})$

1. Run the Steps 1 to 6 of Weak-Eval to compute $\mathbb{x}_{\mathsf{TPoKEDex}}$.
2. Invoke TPoKEDex.Ext(pp, TPoKEDex.$\mathbb{x}$) to obtain $\mathbb{w}_{\mathsf{TPoKEDex}} = (c, \boldsymbol{u})$.
3. Express $c$ in base $\alpha$ as, i.e. invoke $(c_0, \ldots, c_k) := \mathsf{repr}_\alpha(c)$. If $k > N - 1$, abort.
```

4. For each $i \in \{0, \ldots, N-1\}$, do the following:
   (a) Invoke the algorithm described in Lemma B.6 to obtain m_i/k_i which is the closest fraction to c_i/α. If $k_i > N^{8\mu} \cdot (2\mu)^\lambda$, the extractor aborts.
   (b) Compute $n_i = c_i k_i - m_i \alpha$.
   (c) Set $\tilde{\mathsf{p}}_i = \frac{n_i}{k_i} + \frac{m_{i-1}}{k_{i-1}}$ where $m_{-1} = 0$ and $k_{-1} = 1$.
5. Compute $p = \tilde{\mathbf{p}} \mod q$ and output $(p, \tilde{\mathbf{p}})$.

The remainder of the proof demonstrates that the extractor successfully extracts a valid witness $\mathbb{w}^* = (p, \tilde{\mathbf{p}})$ for the semi-adaptive instance with overwhelming probability, thereby ensuring that Section 2.5 holds.

Extraction success probability. We need to show that if the TPoKEDex extractor succeeds, with overwhelming probability, the following conditions hold:

1. Commitment Opening: $\mathsf{PCS.Open}(\mathsf{pp}, \mathsf{cm}, p, \tilde{\mathbf{p}}) = 1$:
   (a) $\tilde{\mathbf{p}} \in \mathcal{H}^N$ where $\mathcal{H} = \left\{ \frac{a}{b} \in \mathbb{Q} : 0 \leq |\frac{a}{b}| \leq (Nq^{2\mu+1}), 0 < b \leq N^{8\mu} \cdot (2\mu)^\lambda \right\}$
   (b) $C = (\sum_{i=0}^{2^\mu - 1} \tilde{\mathsf{p}}_i \alpha^i) \cdot G_1$
   (c) $p \in \mathbb{F}_q[X_1, \ldots, X_\mu]$ with a coefficient vector $\mathbf{p}$ such that $\mathbf{p} = \tilde{\mathbf{p}} \mod q$.
2. Polynomial Evaluation: $z = p(\mathbf{y}) \mod q$,

Note that Item 1c is satisfied by construction of the extractor. We now show that Items 1, 1b and 2 are satisfied with overwhelming probability.

Proving Item 1a. To prove that the extracted opening is valid, i.e., $\tilde{\mathbf{p}} \in \mathcal{H}^N$, we first show that the extracted coefficients from TPoKEDex can be written as

$$c_i = \frac{m_i \alpha + n_i}{k_i} \quad \text{for all } i \in \{0 \ldots, N-1\} \tag{7}$$

where $m_i, n_i, k_i \in \mathbb{Z}$, $m_i \geq 0, |n_i| < \alpha, k_i > 0$ and $\gcd(m_i, k_i) = \gcd(n_i, k_i) = \gcd(k_i, \alpha) = 1$. By construction, $c_i < \alpha$; Hence, we immediately get $m_i < k_i$. Moreover, note that for each c_i, there is at least one trivial instance of such a representation, which is of the form $c_i = \frac{0\alpha + c_i}{1}$ (note that $\gcd(0, 1) = 1$). We aim to show the existence of a representation for each c_i with further constraints on both the numerator and denominator. Looking ahead, these bounds will allow us to extract a valid opening for the commitment scheme, i.e. allows us to show $\tilde{\mathbf{p}} \in \mathcal{H}^N$.

The TPoKEDex extractor outputs integers c, h, s, t, u such that with overwhelming probability we have:

$$\begin{gathered} C = c \cdot G, \quad U = h \cdot G_2 + s \cdot G_3 + t \cdot G_4 + u \cdot G_5 \\ c \cdot \sigma = s + t \cdot \alpha^{N-1} + u \cdot \alpha^N \quad t = z + h \cdot q \end{gathered} \tag{8}$$

such that $0 \leq s < Nq^{\mu+1}\alpha^{N-2}$, $0 \leq t < Nq^{\mu+1}$ and $0 \leq u < q^{\mu+1}\alpha^{N-2}$. Further, we can write $s = s_0 + s_1$ and plug it into Eq. (8) to get:

$$c \cdot \sigma = s_0 + s_1 \cdot \alpha^{N-2} + t \cdot \alpha^{N-1} + u \cdot \alpha^N \tag{9}$$

where by definition $s_1 = \lfloor \frac{s}{\alpha^{N-2}} \rfloor < Nq^{\mu+1}$ and $s_0 = s \mod \alpha^{N-2}$ which implies $s_0 < \alpha^{N-2}$. The decomposition of s into s_0 and s_1 is done to show a bound on the term preceding the central term, i.e. $t \cdot \alpha^{N-1}$ using the carefully crafted check on s in the protocol. No through a series of comparisons between the LHS and RHS of Eq. (9), we derive bounds on the coefficients m_i, k_i, and n_i.

Comparing LHS and RHS of Eq. (9) in base-α. On the RHS of Eq. (9), the coefficient of α^{N-2} is s_1 and the coefficient of α^{N-1} is t, which are both bounded as shown earlier. On the LHS, by definition, the coefficient of α^{N-2} and α^{N-1}, are $\lfloor \frac{c \cdot \sigma}{\alpha^{N-2}} \rfloor \mod \alpha$ and $\lfloor \frac{c \cdot \sigma}{\alpha^{N-1}} \rfloor \mod \alpha$ respectively.

$$\left\lfloor \frac{c \cdot \sigma}{\alpha^{N-2}} \right\rfloor \mod \alpha = \left\lfloor \frac{\sum_{i=0}^{N-1} c_i \cdot \alpha^i \sum_{j=0}^{N-1} \mathsf{r}_{N-1-j} \cdot \alpha^j}{\alpha^{N-2}} \right\rfloor \mod \alpha$$

$$= \left\lfloor \frac{\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} c_i \cdot \mathsf{r}_{N-1-j} \cdot \alpha^{i+j}}{\alpha^{N-2}} \right\rfloor \mod \alpha$$

$$= \left\lfloor \frac{\sum_{0 \le i+j < N-2} c_i \cdot \mathsf{r}_{N-1-j} \cdot \alpha^{i+j}}{\alpha^{N-2}} + \sum_{i+j=N-2} c_i \cdot \mathsf{r}_{N-1-j} \right.$$

$$\left. + \alpha \cdot \left(\sum_{i+j > N-2} c_i \cdot \mathsf{r}_{N-1-j} \cdot \alpha^{i+j-N+1} \right) \right\rfloor \mod \alpha$$

$$= \left\lfloor \frac{\sum_{0 \le i+j \le N-3} c_i \cdot \mathsf{r}_{N-1-j} \cdot \alpha^{i+j}}{\alpha^{N-2}} \right\rfloor + \sum_{i=0}^{N-2} c_i \cdot \mathsf{r}_{i+1} \mod \alpha \tag{10}$$

We denote by e as the fraction inside the above equation. Equating the coefficients of α^{N-2} on both sides, we get that $\sum_{i=0}^{N-2} c_i \mathbf{r}_{i+1} = s_1 - \lfloor e \rfloor \mod \alpha$, which implies there exists $m \in \mathbb{Z}$ such that

$$\sum_{i=0}^{N-2} c_i \mathbf{r}_{i+1} = (s_1 - \lfloor e \rfloor) + m\alpha \tag{11}$$

Using any rational representation of the form specified in Eq. (7), rewrite Eq. (11) as follows:

$$\sum_{i=0}^{N-2} \left(\frac{m_i \alpha + n_i}{k_i} \right) \mathbf{r}_{i+1} = (s_1 - \lfloor e \rfloor) + m\alpha$$

or equivalently,

$$\alpha \left(\sum_{i=0}^{N-2} \frac{m_i}{k_i} \mathbf{r}_{i+1} - m \right) = s_1 - \lfloor e \rfloor - \sum_{i=0}^{N-2} \frac{n_i}{k_i} \mathbf{r}_{i+1}$$

Define $L := \mathsf{lcm}(k_0, \ldots, k_{N-2})$ for the chosen representation. Then, we have

$$(s_1 - \lfloor e \rfloor)L - \sum_{i=0}^{N-2} \frac{Ln_i}{k_i} \mathbf{r}_{i+1} = \alpha \sum_{i=0}^{N-2} \frac{Lm_i}{k_i} \mathbf{r}_{i+1} - \alpha m L \tag{12}$$

Local zero to global zero modulus α. We can rewrite Eq. (12) as the following:

$$(s_1 - \lfloor e \rfloor)L - \sum_{i=0}^{N-2} \frac{Ln_i}{k_i} \mathbf{r}_{i+1} = \alpha \left(\sum_{i=0}^{N-2} \frac{Lm_i}{k_i} \mathbf{r}_{i+1} - mL \right) = 0 \mod \alpha \tag{13}$$

$$\implies p_1(\mathbf{y}) = 0 \mod \alpha \tag{14}$$

where $p_1(\mathbf{X})$ is a polynomial with coefficients $\frac{Ln_i}{k_i}$ for all $i \in [N-2]$ and a constant term $(s_1 - \lfloor e \rfloor)L$. We aim to use Corollary B.3 for p_1, hence we need to argue that:

- **$\mathbf{p_1(X)}$ is coprime with $\boldsymbol{\alpha}$**: Since $k_i = \gcd(L, \alpha)$ for all $i \in \{0 \ldots, N-2\}$, then we have $\gcd(L, \alpha) = 1$. Hence, all the coefficients are coprime with α.
- **Modulus $\boldsymbol{\alpha}$ is lower-bounded by $\boldsymbol{N^{8\mu} \cdot (2\mu)^{\lambda}}$** We prove the following holds:

$$\alpha = q^{7\mu} > 2^{7\mu\lambda} > 2^{28\mu^2} \overset{?}{>} N^{8\mu} \cdot (2\mu)^{\lambda} = 2^{8\mu^2} \cdot 2^{\lambda} \cdot \mu^{\lambda} \tag{15}$$

which reduces to showing

$$2^{20\mu^2-\lambda} \overset{?}{>} \mu^{\lambda}.$$

Taking base-2 logarithms on both sides and rearranging gives us:

$$20\mu^2 \overset{?}{>} \lambda(1+\log_2\mu) \geq 4\mu(1+\log_2\mu)$$

Note that both sides are strictly increasing functions of μ and for $\mu = 7$, the left-hand side is 980 and the right-hand side is approximately 107. Since the left-hand side grows much faster, the inequality holds for all $\mu \geq 7$.

Now, ensured the conditions for using Corollary B.3 are met, note that p_1 evaluates to 0 on the random point $\mathbf{r}$ which is derived from the random point $\mathbf{y}$. Therefore, by using Corollary B.3 on Eq. (14), we see that except with negligible probability $2^{-\lambda} + \frac{\mu}{q}$, p_1 is identical to an all zero polynomial modulu α.

global zero Modulus α to global zero on integers. Here, we show that all the coefficients of p_1 are smaller than the modulus α; Hence, are prime to α. This, along with p_1 being identical to the zero polynomial modulus α, implies that p_1 is identical to zero on integers. The proof is as follows:

- For coefficients $\frac{Ln_i}{k_i}$, we know that for all $i \in \{0,\dots,N-2\}$: by construction we have that $\gcd(k_i,\alpha) = 1$ and hence $L = \mathsf{lcm}(k_1,\dots,k_{N-2})$. Also by definition, we have $|n_i|< \alpha$ and hence $\gcd(n_i,\alpha) = 1$. Therefore, we have $\gcd(\frac{Ln_i}{k_i},\alpha) = 1$.
- For the constant term $(s_1 - \lfloor e \rfloor)L$, we already established that $\gcd(L,\alpha) = 1$. Now, we wish to upperbound $\lfloor s_1 - e \rfloor$ to show that the constant term is smaller than α and hence coprime to α. we know that by definition $0 \leq s_1 < Nq^{\mu+1}$. To argue about e, recall that due to extractability of $\mathsf{TPoKEDex}$, with overwhelming probability, each term on the RHS of Eq. (9) is positive. Also by construction we have $0 \leq \mathsf{r}_i \leq q^{\mu}$ which implies $\sigma = \widetilde{\overline{\mathbf{r}}}(\alpha) > 0$. This means that $c \geq 0$ and we have $0 \leq c_i \leq \alpha$. Hence, we have:

$$\begin{aligned} 0 \leq \lfloor e \rfloor &= \left\lfloor \frac{\sum_{0\leq i+j \leq N-3} c_i \cdot \mathsf{r}_{N-1-j} \cdot \alpha^{i+j}}{\alpha^{N-2}} \right\rfloor \\ &\leq \left\lfloor \frac{\sum_{i=0}^{N-3}\sum_{j=0}^{N-3-i} q^{\mu}\cdot \alpha^{i+j+1}}{\alpha^{N-2}} \right\rfloor \\ &\leq \left\lfloor \frac{N^2\cdot q^{\mu}\cdot\alpha^{N-2}}{\alpha^{N-2}} \right\rfloor \\ &= N^2 \cdot q^{\mu} \end{aligned}$$

Therefore, we have $\lfloor s_1 - e \rfloor \leq \max\{e, s_1\}$. We know that $s_1 \leq Nq^{\mu+1}$ and $\lfloor e \rfloor \leq N^2 q^{\mu}$. Now, note that $q > N$; because $q > 2^{\lambda}$ and $N < 2^{\lambda/4}$. Hence, we have

$$\lfloor s_1 - e \rfloor \leq \max\{s_1\} \leq Nq^{\mu+1} \leq q^{\mu+2} \leq q^{7\mu} = \alpha$$

Therefore, we have $\gcd(\lfloor s_1 - e \rfloor, \alpha) = 1$. Hence all the coeffitions of p_1 are coprime to α and 0 modulus α. Therefore, p_1 is identical to the zero polynomial on integers.

Upper bounding L. Now p_1 being identical to the zero polynomial implies that the left and right hand sides of Eq. (13) are identically zero. Hence, we have:

$$\sum_{i=0}^{N-2} \frac{Lm_i}{k_i} \mathbf{r}_{i+1} = mL = 0 \mod L \tag{16}$$

and also

$$\sum_{i=0}^{N-2} \frac{Ln_i}{k_i}\mathbf{r}_{i+1} = (s_1 - \lfloor e \rceil)L = 0 \mod L \tag{17}$$

$$\implies p_2(\mathbf{y}) = 0 \mod L \tag{18}$$

where $p_2(\mathbf{X})$ is a polynomial with coefficients $[Ln_i/k_i]_{i=0}^{N-2}$. Since $L = \mathsf{lcm}(k_1, \ldots, k_{N-2})$, the coefficients of the p_2 are coprime to L. Hence, we can use Corollary B.4 to calculate a bound on the size of L:

$$L < N^{8\mu} \cdot (2\mu)^{\lambda} \tag{19}$$

Upper bounding k_i and m_i. Since by definition $L = \mathsf{lcm}(k_0, \ldots, k_{N-2})$, in turn implies that each $k_i < N^{8\mu} \cdot (2\mu)^{\lambda}$. Also, note that since $c_i < \alpha$ for all i, we have that $m_i \le k_i$ for all i, else $c_i \ge \alpha$.

Upper bounding n_i. To bound the numerator term n_i, we use Lemma B.5. In equation (17), we have

$$p_2(\mathbf{r}) = (s_1 - \lfloor e \rceil)L \le Nq^{\mu+1} \cdot L$$

Lemma B.5 implies that $\|f\|_{\infty} < q^{\mu} \cdot (Nq^{\mu+1}L)$. Hence, for all $i \le N-2$, we must have that

$$\left|\frac{Ln_i}{k_i}\right| \le q^{\mu} \cdot Nq^{\mu+1}L \implies |n_i| \le q^{2\mu+1} \cdot N \cdot k_i$$

Final representation for $i \le N-2$. We have now showed that for each extracted c_i where $i \le N-2$, there exists $k_i < N^{8\mu} \cdot (2\mu)^{\lambda}$, $|n_i| < q^{2\mu+1} \cdot N \cdot k_i$ and $m_i \le k_i$ such that

$$c_i = \frac{m_i\alpha + n_i}{k_i}$$

Handling case $i = N-1$. Note that so far we only obtained representations for $c_1, \ldots, c_{N-2}$ with corresponding bounds on k_i-s, m_i-s and n_i-s. For the case $i = N-1$, we put an extra constraint of $n_{N-1} = 0$. Looking ahead, this constraint is used to recover the commitment and prove Item 1b.

We go back to Eq. (9) and denote by u' coefficient of α^{2N-2}. On the RHS, due to the bound $0 \le u \le q^{\mu+1}\alpha^{N-2}$, the coefficient of α^{2N-2} is bounded by $q^{\mu+1}$, i.e. $u' < q^{\mu+1}$. On the LHS, we have the coefficient of α^{2N-2} as $\lfloor \frac{c\cdot\sigma}{\alpha^{2N-2}} \rfloor \mod \alpha$. We can simplify this as follows:

$$\begin{aligned}
\left\lfloor \frac{c \cdot \sigma}{\alpha^{2N-2}} \right\rfloor \mod \alpha &= \left\lfloor \frac{\sum_{i=0}^{N-1} c_i\alpha^i \sum_{j=0}^{N-1} \mathbf{r}_{N-1-j}\alpha^j}{\alpha^{2N-2}} \right\rfloor \mod \alpha \\
&= \left\lfloor \frac{\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} c_i\mathbf{r}_{N-1-j}\alpha^{i+j}}{\alpha^{2N-2}} \right\rfloor \mod \alpha \\
&= \left\lfloor \frac{\sum_{i+j<2N-2} c_i\mathbf{r}_{N-1-j}\alpha^{i+j}}{\alpha^{2N-2}} \right\rfloor + c_{N-1}\mathbf{r}_0 \mod \alpha
\end{aligned}$$

Let the fraction inside the above floor term be e'. Similar to the bound on the floor term from earlier, we can bound this term:

$$\lfloor e' \rfloor \le \left\lfloor \frac{\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} q^{\mu}\alpha^{i+j+1}}{\alpha^{2N-2}} \right\rfloor$$

$$\leq \left\lfloor \frac{N^2 q^\mu \alpha^{2N-2}}{\alpha^{2N-2}} \right\rfloor$$
$$= N^2 q^\mu$$

Equating the coefficients of α^{2N-2} on both sides, we get $c_{N-1}\mathsf{r}_0 = u' - \lfloor e' \rfloor \mod \alpha$. Applying an analysis similar to those applied for Eq. (11), we have:

$$\left(\frac{m_{N-1}\alpha + n_{N-1}}{k_{N-1}}\right)\mathsf{r}_0 = u' - \lfloor e' \rfloor \mod \alpha$$

Note that we have a constraint of $m_{N-1} = 0$. Also, the above modular equation implies that there exists $m \in \mathbb{Z}$ such that:

$$n_{N-1}\mathsf{r}_0 - (u' - \lfloor e' \rfloor)k_{N-1} = mk_{N-1}\alpha = 0 \mod \alpha$$

Similar to the analysis for $i < N-1$, we can show that 1. the polynomial $n_{N-1}\mathsf{r}_0 - (u' - \lfloor e' \rfloor)k_{N-1}$ is coprime to α and 2. the modulus α is big enough; Hence, by Corollary B.3 and the fact that $u' - \lfloor e' \rfloor < \alpha$, we have:

$$n_{N-1}\mathsf{r}_0 - (u' - \lfloor e' \rfloor)k_{N-1} = 0$$

which implies

$$n_{N-1}\mathsf{r}_0 = (u' - \lfloor e' \rfloor)k_{N-1} = 0 \mod k_{N-1}$$

By Corollary B.4, we have that $k_{N-1} < N^{8\mu} \cdot (2\mu)^\lambda$. Moreover, by applying Lemma B.5, we get

$$|n_i| \leq q^\mu \cdot Nq^{\mu+1} = Nq^{2\mu+1}$$

Final step: $\tilde{\mathbf{p}} \in \mathcal{H}$. For all $i \in \{0, \dots, N\}$ We defined $\tilde{\mathsf{p}}_i = \frac{n_i}{k_i} + \frac{m_{i-1}}{k_{i-1}}$. Now we want to show $\tilde{\mathsf{p}}_i \in \mathcal{H}$ where $\mathcal{H} = \left\{\frac{a}{b} \in \mathbb{Q} : 0 \leq |\frac{a}{b}| \leq (Nq^{2\mu+1}+1), 0 \leq b \leq N^{8\mu} \cdot (2\mu)^\lambda\right\}$. We define $l = \mathsf{lcm}(k_i, k_{i-1}) < L$ and we hava that

$$\tilde{\mathsf{p}}_i = \frac{n_i \cdot \frac{l}{k_i} + m_{i-1} \cdot \frac{l}{k_{i-1}}}{l}$$

We use the above form to derive our bounds on the numerator and denominator:

- Numerator: We know that $|n_i| < Nq^{2\mu+1} \cdot k_i$ and $m_{i-1} < k_{i-1}$. Hence,

$$n_i \frac{l}{k_i} + m_{i-1} \frac{l}{k_{i-1}} < (Nq^{2\mu+1}+1) \cdot l < (Nq^{2\mu+1}+1) \cdot N^{8\mu} \cdot (2\mu)^\lambda$$

- Denominator: We know that $l = \mathsf{lcm}(k_i, k_{i-1}) < L < N^{8\mu} \cdot (2\mu)^\lambda$.

By definition of $\mathcal{H}$, we have that $\tilde{\mathbf{p}} \in \mathcal{H}^N$.

Proving Item 1b. We have

$$\left(\sum_{i=0}^{N-1} \tilde{\mathsf{p}}_i \alpha^i\right) \cdot G_1 = \left(\sum_{i=0}^{N-1} \left(\frac{n_i}{k_i} + \frac{m_{i-1}}{k_{i-1}}\right)\alpha^i\right) \cdot G_1$$
$$= \left(\sum_{i=0}^{N-1} \frac{n_i}{k_i}\alpha^i\right) \cdot G_1 + \left(\sum_{i=-1}^{N-2} \frac{m_i}{k_i}\alpha^{i+1}\right) \cdot G_1$$

$$= \left(\frac{m_{-1}}{k_{-1}} + \frac{n_{N-1}}{k_{N-1}}\alpha^{N-1} + \sum_{i=0}^{N-2}\frac{m_i\alpha + n_i}{k_i}\alpha^i\right)\cdot G_1 \qquad \text{Note that } m_1 = m_{N-1} = 0$$
$$= \left(\sum_{i=0}^{N-1}\frac{m_i\alpha + n_i}{k_i}\alpha^i\right)\cdot G_1$$
$$= \mathsf{cm}$$

Proving Item 2. So far, we only showed that the opening hint has the correct format. Now, we show that the extracted polynomial p derived from the opening $\tilde{\mathbf{p}}$ has the correct evaluation, i.e. $p(\mathbf{y}) = z$. To show this, consider the coefficient of α^{N-1} on the LHS.

$$\begin{aligned}
\left\lfloor\frac{c\cdot\sigma}{\alpha^{N-1}}\right\rfloor \mod \alpha &= \left\lfloor\frac{\sum_{i=0}^{N-1} c_i\alpha^i \sum_{j=0}^{N-1}\mathbf{r}_{N-1-j}\alpha^j}{\alpha^{N-1}}\right\rfloor \mod \alpha\\
&= \left\lfloor\frac{\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} c_i\mathbf{r}_{N-1-j}\alpha^{i+j}}{\alpha^{N-1}}\right\rfloor \mod \alpha\\
&= \Bigg\lfloor\frac{\sum_{i+j<N-1} c_i\mathbf{r}_{N-1-j}\alpha^{i+j}}{\alpha^{N-1}} + \sum_{i+j=N-1} c_i\mathbf{r}_{N-1-j}\\
&\qquad + \alpha\sum_{i+j>N-1} c_i\mathbf{r}_{N-1-j}\alpha^{i+j-N+1}\Bigg\rfloor \mod \alpha\\
&= \left\lfloor\frac{\sum_{i+j<N-1} c_i\mathbf{r}_{N-1-j}\alpha^{i+j}}{\alpha^{N-1}}\right\rfloor + \sum_{i=0}^{N-1} c_i\mathbf{r}_i \mod \alpha\\
&= \left\lfloor\frac{\sum_{i+j<N-2} c_i\mathbf{r}_{N-1-j}\alpha^{i+j}}{\alpha^{N-1}} + \frac{\sum_{i=0}^{N-2} c_i\mathbf{r}_{i+1}}{\alpha}\right\rfloor + \sum_{i=0}^{N-1} c_i\mathbf{r}_i \mod \alpha\\
&= \left\lfloor\frac{e + s_1 - \lfloor e\rfloor}{\alpha}\right\rfloor + m + \sum_{i=0}^{N-1} c_i\mathbf{r}_i \mod \alpha\\
&= \sum_{i=0}^{N-1} c_i\mathbf{r}_i + m \mod \alpha\\
&= \sum_{i=0}^{N-1}\left(\frac{m_i\alpha + n_i}{k_i}\right)\mathbf{r}_i + \sum_{i=0}^{N-2}\frac{m_i}{k_i}\mathbf{r}_{i+1} \mod \alpha\\
&= \sum_{i=0}^{N-1}\left(\frac{n_i}{k_i} + \frac{m_{i-1}}{k_{i-1}}\right)\mathbf{r}_i \mod \alpha\\
&= \sum_{i=0}^{N-1}\tilde{\mathsf{p}}_i\mathbf{r}_i \mod \alpha
\end{aligned}$$

Therefore, we get $\sum_{i=0}^{N-1}\tilde{\mathsf{p}}_i\mathbf{r}_i = t \mod \alpha$ where the last few equalities follow from equations (11) and (16), defining $m_{-1} = 0, k_{-1} = 1$. From the check made in TPoKEDex, we also know that with overwhelming probability $t = z \mod q$. Since $\alpha = 0 \mod q$, we can equate the LHS and RHS to get

$$\tilde{\mathbf{p}}(\mathbf{y}) = t \mod \alpha \implies p(\mathbf{y}) = z \mod q$$

B.3 Security of DewTwo

Here, we prove that Eval is an argument of knowledge for the relation Definition 2.1 with adaptive knowledge soundness. We prove that for all $N \in \mathbb{N}$ and all expected poly-time stateful adversaries $\mathcal{A} := (\mathcal{A}_1, \mathcal{A}_2)$ there exists an expected poly-time extractor Ext such that $\mathsf{Adv}^{\mathsf{Adap\text{-}KS}}_{\mathsf{ARG},\mathsf{Ext},\mathcal{A}}(\lambda) :=$

$$\Pr\left[\begin{array}{c} \langle \mathcal{A}_2(\mathsf{pp}), \mathcal{V}(\mathsf{pp}, \mathbb{x})\rangle = 1 \\ \wedge \\ (\mathbb{x}, \mathbb{w}^*) \notin \mathscr{R}^{\mathsf{pp}}_{\mathsf{Eval}} \end{array} \middle| \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, N) \\ (q, \mu, \mathsf{cm}, \mathbf{y}, z) \leftarrow \mathcal{A}_1(\mathsf{pp}) \\ \hline \mathbb{x} := (q, \mu, \mathsf{cm}, \mathbf{y}, z) \\ \mathbb{w}^* \leftarrow \mathsf{Ext}^{\mathcal{A}}(\mathsf{pp}, \mathbb{x}) \end{array}\right] = \mathrm{negl}(\lambda)$$

Note that Weak-Eval is invoked twice in Eval; Hence, for any pair of expected poly-time stateful adversaries $\mathcal{A} := (\mathcal{A}_1, \mathcal{A}_2)$ we can define two tuples of expected poly-time stateful adversaries Weak-Eval.$\mathcal{A}$:= (Weak-Eval.$\mathcal{A}_1$, Weak-Eval.$\mathcal{A}_2$, Weak-Eval.$\mathcal{A}_3$). Moreover, the semi-adaptive knowledge-soundness of Weak-Eval implies the existance of an expected poly-time extractor Weak-Eval.Ext that fails with a negligible probability, which we denote by $\epsilon_{\text{Weak-Eval}}(\lambda)$. We define the extractor Ext to invoke Weak-Eval.Ext and output the extracted witness. The rest of the proof shows that Ext is a valid extractor for Eval.

To show that Ext extracts a valid witness, again we need to show that Items 1 and 2 hold. Due to the knowledge-soundness of Weak-Eval.Ext, we know that *Item* 1 holds with overwhelming probability. We only show that Item 2 holds.

Note that the second call is not made with a random $\mathbf{y}$, but with $\mathbf{y}$ in the instance. Let $\mathbf{y}' \in \mathbb{Z}^N$ be the exponential expansion of $\mathbf{y}$ as a monomial product: For $i \in [N]$,

$$\mathbf{y}'_i = \prod_{j=0}^{\mu-1} \mathbf{y}_j^{i_j}$$

and $\sigma = \sum_{i=0}^{N-1} \mathbf{y}'_{N-1-i} \alpha^i$. Also, due to the extractability of $\mathsf{TPoKEDex}$, similar to Eq. (9), we also have the equation

$$c \cdot \sigma = s_0 + s_1 \cdot \alpha^{N-2} + t \cdot \alpha^{N-1} + u \cdot \alpha^N \tag{20}$$

Additionally, we know from the knowledge-soundness proof for Argument 1 that each c_i is of the form stated in Eq. (7). Now, consider the coefficient of α^{N-2} on both sides of Eq. (20). On the right hans side, we have s_1. On the left hand side, we have

$$\begin{aligned} \left\lfloor \frac{c \cdot \sigma}{\alpha^{N-2}} \right\rfloor &= \left\lfloor \frac{\sum_{i=0}^{N-1} c_i \alpha^i \sum_{j=0}^{N-1} \mathbf{y}'_{N-1-j} \alpha^j}{\alpha^{N-2}} \right\rfloor \mod \alpha \\ &= \left\lfloor \frac{\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} c_i \mathbf{y}'_{N-1-j} \alpha^{i+j}}{\alpha^{N-2}} \right\rfloor \mod \alpha \\ &= \left\lfloor \frac{\sum_{i+j<N-2} c_i \mathbf{y}'_{N-1-j} \alpha^{i+j}}{\alpha^{N-2}} \right\rfloor + \sum_{i=0}^{N-2} c_i \mathbf{y}'_{i+1} \mod \alpha \end{aligned}$$

Equating both sides, we get

$$\sum_{i=0}^{N-2} c_i \mathbf{y}'_{i+1} = s_1 - \lfloor e \rfloor \mod \alpha$$

This implies that there exists $m \in \mathbb{Z}$ such that

$$\sum_{i=0}^{N-2} \frac{m_i \alpha + n_i}{k_i} \mathbf{y}'_{i+1} = (s_1 - \lfloor e \rfloor) + m\alpha \implies \alpha \left(\sum_{i=0}^{N-2} \frac{m_i}{k_i} \mathbf{y}'_{i+1} - m \right) = (s_1 - \lfloor e \rfloor) - \sum_{i=0}^{N-2} \frac{n_i}{k_i} \mathbf{y}'_{i+1}$$

$$\implies \sum_{i=0}^{N-2} \frac{m_i}{k_i}\mathbf{y}'_{i+1} = m \in \mathbb{Z}$$

Now, consider the coefficient of α^{N-1} on the LHS.

$$\begin{aligned}
\left\lfloor \frac{c \cdot \sigma}{\alpha^{N-1}} \right\rfloor &= \left\lfloor \frac{\sum_{i=0}^{N-1} c_i \alpha^i \sum_{j=0}^{N-1} \mathbf{y}'_{N-1-j}\alpha^j}{\alpha^{N-1}} \right\rfloor \mod \alpha \\
&= \left\lfloor \frac{\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} c_i \mathbf{y}'_{N-1-j}\alpha^{i+j}}{\alpha^{N-1}} \right\rfloor \mod \alpha \\
&= \left\lfloor \frac{\sum_{i+j<N-1} c_i \mathbf{y}'_{N-1-j}\alpha^{i+j}}{\alpha^{N-1}} \right\rfloor + \sum_{i=0}^{N-1} c_i \mathbf{y}'_i \mod \alpha \\
&= \left\lfloor \frac{\sum_{i+j<N-2} c_i \mathbf{y}'_{N-1-j}\alpha^{i+j}}{\alpha^{N-1}} + \frac{\sum_{i=0}^{N-2} c_i \mathbf{y}'_{i+1}}{\alpha} \right\rfloor + \sum_{i=0}^{N-1} c_i \mathbf{y}'_i \mod \alpha \\
&= \left\lfloor \frac{e + s_1 - \lfloor e \rfloor}{\alpha} \right\rfloor + m + \sum_{i=0}^{N-1} c_i \mathbf{y}'_i \mod \alpha \\
&= \sum_{i=0}^{N-1} \left(\frac{n_i}{k_i} + \frac{m_{i-1}}{k_{i-1}} \right) \mathbf{y}'_i \mod \alpha \\
&= \sum_{i=0}^{N-1} \tilde{\mathbf{p}}_i \mathbf{y}'_i \mod \alpha
\end{aligned}$$

Hence, we have that $\tilde{\mathbf{p}}(\mathbf{y}) = t \mod \alpha$. Since $\alpha = 0 \mod q$ and $z = t \mod q$, we also have that $p(\mathbf{y}) = z \mod q$, completing the proof.

□

C Newton-Raphson

The Newton-Raphson method is a numerical root-finding algorithm, meaning that given a univariate function $f(x)$ and an initial guess x_0, it iteratively refines the guess to find x^* such that $f(x^*) = 0$. A single iteration of the Newton-Raphson method is given by the following update rule:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

The following theorem guarantees the convergence of the Newton-Raphson method under certain conditions.

Theorem C.1 ([KC09], Chapter 3)**.** *Let f be an increasing and convex function on real numbers with continues f'' everywhere. If f has a zero, then the zero is unique and the Newton-Raphson method converges to it from* any *initial guess x_0.*

We confine our discussion on the convergence and complexity of the algorithm to the special case of finding the *square* root of a number a.

C.1 Finding square roots

To find the square root of a number a, we use the Newton-Raphson method to find the root of the following function:

$$f(x) := \begin{cases} x^2 - a & \text{if } x \neq 0 \\ 0 & \text{o.w.} \end{cases}$$

The update rule for the case $x > 0$ is:

$$f(x) = x^2 - a \implies x_{n+1} = x_n - \frac{x_n^2 - a}{2x_n} = \frac{1}{2}\left(x_n + \frac{a}{x_n}\right) \tag{21}$$

by the definition of $f(x)$ and the Theorem C.1 we have the following lemma.

Lemma C.2 ([BF10; KC09])**.** *The recursion Eq. (21) converges to $r = \sqrt{a}$ for any error tolerance $\delta > 0$ and any initial guess x_0. Moreover, the convergence is quadratic, i.e., for some constant $c > 0$:*

$$|x_{n+1} - r| \leq c\,(x_n - r)^2 \quad (n \geq 0)$$

which leads to the following algorithm which is very ancient and is credited to Heron, who lived sometime between 100 B.C. and 100 A.D. [KC09].

```
SquareRoot$(a, \delta) \to b$
1. Initialize $x_{\text{old}} := 1$ and $x_{\text{new}} := \frac{x_{\text{old}} + \frac{a}{x_{\text{old}}}}{2}$.
2. While $|x_{\text{new}} - x_{\text{old}}| > \delta$:
    (a) Save the previous answer $x_{\text{old}} := x_{\text{new}}$.
    (b) Update the answer $x_{\text{new}} := \frac{x_{\text{old}} + \frac{x}{x_{\text{old}}}}{2}$.
3. Output $b = x_{\text{new}}$.
```

Corollary C.3. *The* SquareRoot *algorithm computes the square root of a number a with an error tolerance δ in $O(\log\log(a/\delta))$ iterations.*

Proof. Due to lemma C.2, the algorithm converges quadratically, which implies that the number of bits discovered at each iteration is doubled. Therefore, the algorithm incurs logarithmic cost in the number of desired bits, $\log(a/\delta)$, which gives us $O(\log\log a/\delta)$. $\square$

Now, we state the complexity of the $\mathsf{SquareRoot}$ algorithm with $\delta = 0.1$, tailored for our application in Section 6.

Corollary C.4. *The* $\mathsf{SquareRoot}$ *algorithm computes the square root of an n-bit number a with an error tolerance $\delta = 0.1$ in $O(\log n \mathsf{M}(n))$.*

Proof. By Corollary C.3, the algorithm requires $O(\log n)$ iterations. In each iteration, the algorithm complexity is dominated by the cost of the division operation, which is $O(\mathsf{M}(n))$ as shown in [GG03, Theorem 9.8]. $\square$

D Deferred toolbox proofs

D.1 PoKE deferred proofs

Here, we present the deferred knowledge soundness proof of Theorem 7.2 under the modular consistency assumption (see Definition 3.4): Assume for contradiction that there exist polynomial time adversaries $\mathsf{PoKE}.\mathcal{A} := (\mathsf{PoKE}.\mathcal{A}_1, \mathsf{PoKE}.\mathcal{A}_2)$ such that for all polynomial-time extractors Ext^* we have (see Section 2.3):

$$\mathsf{Adv}^{\mathsf{Adap\text{-}KS}}_{\mathsf{PoKE},\mathsf{Ext}^*,\mathsf{PoKE}.\mathcal{A}} > \epsilon(\lambda)$$

We show that this implies a contradiction to the MCA assumption. In particular, we build PoKE.Ext, an extractor for PoKE, and use it to build an adversary for MCA. We then show that the advantage of the MCA adversary is equal to the advantage of PoKE.$\mathcal{A}$ which is non-negligible.

PoKE Extractor. We first construct PoKE.TrExt that given the desired size T, constructs a list of accepting transcripts of size T by rewinding the adversary $\mathsf{PoKE}.\mathcal{A}_2$ as many times as needed:

```
$\mathsf{PoKE.TrExt}(\mathsf{pp}, \mathbb{x}, T) \to \mathbf{tr}$

1. Set $\mathbf{Tr} := \emptyset$.
2. While $|\mathbf{Tr}| < \mathbf{T}$:
   (a) Invoke the verifier's next-message function
           $(\mathsf{st}, \ell_i) := \mathcal{V}(\emptyset, \mathsf{vk}, \mathbb{x})$
       to get a fresh random challenge.
   (b) Invoke the adversary's next-message function on the fresh challenge
           $(Q_i, \boldsymbol{r}_i) := \mathsf{PoKE}.\mathcal{A}_2(\emptyset, \mathsf{state}, \ell_i)$
       where $\boldsymbol{r} := (r_{i,1}, \ldots, r_{i,m}) \in \mathbb{Z}^m$ and $Q_i \in \mathbb{G}$ are the responses.
   (c) If the verifier accepts, i.e.
           $\mathcal{V}(\mathsf{st}, \mathsf{vk}, \mathbb{x}, (Q_i, \boldsymbol{r}_i)) = 1$
       then record the accepting partial transcript $\mathbf{Tr} \leftarrow \mathbf{Tr} \cup \{(\ell_\mathbf{i}, \mathbf{r_i})\}$.
3. Output the set of accepting transcripts $\mathbf{Tr}$.
```

We now construct PoKE.Ext which invokes PoKE.TrExt to obtain T accepting transcripts and uses the Chinese remainder theorem to extract the witness:

```
$\mathsf{PoKE.Ext}(\mathsf{pp}, \mathbb{x}, \mathsf{state}) \to \mathbb{w}^*$

1. Let $\mathsf{PoKE}.\mathcal{A}_1$ be a $T$-step Turing machine.
2. Invoke the accepting transcript extractor:
       $\mathbf{Tr} \leftarrow \mathsf{PoKE.TrExt}(\mathsf{pp}, \mathbb{x}, \mathbf{T})$
3. For $j \in [m]$, Invoke the CRT algorithm given by the Chinese remainder
   theorem (see Theorem 3.3) on the partial transcripts
       $u_j := \mathsf{CRT}([r_{i,j}]_{i=1}^T, [\ell_i]_{i=1}^T)$
```

```
    Then output the unique CRT solution $\boldsymbol{u} = (u_1, \ldots, u_m)$
```

Adversary runtime. Note that the CRT algorithm is efficient; hence, the $\mathsf{PoKE.Ext}$ runtime is proportional to the $\mathsf{PoKE.TrExt}$ runtime. Since the $\mathsf{PoKE}.\mathcal{A}$'s success probability is $\epsilon(\lambda)$, then the average runtime of $\mathsf{PoKE.TrExt}$ is polynomial $T\epsilon(\lambda)$. Since T, the running time of $\mathsf{PoKE}.\mathcal{A}_1$, is polynomial time, the $\mathsf{PoKE.Ext}$ is efficient.

Reduction to MCA. We show how to build a pair of adversaries $\mathsf{MC}.\mathcal{A} = (\mathsf{MC}.\mathcal{A}_1, \mathsf{MC}.\mathcal{A}_2)$ for the MCA game using the pair of adversaries $\mathsf{PoKE}.\mathcal{A} = (\mathsf{PoKE}.\mathcal{A}_1, \mathsf{PoKE}.\mathcal{A}_2)$. We then show that the advantage of $\mathsf{MC}.\mathcal{A}$ is greater than the advantage of $\mathsf{PoKE}.\mathcal{A}$.

The adversary $\mathsf{MC}.\mathcal{A}_1$ is defined as follows. We denote the runtime of $\mathsf{MC}.\mathcal{A}_1$ by T.

```
$\mathsf{MC}.\mathcal{A}_1(\mathbb{G}, \boldsymbol{G}) \to (U, \mathsf{state})$

1. Set $\mathsf{pp} := (N, \mathbb{G}, \mathbf{G})$ where $N := |\boldsymbol{G}|$.
2. Invoke PoKE adversary
       $(\mathbb{x}, \mathsf{PoKE.state}) := \mathcal{A}_1(\mathsf{pp})$
3. Set $U := \mathbb{x}$ and $\mathsf{state} := (\mathsf{PoKE.state}, \mathbb{x})$, and output $(U, \mathsf{state})$.
```

```
$\mathsf{MC}.\mathcal{A}_2(\mathsf{state}) \to [\boldsymbol{r}_i, Q_i, \ell_i]_{i=1}^T$

1. Parse $\mathsf{state} := (\mathsf{PoKE.state}, \mathbb{x})$
2. Invoke the accepting transcript extractor:
       $\mathbf{Tr} \leftarrow \mathsf{PoKE.TrExt}(\mathsf{pp}, \mathbb{x}, \mathbf{T})$
3. Parse $\mathbf{Tr} := [\mathbf{r_i}, \mathbf{Q_i}, \ell_\mathbf{i}]_{\mathbf{i=1}}^{\mathbf{T}}$ and output $([Q_i]_{i=1}^T, [\boldsymbol{r}_i]_{i=1}^T)$
```

Advantage Analysis. Let $T = \mathrm{poly}(\lambda)$ be the runtime of $\mathsf{PoKE}.\mathcal{A}_1$. Let $T' = O(T) = \mathrm{poly}(\lambda)$ be the runtime for $\mathsf{MC}.\mathcal{A}_1$. We show that the advantage of $\mathsf{MC}.\mathcal{A} = (\mathsf{MC}.\mathcal{A}_1, \mathsf{MC}.\mathcal{A}_2)$ where $\mathsf{MC}.\mathcal{A}_1$ is a T'-step (or equivalently T''-rep) Turing machine is non-negligible. We have $\mathsf{Adv}^{\mathsf{MC}}_{\mathsf{GGen},T'',\mathsf{MC}.\mathcal{A}_1,\mathsf{MC}.\mathcal{A}_2,n}(\lambda) =$

$$\Pr\left[\begin{array}{c} \forall i \in [T] \;:\; \langle \boldsymbol{r}_i, \boldsymbol{G}\rangle + \ell_i Q_i = U \\ \text{s.t. } \boldsymbol{G} := (G_1, \ldots, G_n) \in \mathbb{G} \\ \wedge \\ \langle \boldsymbol{x}, \boldsymbol{G}\rangle \neq U \end{array} \;\middle|\; \begin{array}{l} \mathbb{G} \leftarrow \mathsf{GGen}(1^\lambda) \\ (U, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathbb{G}) \\ [\boldsymbol{r}_i, Q_i, \ell_i]_{i=1}^T \leftarrow \mathcal{A}_2(\mathsf{st}) \text{ s.t. } \boldsymbol{r}_i = [r_{i,j}]_{j\in[n]} \\ \ell_1, \ldots, \ell_T \text{ are co-prime integers} \\ \forall j \in [n] : x_j \leftarrow \mathsf{CRT}([r_{i,j}]_{i\in[T]}, [\ell_i]_{i=1}^T) \end{array}\right]$$

By construction and using the fact that the CRT solution is unique, $\boldsymbol{x} \in \mathbb{Z}^n$ is also the output of $\mathsf{PoKE.Ext}$ rewinding $\mathsf{PoKE}.\mathcal{A}$. Hence, we have $\mathsf{Adv}^{\mathsf{MC}}_{\mathsf{GGen},T',\mathsf{MC}.\mathcal{A}_1,\mathsf{MC}.\mathcal{A}_2,n}(\lambda) =$

$$\Pr\left[\begin{array}{c} \langle \mathcal{A}_2(\mathsf{pp}, \mathsf{state}), \mathcal{V}(\mathsf{pp}, \mathbb{x})\rangle = 1 \\ \wedge \\ (\mathbb{x}, \mathbb{w}^*) \notin \mathscr{R}^{\mathsf{pp}} \end{array} \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, N) \\ (\mathbb{x}, \mathsf{state}) \leftarrow \mathcal{A}_1(\mathsf{pp}) \\ \mathbb{w}^* \leftarrow \mathsf{Ext}^{\mathcal{A}}(\mathsf{pp}, \mathbb{x}, \mathsf{state}) \end{array}\right] \geq \epsilon(\lambda)$$

This concludes the proof.

D.2 PoKEMath deferred proofs

Here, we present the deferred proof of Theorem 7.4. Note that here we use (a) the multi-rational root assumption (see Definition A.7) which is implied by the strong RSA assumption (see Lemma A.8), (b) the adaptive root (AR) assumption (see Definition A.1) which is implied by the modular consistency assumption (see Definition 3.4 and Lemma A.2) and, (c) the knowledge-soundness of PoKE (see Theorem 7.2) which is also implied by the modular consistency assumption (see Theorem 7.2).

Before proceeding with the proof, we first present a lemma stating that any remainder vector $\boldsymbol{r}$ from an accepting transcript of PoKE is equal to the witness modulo the corresponding challenge. This is straightforward to prove in the Generic Group Model (GGM); however, without GGM, the proof relies on the single-rational root assumption.

Lemma D.1. *Fix $N \in \mathbb{N}$, an expected poly-time stateful adversary $\mathcal{A} := (\mathcal{A}_1, \mathcal{A}_2)$ for the knowledge-soundness game of* PoKE*, and the corresponding expected poly-time extractor* PoKE.Ext*. Assuming that mult-rational (see Definition A.7) root assumption holds, we have*

$$\Pr\left[\begin{array}{c|l} \langle \mathsf{PoKE}.\mathcal{A}_2(\mathsf{pp}), \mathcal{V}(\mathsf{pp}, \mathbb{x})\rangle \textit{ accepts on transcript } (\ell, Q, \boldsymbol{r}) & \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, N) \\ \wedge & \mathbb{x} \leftarrow \mathsf{PoKE}.\mathcal{A}_1(\mathsf{pp}) \\ \boldsymbol{r} \neq \boldsymbol{u}^* \mod \ell & \boldsymbol{u}^* \leftarrow \mathsf{PoKE.Ext}(\mathsf{pp}, \mathbb{x}) \end{array}\right] = \mathrm{negl}(\lambda)$$

Proof. Since PoKE is knowledge-sound, with overwhelming probability, we have

$$u_1^* \cdot G_1 + \cdots + u_m^* \cdot G_m = U$$

On the other hand, since $(\ell, U, \boldsymbol{r})$ is an accepting transcript, we have

$$U = \ell Q + (r_1 G_1 + \cdots + r_m G_m)$$

which implies

$$u_1^* \cdot G_1 + \cdots + u_m^* \cdot G_m = \ell Q + (r_1 G_1 + \cdots + r_m G_m)$$

Now, assume for contradiction and WLOG that $i \in [m]$ is the smallest index, such that for all $j > i$, $u_i^* = r_i \mod \ell$, Hence, for some $q_1, \ldots q_m \in \mathbb{Z}$ and $0 < s_1, \ldots, s_i < \ell$ we will have

$$\forall j \in [i] : u_j^* = r_j + q_j \ell + s_i$$
$$\forall j \in \{i+1, \ldots, m\} : u_i^* = r_i + q_i \ell$$

As a result, we have

$$\begin{aligned} ((r_1 + q_1\ell + s_1)G_1 + \cdots + (r_i + q_i\ell + s_i)G_i) + ((r_{i+1} + q_{i+1}\ell)G_{i+1} + \cdots + (r_m + q_m\ell)G_m) \\ = \ell Q + (r_1 G_1 + \cdots + r_m G_m) \end{aligned}$$

canceling out the $r_i G_i$ terms, we get

$$((q_1\ell + s_1)G_1 + \cdots + (q_i\ell + s_i)G_i) + (q_{i+1}\ell G_{i+1} + \cdots + q_m\ell G_m) = \ell Q$$

which implies

$$s_1 G_1 + \cdots + s_i G_i = \ell(Q - q_1 G_1 - \cdots - q_m G_m)$$

Now, this gives us a strategy to build an adversary for the multi-rational root assumption (see Definition A.7).

```
$\mathsf{MRR}.\mathcal{A}(\mathbb{G}, \boldsymbol{G}) \to (U_{\mathsf{MRR}}, m_{\mathsf{MRR}}, \boldsymbol{r}_{\mathsf{MRR}})$

1. Invoke the setup algorithm $\mathsf{pp} := \mathsf{Setup}(1^\lambda, N)$.
2. Invoke the PoKE adversary $\mathbb{x} := \mathsf{PoKE}.\mathcal{A}_1(\mathsf{pp})$.
3. Invoke the PoKE extractor $\boldsymbol{u}^* := \mathsf{PoKE.Ext}(\mathsf{pp}, \mathbb{x})$
4. Invoke the interactive argument $\langle \mathsf{PoKE}.\mathcal{A}_2(\mathsf{pp}), \mathcal{V}(\mathsf{pp}, \mathbb{x})\rangle$ and acquire an accepting transcript $(\ell, Q, \boldsymbol{r})$.
5. Parse the extracted witness components of the form
       $\forall j \in [i] : u_j^* = r_j + q_j\ell + s_i$
       $\forall j \in \{i+1, \dots, m\} : u_i^* = r_i + q_i\ell$
6. Output the following:
       $U_{\mathsf{MRR}} := Q - q_1 G_1 - \cdots - q_m G_m$
       $m_{\mathsf{MRR}} := \ell$
       $\boldsymbol{r}_{\mathsf{MRR}} := (s_1, \dots, s_i)$
```

By the above analysis and the fact that $\ell \in \mathsf{Primes}[\lambda, 2\lambda]$ and $0 < s_i < \ell$, then we have $\ell \neq 0$ and $\gcd(\ell, s_i) = 1$ which means that the adversary $\mathsf{MRR}.\mathcal{A}$ wins the multi-rational root assumption game with non-negligible probability.

□

Now we proceed with the knowledge-soundness proof of PoKEMath.

Proof. Assume for contradiction that there exist $N \in \mathbb{N}$, a non-negligible function $\epsilon(\lambda)$, and polynomial time adversaries $\mathsf{PoKEMath}.\mathcal{A} := (\mathsf{PoKEMath}.\mathcal{A}_1, \mathsf{PoKEMath}.\mathcal{A}_2)$ such that for all polynomial-time extractors Ext^* we have

$$\mathsf{Adv}^{\mathsf{Adap\text{-}KS}}_{\mathsf{PoKEMath},\mathsf{Ext}^*,\mathsf{PoKEMath}.\mathcal{A}} = \Pr[\mathsf{Adap\text{-}KS} \wedge U \neq \langle \boldsymbol{u}, \boldsymbol{G}\rangle] + \Pr[\mathsf{Adap\text{-}KS} \wedge f(u) \neq 0] > \epsilon(\lambda)$$

PoKEMath Extractor. Since PoKE is a sub-protocol of PoKEMath, we can define the PoKE adversary pair $\mathsf{PoKE}.\mathcal{A} = (\mathsf{PoKE}.\mathcal{A}_1, \mathsf{PoKE}.\mathcal{A}_2)$ by using $\mathsf{PoKEMath}.\mathcal{A}$. Moreover, Due to the knowledge-soundness of PoKE, There exists PoKE.Ext; Hence, we define PoKEMath.Ext to invoke PoKE.Ext and output the extracted witness.

By the construction of the extractor, with overwhelming probability, we have $\langle \boldsymbol{u}, \boldsymbol{G}\rangle = U$; hence,

$$\Pr[\mathsf{Adap\text{-}KS} \wedge f(u) \neq 0] = \mathrm{negl}(\lambda) \implies \Pr[\mathsf{Adap\text{-}KS} \wedge f(u) \neq 0] = \epsilon(\lambda) - \mathrm{negl}(\lambda)$$

meaning that $\Pr[\mathsf{Adap\text{-}KS} \wedge f(u) \neq 0]$ is non-negligible.

AR adversaries. By using the given PoKEMath adversary $\mathsf{PoKEMath}.\mathcal{A}$ and the constructed PoKEMath.Ext, we build AR adversaries $(\mathsf{AR}.\mathcal{A}_1, \mathsf{AR}.\mathcal{A}_2)$ as follows:

```
$\mathsf{AR}.\mathcal{A}_1(\mathbb{G}) \to Y$

1. invoke the setup algorithm $\mathsf{pp} := \mathsf{Setup}(1^\lambda, N)$.
2. invoke the PoKEMath knowledge-soundness adversary $\mathbb{x}_{\mathsf{PoKEMath}} := \mathsf{PoKEMath}.\mathcal{A}_1(\mathsf{pp})$.
3. Extract the underlying PoKEMath witness $\boldsymbol{u}^* := \mathsf{PoKEMath.Ext}(\mathsf{pp}, \mathbb{x}_{\mathsf{PoKEMath}})$.
4. Compute the first move for adaptive root $Y := f(\boldsymbol{u}^*)G$
5. Pass $\boldsymbol{u}^*$ as the state to the other adaptive root adversary.
```

```
AR.$\mathcal{A}_2(\ell) \to X$
1. Obtains $\boldsymbol{u}^*$ as the state.
2. Invoke the interaction $\langle \mathsf{PoKEMath}.\mathcal{A}_2(\mathsf{pp}), \mathsf{PoKEMath}.\mathcal{V}(\mathsf{pp}, \mathbb{x}_1)\rangle$ to acquire an accepting transcript $(Q, \boldsymbol{r})$.
3. Compute the integer division $k := (f(\boldsymbol{u}^*) - f(\boldsymbol{r}))/\ell$
4. Compute the integer division $k' := f(\boldsymbol{r}/\ell)$.
5. Output the root $X := (k + k')G$.
```

Since PoKE is knowledge-sound, by Lemma D.1, we have that $\boldsymbol{u}^* = \boldsymbol{r} \mod \ell$. Hence, we have

$$f(\boldsymbol{u}^*) = f(\boldsymbol{r}) \mod \ell \implies f(\boldsymbol{u}^*) = f(\boldsymbol{r}) + k\ell \text{ for some } k \in \mathbb{Z}$$

On the other hand, since $\mathsf{PoKEMath}.\mathcal{V}$ accepts, we have

$$f(\boldsymbol{r}) = 0 \mod \ell \implies f(\boldsymbol{r}) = k'\ell \text{ for some } k' \in \mathbb{Z}$$

Hence, we have $f(u^*) = (k + k')\ell$. Also, if $f(u^*) \neq 0$, then $Y \neq 0$. Hence the above implies that AR.$\mathcal{A}$ wins the AR game with non-negligible probability. This gives us a contradiction to the adaptive root assumption, and hence the proof is complete. □

D.3 PoKEDEx deferred proofs

In this section we present the deferred proof of Theorem 7.6; namely, we discuss the cost of the PoKEDEx protocol, and prove its knowledge-soundness.

D.3.1 Discussion of cost

Prover Efficiency. The prover complexity of PoKEDEx consists of (i) invoking t instances of IntegerSquareDecompose, which requires a total of $t\mathsf{M}(n_{\mathsf{isd}}) \log^2 n_{\mathsf{isd}}$ bit operations, plugging $n_{\mathsf{isd}} = 2n$ we get $t\mathsf{M}(2n) \log^2(2n)$ (ii) computing a pedersen commitment of size $|\boldsymbol{v}| = t \log(2n)$, which requires $O(\mathsf{MSM}(t \log(2n), n))$ group operations. (iii) invoking PoKEMath which incurs $\mathsf{MSM}(m_{\mathsf{PoKEMath}}, n)$ group operations and $O(m_{\mathsf{PoKEMath}} \cdot \mathsf{M}(n_{\mathsf{PoKEMath}}))$ bit operations, plugging $m_{\mathsf{PoKEMath}} = m + t \log(2n)$ and $n_{\mathsf{PoKEMath}} = n$, we get $O(\mathsf{MSM}(m + t \log n, n))$ group operations and $O(\mathsf{M}(n)(m + t \log(2n)))$ bit operations.

By summing up the above, we get a total of $O(\mathsf{MSM}(m + t \log n, n))$ group operations and $O(\mathsf{M}(n)(m + t \log n))$ bit operations.

Verifier Efficiency. The verifier complexity of PoKEDEx is dominated by invoking PoKEMath which incurs $\mathsf{MSM}(m_{\mathsf{PoKEMath}}, \lambda)$ group operations and $O(s_{\mathsf{PoKEMath}}\mathsf{M}(\lambda))$ bit operations. Plugging $m_{\mathsf{PoKEMath}} = m + t \log(2n)$ and $s_{\mathsf{PoKEMath}} = O(s + t \log n)$, we get a total of $O(\mathsf{MSM}(m + t \log n, \lambda))$ group operations and $O(\mathsf{M}(\lambda)(s + t \log n))$ bit operations.

Proof Size. The proof consists of (i) a single group element for the commitment to the decomposition vector (ii) t integers each of size $\lceil \log \lceil \log(2n) + 5 \rceil \rceil$ (iii) the proof for the invoked PoKEMath which consists of a single group element and $2m_{\mathsf{PoKEMath}}\lambda$ bits, plugging $m_{\mathsf{PoKEMath}} = m + t \log 2n$ (keeping the constants for accuracy), we get $2\lambda(m + t\lceil \log 2n \rceil)$ bits.

By summing up the above, we get a total proof size of 2 group elements and $\lceil \log \lceil \log(2n) + 5 \rceil \rceil + 2\lambda(m + t\lceil \log 2n \rceil)$ bits.

D.3.2 Proof of knowledge-Soundness

Here, we prove the knowledge-soundness of $\mathsf{PoKEDEx}$ using (a) the knowledge-soundness of $\mathsf{PoKEMath}$ which is implied by the modular consistency and strong RSA assumptions (see Theorem 7.4) and, (b) the adaptive root assumption (see Definition A.1) which is also implied by the modular consistency assumption (see Lemma A.2).

Since $\mathsf{PoKEMath}$ is a sub-protocol of $\mathsf{PoKEDEx}$, a pair of adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ for the knowledge-soundness game in $\mathsf{PoKEDEx}$ implies a pair of adversaries $\mathsf{PoKEMath}.\mathcal{A} = (\mathsf{PoKEMath}.\mathcal{A}_1, \mathsf{PoKEMath}.\mathcal{A}_2)$ for the knowledge-soundness game of $\mathsf{PoKEMath}$. $\mathsf{PoKEMath}.\mathcal{A}_1$ invokes Steps 2 to 7 of $\mathcal{A}_1$ to get $\mathbb{x}_{\mathsf{PoKEMath}}$. $\mathsf{PoKEMath}.\mathcal{A}_2$ invokes the rest of $\mathsf{PoKEDEx}$. Due to the knowledge-soundness of $\mathsf{PoKEMath}$, there exists an extractor $\mathsf{PoKEMath.Ext}$ that extracts the $\mathsf{PoKEMath}$ witness with a negligible probability of failure. Now we define $\mathsf{PoKEDEx.Ext}$ that invokes $\mathsf{PoKEMath.Ext}$ and outputs the extracted witness.

$\mathsf{PoKEDEx.Ext}(\mathsf{pp}, \mathbb{x}) \to \mathbb{w}^*$

1. Invoke Steps 2 to 7 of the interaction between $\mathcal{A}_1$ and $\mathcal{V}$ to get $\mathsf{pp}_{\mathsf{PoKEMath}}$ and $\mathbb{x}_{\mathsf{PoKEMath}}$.
2. Invoke the $\mathsf{PoKEMath}$ extractor $(\boldsymbol{u}^*, \boldsymbol{v}^*) := \mathsf{PoKEMath.Ext}(\mathsf{pp}_{\mathsf{PoKEMath}}, \mathbb{x}_{\mathsf{PoKEMath}})$.
3. Output the first part of the $\mathsf{PoKEMath}$ witness: $\mathbb{w}^* := \boldsymbol{u}^*$.

Now, we argue that with overwhelming probability the extracted witness $\mathbb{w}^*$ is a valid witness for $\mathsf{PoKEDEx}$. Since $f'(\boldsymbol{u}, \boldsymbol{v}) = 0$, by definition of f' (see Step 7), we have

(a) $f(\boldsymbol{u}) = 0$
(b) For all $i \in [m-t+1, m]$: $(u_i - a_i)(b_i - u_i) = \langle \boldsymbol{v}_i, \boldsymbol{v}_i \rangle$, where $\boldsymbol{v}_i$ is the i-th sub-vector of $\boldsymbol{v}$ of length m_i. This implies that $(u_i - a_i)(b_i - u_i) \geq 0$ or equivalently $u_i \in [a_i, b_i]$.

It only remains to argue that $\langle \boldsymbol{u}^*, \boldsymbol{G}|_{[1:m]} \rangle = U$ which leverges the stitching technique. Assume for contradiction that $\langle \boldsymbol{u}^*, \boldsymbol{G}|_{[1:m]} \rangle \neq U$. We build the following adversaries for the adaptive root assumption (see Definition A.1).

$\mathsf{AR}.\mathcal{A}_1(\mathbb{G}) \to Y$

1. Invoke $\mathsf{pp} := \mathsf{Setup}(1^\lambda, N)$.
2. Invoke $\mathbb{x}_{\mathsf{PoKEDEx}} := \mathsf{PoKEDEx}.\mathcal{A}_1(\mathsf{pp})$
3. Invoke Steps 2 to 7 of the interaction between $\mathcal{A}_1$ and $\mathcal{V}$ to get $\mathsf{pp}_{\mathsf{PoKEMath}}$, $\mathbb{x}_{\mathsf{PoKEMath}}$, and V.
4. Invoke the $\mathsf{PoKEMath}$ extractor $(\boldsymbol{u}^*, \boldsymbol{v}^*) := \mathsf{PoKEMath.Ext}(\mathsf{pp}_{\mathsf{PoKEMath}}, \mathbb{x}_{\mathsf{PoKEMath}})$.
5. Pass $(\mathsf{pp}, \mathbb{x}_{\mathsf{PoKEDEx}})$ as the state and output $Y := \langle \boldsymbol{v}^*, \boldsymbol{G}|_{[m+1:m+|\boldsymbol{v}|]} \rangle - V$

$\mathsf{AR}.\mathcal{A}_2(\ell) \to X$

1. Parse pp and $\mathbb{x}_{\mathsf{PoKEDEx}} := (U, m, t, (a_i, b_i)_{i=1}^t, f)$ as the state.
2. Invoke the $\mathsf{PoKEDEx}$ extractor $\boldsymbol{u}^* := \mathsf{PoKEDEx.Ext}(\mathsf{pp}, \mathbb{x}_{\mathsf{PoKEDEx}})$.
3. Output $X := U - \langle \boldsymbol{u}^*, \boldsymbol{G}|_{[1:m]} \rangle$

By extractability of $\mathsf{PoKEMath}$, we have $\beta U + V = \langle (\boldsymbol{u}^*, \boldsymbol{v}^*), \boldsymbol{G}' \rangle$ or equivalently

$$\beta U + V = \langle \boldsymbol{u}^*, \beta \boldsymbol{G}|_{[1:m]} \rangle + \langle \boldsymbol{v}^*, \boldsymbol{G}|_{[m+1:m+|\boldsymbol{v}|]} \rangle$$

rearranging the above, we get

$$\langle \boldsymbol{v}^*, \boldsymbol{G}|_{[m+1:m+|\boldsymbol{v}|]} \rangle - V = \beta U - \beta \langle \boldsymbol{u}^*, \boldsymbol{G}|_{[1:m]} \rangle = \beta (U - \langle \boldsymbol{u}^*, \boldsymbol{G}|_{[1:m]} \rangle) \neq 0$$

which implies that $Y \neq 0$ and hence the adaptive root assumption is broken. This gives us a contradiction to the adaptive root assumption, and hence the proof is complete.

D.4 TPoKEDEx deferred proofs

In this section we present the deferred proof of Theorem 7.9; namely, we discuss the cost of the $\mathsf{TPoKEDex}$ protocol, and prove its knowledge-soundness.

D.4.1 Discussion of cost

Prover Complexity. The prover complexity of $\mathsf{TPoKEDex}$ consists of (i) a constant number of operations for stitching (ii) $\mathsf{MSM}(m_{\mathsf{PoKEMath}}, n_{\mathsf{PoKEMath}}) = O(\mathsf{MSM}(1, n))$ group operations and $O(m_{\mathsf{PoKEMath}}\mathsf{M}(n_{\mathsf{PoKEMath}})) = O(5\cdot\mathsf{M}(n)) = O(\mathsf{M}(n))$ bit operations for the unrolled $\mathsf{PoKEMath}$ (iii) $O(\mathsf{M}(n)\log^2(n))$ bit operations for computing the integer square decompositions (iv) $\mathsf{MSM}(3 \times 2^{n_{\mathsf{TIPA}}}, n) = O(\mathsf{MSM}(\log n, n))$ group operations for commiting to the decomposition vector (v) $O(2^{n_{\mathsf{TIPA}}}) = O(\log(2n)) = O(\log n)$ group operations for TIPA.

Hence, a total of $O(\mathsf{MSM}(\log n, n) + \log n)$ group operations and $O(\mathsf{M}(n)\log^2 n)$ bit operations are required.

Verifier Complexity. The verifier complexity of $\mathsf{TPoKEDex}$ consists of (i) a constant number of operations for stitching (ii) $O(\mathsf{MSM}(m_{\mathsf{PoKEMath}}, \lambda)) = O(\mathsf{MSM}(1, \lambda))$ group operations and $O(s \cdot \mathsf{M}(\lambda))$ bit operations for the unrolled $\mathsf{PoKEMath}$ (iii) $O(2^{n_{\mathsf{TIPA}}}) = O(\log 2n) = O(\log n)$ group operations for TIPA.

Hence, a total of $O(\mathsf{MSM}(1, \lambda), \log n)$ group operations and $O(s\mathsf{M}(\lambda))$ bit operations are required.

Proof Size. The proof consists of 1 group element and $2m_{\mathsf{PoKEMath}}\lambda = 10\lambda$ bits for the unrolled $\mathsf{PoKEMath}$, and $2n_{\mathsf{TIPA}} + 2 = 2\lceil\log\lceil\log(2n) + 5\rceil\rceil + 2$ group elements and $6n_{\mathsf{TIPA}}(2\lambda) + 12\lambda = 12\lambda\lceil\log\lceil\log(2n) + 5\rceil\rceil + 12\lambda$ bits for TIPA.

Hence, a total of $2\lceil\log\lceil\log(2n) + 5\rceil\rceil + 3$ group elements and $12\lambda\lceil\log\lceil\log(2n) + 5\rceil\rceil + 22\lambda$ bits is requires.

D.4.2 Proof of knowledge-Soundness

Here, we prove the knowledge-soundness of $\mathsf{TPoKEDex}$ using (a) the knowledge-soundness of TIPA which is implied by the strong RSA and discrete log assumption (see Theorem E.4) and, (b) the knowledge-soundness of PoKE which is implied by the modular consistency assumption (see Theorem 7.2) and, (c) the knowledge-soundness of $\mathsf{PoKEMath}$ which is implied by the modular consistency and strong RSA assumptions (see Theorem 7.4) and, (d) the adaptive root assumption (see Definition A.1) which is implied by the modular consistency assumption (see Lemma A.2).

Assuming that we have polynomial time adversaries $\mathsf{TPoKEDex}.\mathcal{A} := (\mathsf{TPoKEDex}.\mathcal{A}_1, \mathsf{TPoKEDex}.\mathcal{A}_2)$, we build a $\mathsf{TPoKEDex}$ extractor as follows:

TPoKEDex Extractor. Since PoKE is a sub-protocol of $\mathsf{TPoKEDex}$, we can define the PoKE adversary pair $\mathsf{PoKE}.\mathcal{A} = (\mathsf{PoKE}.\mathcal{A}_1, \mathsf{PoKE}.\mathcal{A}_2)$ by using $\mathsf{TPoKEDex}.\mathcal{A}$. Moreover, Due to the knowledge-soundness of PoKE, There exists $\mathsf{PoKE}.\mathsf{Ext}$; Hence, we define $\mathsf{TPoKEDex}.\mathsf{Ext}$ as follows:

$\mathsf{TPoKEDex}.\mathsf{Ext}(\mathsf{pp}, \mathbb{x}) \to \mathbb{w}^*$

---

1. Invoke Steps 2 and 3 of the interaction between $\mathcal{A}_1$ and $\mathcal{V}$ to get

$$\mathsf{pp}_{\mathsf{PoKE}} = (N, \mathbb{G}, \boldsymbol{G}')$$
$$\mathbb{x}_{\mathsf{PoKE}} = (U', m)$$

```
2. Invoke the PoKE extractor and output $u^* := \mathsf{PoKE.Ext}(\mathsf{pp}_{\mathsf{PoKE}}, \mathbb{x}_{\mathsf{PoKE}})$.
```

Now, we have to prove that with overwhelming probability the following hold:

1. $c \cdot G_1 = C$ and $\langle \boldsymbol{u}, (G_1, G_2, G_3, G_4) \rangle = U$
2. $f(c, \boldsymbol{u}) = 0$
3. $u_1 \in [a_1, b_1], u_2 \in [a_2, b_2], u_3 \in [a_3, b_3]$

Proving Item 1. Here, we use the stitching argument, similar to the proof in Appendix D.3.2. By the knowledge-soundness of PoKE, we know that for the extracted (c^*, u^*) with overwhelming probability, $\langle (c^*, u^*), \boldsymbol{G}' \rangle = U'$ which implies

$$\langle c^*, \beta G_1 \rangle + \langle \boldsymbol{u}^*, \boldsymbol{G}|_{[2:5]} \rangle = \beta C + U \implies \langle \boldsymbol{u}^*, \boldsymbol{G}|_{[2:5]} \rangle - U = \beta(C - \langle c^*, G_1 \rangle)$$

This gives us a stretegy to break the adaptive root assumption (see Definition A.1):

```
AR.$\mathcal{A}_1(\mathbb{G}) \to Y$
1. Invoke $\mathbb{x}_{\mathsf{TPoKEDex}} := \mathsf{TPoKEDex}.\mathcal{A}_1(\mathsf{pp})$ where $\mathbb{x}_{\mathsf{TPoKEDex}} = (C, U, (a_i, b_i)_{i=2}^4, f)$.
2. Invoke $\mathbb{w}^* := \mathsf{TPoKEDex.Ext}(\mathsf{pp}, \mathbb{x}_{\mathsf{TPoKEDex}})$ where $\mathbb{w}^* := (c^*, \boldsymbol{u}^*)$.
3. Output $Y := \langle \boldsymbol{u}^*, \boldsymbol{G}|_{[2:5]} \rangle - U$
```

```
AR.$\mathcal{A}_2(\beta)$
1. Output the adaptive root $X := C - \langle c^*, G_1 \rangle$
```

Now, if either $c \cdot G_1 \neq C$ or $\langle \boldsymbol{u}, (G_1, G_2, G_3, G_4) \rangle \neq U$, $Y \neq 0$ and the above adversaries break the adaptive root assumption. Hence, with overwhelming probability, we have

$$c \cdot G_1 = C$$
$$\langle \boldsymbol{u}, \boldsymbol{G}|_{[2:5]} \rangle = U$$

Proving Item 2. Since TPoKEDex runs a full PoKEMath protocol (unrolled), we can use the exact argument for PoKEMath in Appendix D.2. In particular, if single rational root assumption (see Definition A.5) and adaptive root assumption (see Definition A.1) hold, then with overwhelming probability

$$f(c, \boldsymbol{u}) = 0$$

Proving Item 3. In this part, we use the same technique as the one we used to prove Item 2. However, the function being checked is implicitly defined by the TIPA relation, unlike the scenario in PoKEDEx, where the function was explicitly checked in the definition of f. Here, we define $g : \mathbb{Z} \to \mathbb{Z}$ as $g(X) = X - |X|$ which is an indicator for positivity.

Assume for contradiction that there exists $i \in \{2, 3, 4\}$ such that $u_i \notin [a_i, b_i]$ or equivalently $(b_i - u_i)(u_i - a_i) < 0$. Now we can define adaptive root adversaries AR.$\mathcal{A}_1$ and AR.$\mathcal{A}_2$:

```
AR.$\mathcal{A}_1(\mathbb{G}) \to Y$
1. Invoke $\mathsf{pp} := \mathsf{Setup}(1^\lambda, N)$.
2. Invoke $\mathbb{x}_{\mathsf{TPoKEDex}} := \mathsf{TPoKEDex}.\mathcal{A}_1(\mathsf{pp})$
```

```
3. Invoke $(c^*, \boldsymbol{u}^*) := \mathsf{TPoKEDex.Ext}(\mathsf{pp}, \mathbb{x}_{\mathsf{TPoKEDex}})$
4. Set $i \in \{1,2,3\}$ to be the index such that $u_i \notin [a_i, b_i]$.
5. Output $Y = g((b_i - u_i)(u_i - a_i)) \cdot G = 2(b_i - u_i)(u_i - a_i) \cdot G$
```

```
$\mathsf{AR}.\mathcal{A}_2(\ell) \to X$

1. Interact with $\mathsf{TPoKEDex}.\mathcal{A}_2(\mathsf{pp})$ by feeding it a random stitching challenge $\beta \xleftarrow{\$} \mathbb{F}_q$ and PoKE challenge
   $\ell \in \mathbb{F}_q$ to obtain a transcript $\mathsf{tr}$.
2. If $\mathcal{V}$ does not accept on the transcript $\mathsf{tr}$, return to Step 1.
3. Parse $i \in \{1,2,3\}$ from the state.
4. Output $X := (g((b_{i-1} - u_i)(u_i - a_{i-1}))/\ell)G$.
```

Due to the soundness property of TIPA, with overwhelming probability, for all $j \in \{2,3,4\}$ and for some $k_j \in \mathbb{N}$, $m \in \mathbb{N}$, and $\boldsymbol{v}_j \in \mathbb{Z}^m$ we have

$$(b_j - r_j)(r_j - a_j) = \langle \boldsymbol{v}_j, \boldsymbol{v}_j \rangle \mod \ell$$
$$(b_j - r_j)(r_j - a_j) = \langle \boldsymbol{v}_j, \boldsymbol{v}_j \rangle + k_j \ell$$

Due to Lemma D.1, with overwhelming probability we have

$$(b_{i-1} - u_i)(u_i - a_{i-1}) = \langle \boldsymbol{v}_i, \boldsymbol{v}_i \rangle + k_i \ell + k_i' \ell$$

Since $(b_{i-1} - u_i)(u_i - a_{i-1}) < 0$ and $\langle \boldsymbol{v}_i, \boldsymbol{v}_i \rangle \geq 0$, we have

$$\begin{aligned} g((b_{i-1} - u_i)(u_i - a_{i-1})) &= (b_{i-1} - u_i)(u_i - a_{i-1}) - |(b_{i-1} - u_i)(u_i - a_{i-1})| \\ &= \langle \boldsymbol{v}_i, \boldsymbol{v}_i \rangle + k_i \ell + k_i' \ell - |\langle \boldsymbol{v}_i, \boldsymbol{v}_i \rangle + k_i \ell + k_i' \ell| \\ &= \langle \boldsymbol{v}_i, \boldsymbol{v}_i \rangle + k_i \ell + k_i' \ell - \langle \boldsymbol{v}_i, \boldsymbol{v}_i \rangle + |k_i \ell + k_i' \ell| \\ &= 2\ell(k_i + k_i') \end{aligned}$$

Then the above construction for $\mathsf{AR}.\mathcal{A}_2$ is computing $g((b_{i-1} - u_i)(u_i - a_{i-1}))/\ell = 2(k_i + k_i')$ and outputting $X := 2(k_i + k_i')G$ which breaks the adaptive root assumption. Hence, with overwhelming probability, we have $u_i \in [a_i, b_i]$ for all $i \in \{2,3,4\}$. This concludes the proof of knowledge-soundness of TPoKEDex.

E Deferred discussion on inner product argument

E.1 Triple inner product arguments (TIPA)

In this section we first recall the TIPA relation, and proceed to present the corresponding deferred constructions and proofs.

Definition [restated] 1 (TIPA relation). *Given* $\mathsf{pp} = (3N, \mathbb{G}, (\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_3)) \leftarrow \mathsf{Setup}(1^\lambda, 3N)$*, we define the NP relation* $\mathscr{R}^{\mathsf{pp}}_{\mathsf{TIPA}}$ *to be the set of tuples*

$$\begin{pmatrix} \mathbb{x}, \\ \mathbb{w} \end{pmatrix} = \begin{pmatrix} (C, \ell, (v_1, v_2, v_3)), \\ (\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3) \end{pmatrix}$$

where $(\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_3) \in \mathbb{G}^{3N}$*,* $C \in \mathbb{G}$*,* $\ell \in \mathbb{N}$ *is a prime larger than* 2^λ*,* $v_1, v_2, v_3 \in \mathbb{Z}_\ell$*, and* $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3 \in \mathbb{Z}^N$ *such that*

$$C = \langle \mathbf{r}_1, \mathbf{G}_1 \rangle + \langle \mathbf{r}_2, \mathbf{G}_2 \rangle + \langle \mathbf{r}_3, \mathbf{G}_3 \rangle \quad \textit{and} \quad v_i = \langle \mathbf{r}_i, \mathbf{r}_i \rangle \mod \ell \quad \forall\, i \in [3]$$

We construct an argument of knowledge for $\mathscr{R}_{\mathsf{TIPA}}$ by definining a sequence of relations, $\mathscr{R}_{\mathsf{TIPA}\text{-}i}$, for $i \in [0, 1, \ldots, n]$ and constructing reductions of knowledge from $\mathscr{R}_{\mathsf{TIPA}\text{-}i}$ to $\mathscr{R}_{\mathsf{TIPA}\text{-}(i+1)}$, for each $i \in [1, \ldots, n]$.

E.1.1 The TIPA-i relation

As a preliminary we define the following function triplefold, that takes as input a vector $\mathbf{G} \in 3 \cdot \mathbb{G}^{2^n}$, an integer $i \in \mathbb{N}$ and challenges $\boldsymbol{\alpha} \in \mathbb{Z}^i$, and 'folds' $\mathbf{G}$ in half i times with respect to the challenges $\boldsymbol{\alpha}$. For $i = 0$, we define $\mathsf{triplefold}(\mathbf{G}, 0, \perp) := \mathbf{G}$.

```
$\mathsf{triplefold}(\mathbf{G}, i, \boldsymbol{\alpha}) \rightarrow \mathbf{G}'$

1. Set $(\mathbf{G}'_1, \mathbf{G}'_2, \mathbf{G}'_3) := \mathbf{G}$.
2. For each $j \in [i]$:
3.     For each $k \in [3]$:
4.         Fold $\mathbf{G}'_k \leftarrow \alpha_j \cdot \mathbf{G}'_{kL} + \mathbf{G}'_{kR}$.
5. Output $\mathbf{G}' := (\mathbf{G}'_1, \mathbf{G}'_2, \mathbf{G}'_3)$.
```

We now define the following relations for each $i \in [0, 1, \ldots, n]$.

Definition E.1 (TIPA-i relation). *Given* $\mathsf{pp} = (3 \cdot 2^n, \mathbb{G}, (\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_3)) \leftarrow \mathsf{Setup}(1^\lambda, 3 \cdot 2^n)$*, we define the NP relation* $\mathscr{R}^{\mathsf{pp}}_{\mathsf{TIPA}\text{-}i}$ *to be the set of tuples*

$$\begin{pmatrix} \mathbb{x}, \\ \mathbb{w} \end{pmatrix} = \begin{pmatrix} (C, \ell, v, \gamma, \boldsymbol{\alpha}), \\ (\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3) \end{pmatrix}$$

where $C \in \mathbb{G}$*,* $\ell \in \mathbb{N}$ *is a prime larger than* 2^λ*,* $v \in \mathbb{Z}_\ell$*,* $\gamma \in [0, \ell)$*,* $\boldsymbol{\alpha} \in [0, \ell)^i$ *and* $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3 \in \mathbb{Z}^{2^{n-i}}$ *such that for* $(\mathbf{G}'_1, \mathbf{G}'_2, \mathbf{G}'_3) := \mathsf{triplefold}((\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_3), i, \boldsymbol{\alpha})$*:*

$$C = \langle \mathbf{r}_1, \mathbf{G}'_1 \rangle + \langle \mathbf{r}_2, \mathbf{G}'_2 \rangle + \langle \mathbf{r}_3, \mathbf{G}'_3 \rangle \quad \textit{and} \quad v = \sum_{i=1}^{3} \gamma^{i-1} \cdot \langle \mathbf{r}_i, \mathbf{r}_i \rangle \mod \ell$$

Notice that for concrete efficiency, we have 'batched' the three inner product claims using a random challenge γ.

E.1.2 The TIPA-0 reduction

In this section we present a reduction of knowledge $\mathsf{TIPA.Reduce}_0$ from $\mathscr{R}_{\mathsf{TIPA}}$ to $\mathscr{R}_{\mathsf{TIPA}\text{-}0}$.

```
TIPA.Reduce_0

⟨P(pp, x, w), V(pp, x)⟩:

Parse: pp = (3·2^n, G, (G_1, G_2, G_3)), x = (C, ℓ, (v_1, v_2, v_3)) and w = (r_1, r_2, r_3).

1. V samples $\gamma \xleftarrow{\$} \mathbb{Z}_\ell$ and sends it to P.
2. P and V set
       $v' := v_1 + \gamma \cdot v_2 + \gamma^2 \cdot v_3 \mod \ell$
3. Define $\mathbb{x}' := (C, \ell, v', \gamma)$ and $\mathbb{w}' := (\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3)$.
4. P receives output $(\mathbb{x}', \mathbb{w}')$ and V receives output $\mathbb{x}'$.
```

Lemma E.2. *For $\mathcal{G} :=$ Setup and the $\mathcal{P}$, $\mathcal{V}$ above, define* $\mathsf{TIPA.Reduce}_0 := (\mathcal{G}, \mathcal{P}, \mathcal{V})$. *If the discrete log assumption (Definition 3.1) holds,* $\mathsf{TIPA.Reduce}_0$ *is a reduction of knowledge from $\mathscr{R}_{\mathsf{TIPA}}$ to $\mathscr{R}_{\mathsf{TIPA\text{-}0}}$. The prover time is $O(1)$ scalar operations, the verifier time is $O(1)$ scalar operations, and the communication complexity is $1|\mathbb{Z}_\ell|$.*

Proof. Completeness and public reducibility follow from inspection, analogous to the proof of Lemma 8.4.

Knowledge soundness. We prove knowledge soundness via tree extraction (Lemma 2.6), which states that if a candidate reduction of knowledge satisfies completeness and public reducibility, it is enough to show that it satisfied tree extractability. That is, there exists a PPT extractor Ext that for all instances $\mathbb{x} \in \mathscr{R}_{\mathsf{SIPA}\text{-}i}$ outputs a satisfying witness $\mathbb{w}^*$ with probability at least $1 - \text{negl}(\lambda)$, given an n-tree of accepting transcripts for $\mathbb{x}$ where the verifier's randomness is sampled from space Q such that $|Q| = O(2^\lambda)$, and $n = \text{poly}(\lambda)$. In our setting, $n = 3$ and $Q = [0, \ell)$.

For each $j \in [3]$, let the transcript tr_j contain $(\mathbb{x}'_j, \mathbb{w}'_j)$ and the verifier's challenge to the prover, γ_j. For each $j \in [3]$, parsing $\mathbb{x}'_j$ as $(C, \ell, v'_j, \gamma_j)$ and $\mathbb{w}'_j$ as $(\mathbf{r}_{1,j}, \mathbf{r}_{2,j}, \mathbf{r}_{3,j})$, since tr_j is accepting, it must be the case that:

$$\begin{aligned} C &= \langle \mathbf{r}_{1,j}, \mathbf{G}_1 \rangle + \langle \mathbf{r}_{2,j}, \mathbf{G}_2 \rangle + \langle \mathbf{r}_{3,j}, \mathbf{G}_3 \rangle, \\ &\quad \text{and} \\ v'_j &= \langle \mathbf{r}_{1,j}, \mathbf{r}_{1,j} \rangle + \gamma_j \cdot \langle \mathbf{r}_{2,j}, \mathbf{r}_{2,j} \rangle + \gamma_j^2 \cdot \langle \mathbf{r}_{3,j}, \mathbf{r}_{3,j} \rangle \mod \ell \\ \implies v_1 + \gamma_j \cdot v_2 + \gamma_j^2 \cdot v_3 &= \langle \mathbf{r}_{1,j}, \mathbf{r}_{1,j} \rangle + \gamma_j \cdot \langle \mathbf{r}_{2,j}, \mathbf{r}_{2,j} \rangle + \gamma_j^2 \cdot \langle \mathbf{r}_{3,j}, \mathbf{r}_{3,j} \rangle \mod \ell \\ \implies 0 &= (\langle \mathbf{r}_{1,j}, \mathbf{r}_{1,j} \rangle - v_1) + \gamma_j \cdot (\langle \mathbf{r}_{2,j}, \mathbf{r}_{2,j} \rangle - v_2) + \gamma_j^2 \cdot (\langle \mathbf{r}_{3,j}, \mathbf{r}_{3,j} \rangle - v_3) \mod \ell \end{aligned}$$

It must be the case that, except with negligible probability, $\mathbf{r}_{i,1} = \mathbf{r}_{i,2} = \mathbf{r}_{i,3}$ for all $i \in [3]$. If not, we could break the discrete logarithm assumption (Definition 3.1) for $(\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_4)$, which is a uniformly random vector sampled from $\mathbb{G}^{3 \cdot 2^n}$. Thus, we can define $\mathbf{r}_i := \mathbf{r}_{i,1}$, for all $i \in [3]$.

We can now say that for each $j \in [3]$:

$$0 = (\langle \mathbf{r}_1, \mathbf{r}_1 \rangle - v_1) + \gamma_j \cdot (\langle \mathbf{r}_2, \mathbf{r}_2 \rangle - v_2) + \gamma_j^2 \cdot (\langle \mathbf{r}_3, \mathbf{r}_3 \rangle - v_3) \mod \ell$$

We can define a polynomial $p(X) \in \mathbb{Z}_\ell[X]$ of degree 2 to be $p(X) := (\langle \mathbf{r}_1, \mathbf{r}_1 \rangle - v_1) + X \cdot (\langle \mathbf{r}_2, \mathbf{r}_2 \rangle - v_2) + X^2 \cdot (\langle \mathbf{r}_3, \mathbf{r}_3 \rangle - v_3)$. For each $j \in [4]$, define $\beta'_j \in \mathbb{Z}_\ell$ to be $\beta'_j := \beta_j \mod \ell$, and notice that each β'_j is a distinct element of $\mathbb{Z}_\ell$. Since for all $j \in [4]$, $p(\beta'_j) = 0$, the coefficients of the above polynomial must all be 0 (as elements of $\mathbb{Z}_\ell$), implying that $v_j = \langle \mathbf{r}_j, \mathbf{r}_j \rangle \mod \ell$, for each $j \in [3]$. Thus a PPT extractor Ext can output this satisfying witness $(\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3)$ corresponding to $\mathbb{x}$ with probability at least $1 - \text{negl}(\lambda)$. □

E.1.3 The TIPA-i reductions

We now present a reduction of knowledge $\mathsf{TIPA.Reduce}_i$ from $\mathscr{R}_{\mathsf{TIPA}\text{-}i}$ to $\mathscr{R}_{\mathsf{TIPA}\text{-}(i+1)}$, for each $i \in [1, \ldots, n]$.

TIPA.Reduce$_i$

```
⟨$\mathcal{P}(\mathsf{pp}, \mathbb{x}, \mathbb{w}), \mathcal{V}(\mathsf{pp}, \mathbb{x})$⟩:
**Parse:** $\mathsf{pp} = (3 \cdot 2^n, \mathbb{G}, (\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_3))$, $\mathbb{x} = (C, \ell, v, \gamma, \boldsymbol{\alpha})$ and $\mathbb{w} = (\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3)$.
1. $\mathcal{P}$ and $\mathcal{V}$ compute $(\mathbf{G}'_1, \mathbf{G}'_2, \mathbf{G}'_3) := \mathsf{triplefold}((\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_3), i, \boldsymbol{\alpha})$.
2. $\mathcal{P}$ computes $C_L := \sum_{j=1}^{3} \langle \mathbf{r}_{jL}, \mathbf{G}'_{jR} \rangle$ and $C_R := \sum_{j=1}^{3} \langle \mathbf{r}_{jR}, \mathbf{G}'_{jL} \rangle$.
3. For $j \in [3]$: $\mathcal{P}$ computes $v_{jL} := \langle \mathbf{r}_{jL}, \mathbf{r}_{jL} \rangle \mod \ell$ and $v_{jC} := \langle \mathbf{r}_{jL}, \mathbf{r}_{jR} \rangle \mod \ell$.
4. $\mathcal{P}$ computes
       $v_L := v_{1L} + \gamma \cdot v_{2L} + \gamma^2 \cdot v_{3L} \mod \ell$
       $v_C := v_{1C} + \gamma \cdot v_{2C} + \gamma^2 \cdot v_{3C} \mod \ell$
5. $\mathcal{P}$ sends $C_L, C_R, [v_{jL}]_{j \in [3]}, [v_{jC}]_{j \in [3]}$ to $\mathcal{V}$.
6. $\mathcal{V}$ samples $\alpha_{i+1} \xleftarrow{\$} [0, 2^\lambda]$ and sends it to $\mathcal{P}$.
7. For $j \in [3]$:
8.     $\mathcal{P}$ sets $\mathbf{r}'_j := \mathbf{r}_{jL} + \alpha_{i+1} \cdot \mathbf{r}_{jR}$.
9.     $\mathcal{P}$ and $\mathcal{V}$ set
           $v_R := v - v_L \mod \ell,$
           $v' := v_L + 2 \cdot \alpha \cdot v_C + \alpha^2 \cdot v_R \mod \ell$
10. $\mathcal{P}$ and $\mathcal{V}$ set $C' := C_L + \alpha \cdot C + \alpha^2 \cdot C_R$.
11. Define $\mathbb{x}' := (C', \ell, v', \gamma, [\alpha_j]_{j=1}^{i+1})$ and $\mathbb{w}' := (\mathbf{r}'_1, \mathbf{r}'_2, \mathbf{r}'_3)$.
12. $\mathcal{P}$ receives output $(\mathbb{x}', \mathbb{w}')$ and $\mathcal{V}$ receives output $\mathbb{x}'$.
```

Lemma E.3. *For $\mathcal{G} :=$ Setup and the $\mathcal{P}$, $\mathcal{V}$ above, define* TIPA.Reduce$_i$ $:= (\mathcal{G}, \mathcal{P}, \mathcal{V})$*. If the discrete log assumption (Definition 3.1) and the multi-rational root assumption (Definition A.7) hold,* TIPA.Reduce$_i$ *is a reduction of knowledge from $\mathscr{R}_{\mathsf{TIPA}\text{-}i}$ to $\mathscr{R}_{\mathsf{TIPA}\text{-}(i+1)}$. The prover time is $O(2^n)$ group operations, the verifier time is $O(2^n)$ group operations, and the communication complexity is $2|\mathbb{G}|+6|\mathbb{Z}_\ell|$.*

Proof sketch. The proof of the aforementioned lemma follows similarly to that of Lemma 8.4, with the following modification.

In the proof of security of SIPA.Reduce$_i$, the transcripts were used to say:

$$\begin{bmatrix} 1 & \beta_1 & \beta_1^2 \\ 1 & \beta_2 & \beta_2^2 \\ 1 & \beta_3 & \beta_3^2 \end{bmatrix} \begin{bmatrix} C_L \\ C \\ C_R \end{bmatrix} = \begin{bmatrix} \beta_1 \cdot \mathbf{r}_1 & \mathbf{r}_1 \\ \beta_2 \cdot \mathbf{r}_2 & \mathbf{r}_2 \\ \beta_3 \cdot \mathbf{r}_3 & \mathbf{r}_3 \end{bmatrix} \mathbf{G}'$$

In the case of TIPA.Reduce$_i$, the transcripts will give us:

$$\begin{bmatrix} 1 & \beta_1 & \beta_1^2 \\ 1 & \beta_2 & \beta_2^2 \\ 1 & \beta_3 & \beta_3^2 \end{bmatrix} \begin{bmatrix} C_L \\ C \\ C_R \end{bmatrix} = \begin{bmatrix} \beta_1 \cdot \mathbf{r}_{1,1} & \mathbf{r}_{1,1} \\ \beta_2 \cdot \mathbf{r}_{1,2} & \mathbf{r}_{1,2} \\ \beta_3 \cdot \mathbf{r}_{1,3} & \mathbf{r}_{1,3} \end{bmatrix} \mathbf{G}'_1 + \begin{bmatrix} \beta_1 \cdot \mathbf{r}_{2,1} & \mathbf{r}_{2,1} \\ \beta_2 \cdot \mathbf{r}_{2,2} & \mathbf{r}_{2,2} \\ \beta_3 \cdot \mathbf{r}_{2,3} & \mathbf{r}_{2,3} \end{bmatrix} \mathbf{G}'_2 + \begin{bmatrix} \beta_1 \cdot \mathbf{r}_{3,1} & \mathbf{r}_{3,1} \\ \beta_2 \cdot \mathbf{r}_{3,2} & \mathbf{r}_{3,2} \\ \beta_3 \cdot \mathbf{r}_{3,3} & \mathbf{r}_{3,3} \end{bmatrix} \mathbf{G}'_3$$

Again, for notational convenience we define:

For notational convenience, we define

$$\boldsymbol{M} := \begin{bmatrix} 1 & \beta_1 & \beta_1^2 \\ 1 & \beta_2 & \beta_2^2 \\ 1 & \beta_3 & \beta_3^2 \end{bmatrix} \quad \text{and} \quad \boldsymbol{R}_i := \begin{bmatrix} \beta_1 \cdot \mathbf{r}_{j,1} & \mathbf{r}_{j,1} \\ \beta_2 \cdot \mathbf{r}_{j,2} & \mathbf{r}_{j,2} \\ \beta_3 \cdot \mathbf{r}_{j,3} & \mathbf{r}_{j,3} \end{bmatrix} \quad \forall j \in [3]$$

As before, one can compute $\boldsymbol{P} := \mathsf{adj}(\boldsymbol{M})$ such that $\boldsymbol{PM} = \det(\boldsymbol{M}) \cdot \boldsymbol{I}_3$. Defining $m := \det(\boldsymbol{M})$, this gives that:

$$m \cdot \begin{bmatrix} C_L \\ C \\ C_R \end{bmatrix} = \boldsymbol{PR}_1\mathbf{G}'_1 + \boldsymbol{PR}_2\mathbf{G}'_2 + \boldsymbol{PR}_3\mathbf{G}'_3$$

The multi-rational root assumption (Definition A.7) implies that, except with negligible probability, m must divide $\boldsymbol{PR}_1$, $\boldsymbol{PR}_2$ and $\boldsymbol{PR}_3$. This is because if not, this implies the existence of an efficient adversary $\mathcal{A}(\mathbb{G}, \mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_3)$ that with non-negligible can output $(U, m, \mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3)$ such that $m \cdot U = \mathbf{r}_1 \cdot \mathbf{G}_1 + \mathbf{r}_2 \cdot \mathbf{G}_2 + \mathbf{r}_3 \cdot \mathbf{G}_3$, where $m > 1$ and for $j \in [3]$ there exists at least one k_j such that m and r_{j,k_j} are coprime.

Along the same lines as the proof of Lemma A.8, we can use this adversary $\mathcal{A}$ to build an adversary $\mathcal{B}(\mathbb{G}, \mathbf{G})$ that breaks the multi-rational root assumption. Given $\mathbf{G}$, $\mathcal{B}$ samples $\alpha_1, \alpha_2, \alpha_3 \xleftarrow{\$} [0, 3B')$, where B' is an upperbound on the order of $\mathbb{G}$, and for each $j \in [3]$ sets $\mathbf{G}_j := \alpha_j \cdot \mathbf{G}$. As before, with probability at least $\left(1 - \frac{1}{3}\right)^3 \geq \frac{1}{4}$ over the choice of α_1, α_2 and α_3, $\mathbf{G}_1$, $\mathbf{G}_2$ and $\mathbf{G}_3$ look truly random. Thus $\mathcal{B}$ can just obtain $(U, m, \mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3) \leftarrow \mathcal{A}(\mathbb{G}, \mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_3)$ such that $m \cdot U = (\alpha_1\mathbf{r}_1 + \alpha_2\mathbf{r}_2 + \alpha_3\mathbf{r}_3) \cdot \mathbf{G}$ and output $(U, m, \mathbf{r} := \alpha_1\mathbf{r}_1 + \alpha_2\mathbf{r}_2 + \alpha_3\mathbf{r}_3)$. Clearly $\mathcal{B}$ succeeds with non-negligible probability, thus breaking the multi-rational root assumption.

The argument to show that the extracted $(\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3)$ satisfies $v = \sum_{i=1}^{3} \gamma^{i-1} \cdot \langle \mathbf{r}_i, \mathbf{r}_i \rangle \mod \ell$ is a straightforward adaptation of the one in the proof of Lemma 8.4.

E.1.4 The full TIPA protocol

We now present the full argument for $\mathscr{R}^{\mathsf{pp}}_{\mathsf{TIPA}}$ for a given $\mathsf{pp} = (3 \cdot 2^n, \mathbb{G}, (\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_3))$, which applies $\mathsf{TIPA.Reduce}$ iteratively to shrink the size of the instance to length 3, and which is directly checked.

TIPA

$\langle \mathcal{P}(\mathsf{pp}, \mathbb{x}, \mathbb{w}), \mathcal{V}(\mathsf{pp}, \mathbb{x}) \rangle$:

Parse: $\mathsf{pp} = (3 \cdot 2^n, \mathbb{G}, (\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_3))$, $\mathbb{x} = (C, \ell, (v_1, v_2, v_3))$ and $\mathbb{w} = (\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3)$.

```
1. Define $\mathbb{x}_0 := \mathbb{x}$ and $\mathbb{w}_0 := \mathbb{w}$.
2. For $i$ in $[0, \dots, n-1]$:
3.     $(\mathbb{x}_{i+1}, \mathbb{w}_{i+1}) \leftarrow \mathsf{TIPA.Reduce}_i(\langle \mathcal{P}(\mathsf{pp}, \mathbb{x}_i, \mathbb{w}_i), \mathcal{V}(\mathsf{pp}, \mathbb{x}_i) \rangle)$.
4. Parse $\mathbb{x}_n = (C', \ell, (v'_1, v'_2, v'_3), \boldsymbol{\alpha})$ and $\mathbb{w}_n = (r'_1, r'_2, r'_3)$.
5. $\mathcal{P}$ sends $(r'_1, r'_2, r'_3)$ to $\mathcal{V}$.
6. $\mathcal{V}$ accepts if for $(G'_1, G'_2, G'_3) \leftarrow \mathsf{triplefold}((\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_3), n, \boldsymbol{\alpha})$:
```

$$C' = r'_1 \cdot G'_1 + r'_2 \cdot G'_2 + r'_3 \cdot G'_3, \text{ and}$$
$$v'_j = r'_j \cdot r'_j \mod \ell \quad \forall j \in [3]$$

Theorem E.4. *If the discrete log assumption (Definition 3.1) and the multi-rational root assumption (Definition A.7) hold, then* $\mathsf{TIPA} := (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$*, with* $\mathcal{P}$ *and* $\mathcal{V}$ *as above, is an argument of knowledge for* $\mathscr{R}_{\mathsf{TIPA}}$*. The prover time is* $O(2^n)$ *group operations, the verifier time is* $O(2^n)$ *group operations, and the communication complexity is* $2n|\mathbb{G}| + 6n|\mathbb{Z}_\ell| + \sum_{j=1}^{3} |r'_j|$.

The proof of this theorem is analogous to that of Theorem 8.5. Once again, as an optimization we can use $\mathsf{PoKEMath}$ at the end instead of directly sending (r'_1, r'_2, r'_3) to the verifier, bringing the communication complexity down to

$$(2n+1)|\mathbb{G}| + 6n|\mathbb{Z}_\ell| + \mathsf{cost}(\mathsf{PoKEMath}) \leq (2n+2)|\mathbb{G}| + 6n|\mathbb{Z}_\ell| + 12\lambda.$$

E.2 Deferred self inner product argument (SIPA) proofs

In this section we present the deferred knowledge soundness proof of Theorem 7.2.

E.2.1 Proof of Lemma 2

In this section we present the deferred proof of Lemma 8.4.

Proof. **Completeness.** Completeness follows directly from the construction of $\mathsf{SIPA.Reduce}_i$. If $(\mathbb{x}, \mathbb{w}) \in \mathscr{R}^{\mathsf{pp}}_{\mathsf{SIPA}\text{-}i}$, then parsing $\mathbb{x}$ as $(C, \ell, v, \boldsymbol{\alpha} = [\alpha_j]_{j=1}^{i})$, $\mathbb{w}$ as $\mathbf{r}$ and setting $\mathbf{G}' := \mathsf{fold}(\mathbf{G}, i, \boldsymbol{\alpha})$, it must be the case that $C = \langle \mathbf{r}, \mathbf{G}' \rangle$ and $v = \langle \mathbf{r}, \mathbf{r} \rangle$. Given any $\alpha_{i+1} \in [0, 2^{\lambda}]$, we can set:

$$\begin{aligned}
C' &:= C_L + \alpha_{i+1} \cdot C + \alpha_{i+1}^2 \cdot C_R, \\
v_R &:= v - v_L \mod \ell, \\
v' &:= v_L + 2 \cdot \alpha_{i+1} \cdot v_C + \alpha_{i+1}^2 \cdot v_R \mod \ell, \\
\mathbf{r}' &:= \mathbf{r}_L + \alpha_{i+1} \cdot \mathbf{r}_R
\end{aligned}$$

Setting $\mathbb{x}' := (C', \ell, v', , [\alpha_j]_{j=1}^{i+1})$ and $\mathbb{w}' := \mathbf{r}'$, it is easy to see that given the verifier challenge α_{i+1}, $\mathsf{SIPA.Reduce}$ $(\langle \mathcal{P}(\mathsf{pp}, \mathbb{x}, \mathbb{w}), \mathcal{V}(\mathsf{pp}, \mathbb{x}) \rangle)$ outputs precisely $(\mathbb{x}', \mathbb{w}')$. Defining $\mathbf{G}'' := \mathsf{fold}(\mathbf{G}, i+1, [\alpha_j]_{j=1}^{i+1})$ we now have that:

$$\begin{aligned}
C' &= \langle \mathbf{r}_L, \mathbf{G}_R \rangle + \alpha_{i+1} \cdot C + \alpha_{i+1}^2 \cdot \langle \mathbf{r}_R, \mathbf{G}_L \rangle = \langle \mathbf{r}', \mathbf{G}'' \rangle, \\
v' &= \langle \mathbf{r}_L, \mathbf{r}_L \rangle + 2 \cdot \alpha_{i+1} \cdot \langle \mathbf{r}_L, \mathbf{r}_R \rangle + \alpha_{i+1}^2 \cdot \langle \mathbf{r}_R, \mathbf{r}_R \rangle \mod \ell = \langle \mathbf{r}', \mathbf{r}' \rangle \mod \ell
\end{aligned}$$

Thus we have that $(\mathbb{x}', \mathbb{w}') \in \mathscr{R}^{\mathsf{pp}}_{\mathsf{SIPA}\text{-}(i+1)}$, as required.

Public reducibility. Public reducibility can be shown as follows: $\varphi(\mathsf{pp}, \mathbb{x}, \mathsf{tr}) \to \mathbb{x}'$ merely parses $\mathbb{x}$ as $(C, \ell, v, \boldsymbol{\alpha} = [\alpha_j]_{j=1}^{i})$, obtains C', v' and α_{i+1} from tr, and outputs $\mathbb{x}' := (C', \ell, v', \boldsymbol{\alpha}' := [\alpha_j]_{j=1}^{i+1})$.

Knowledge soundness. We prove knowledge soundness via tree extraction (Lemma 2.6), which states that if a candidate reduction of knowledge satisfies completeness and public reducibility, it is enough to show that it satisfied tree extractability. That is, there exists a PPT extractor Ext that for all instances $\mathbb{x} \in \mathscr{R}_{\mathsf{SIPA}\text{-}i}$ outputs a satisfying witness $\mathbb{w}^*$ with probability at least $1 - \text{negl}(\lambda)$, given an n-tree of accepting transcripts for $\mathbb{x}$ where the verifier's randomness is sampled from space Q such that $|Q| = O(2^{\lambda})$, and $\prod_i n_i = \text{poly}(\lambda)$. In our setting, $n = 4$ and $Q = [0, \ell)$.

For each $j \in [4]$, let the transcript tr_j contain $(\mathbb{x}'_j, \mathbb{w}'_j) \leftarrow \mathsf{SIPA.Reduce}_i(\langle \mathcal{P}(\mathsf{pp}, \mathbb{x}, \mathbb{w}), \mathcal{V}(\mathsf{pp}, \mathbb{x}) \rangle)$, C_L, C_R, v_L and v_C, where the verifier's challenge to the prover is β_j. For each $j \in [4]$, parsing $\mathbb{x}'_j$ as $(C_j, \ell, v_j, ([\alpha_k]_{k=1}^{i}, \beta_j))$[8] and $\mathbb{w}'_j$ as $\mathbf{r}_j$, since tr_j is accepting, it must be the case that for $\mathbf{G}'_j := \mathsf{fold}(\mathbf{G}, i+1, ([\alpha_k]_{k=1}^{i}, \beta_j))$ and $\mathbf{G}' := \mathsf{fold}(\mathbf{G}, i, [\alpha_k]_{k=1}^{i})$:

$$\begin{aligned}
C_j &= \langle \mathbf{r}_j, \mathbf{G}'_j \rangle \\
\implies C_L + \beta_j \cdot C + \beta_j^2 \cdot C_R &= \langle \mathbf{r}_j, \beta_j \cdot \mathbf{G}'_L + \mathbf{G}'_R \rangle, \text{ and} \\
v_j &= \langle \mathbf{r}_j, \mathbf{r}_j \rangle \mod \ell \\
\implies v_L + 2 \cdot \beta_j \cdot v_C + \beta_j^2 \cdot v_R &= \langle \mathbf{r}_j, \mathbf{r}_j \rangle \mod \ell
\end{aligned}$$

The first equation, for $j \in [3]$, can be rewritten as:

$$\begin{bmatrix} 1 & \beta_1 & \beta_1^2 \\ 1 & \beta_2 & \beta_2^2 \\ 1 & \beta_3 & \beta_3^2 \end{bmatrix} \begin{bmatrix} C_L \\ C \\ C_R \end{bmatrix} = \begin{bmatrix} \beta_1 \cdot \mathbf{r}_1 & \mathbf{r}_1 \\ \beta_2 \cdot \mathbf{r}_2 & \mathbf{r}_2 \\ \beta_3 \cdot \mathbf{r}_3 & \mathbf{r}_3 \end{bmatrix} \mathbf{G}'$$

[8]Assume that any element v_k, for any subscript k, in the rest of this proof is an element of $\mathbb{Z}_{\ell}$. We sometimes omit writing the necessary $\mod \ell$ for brevity.

For notational convenience, we now define

$$\boldsymbol{M} := \begin{bmatrix} 1 & \beta_1 & \beta_1^2 \\ 1 & \beta_2 & \beta_2^2 \\ 1 & \beta_3 & \beta_3^2 \end{bmatrix} \quad \text{and} \quad \boldsymbol{R} := \begin{bmatrix} \beta_1 \cdot \mathbf{r}_1 & \mathbf{r}_1 \\ \beta_2 \cdot \mathbf{r}_2 & \mathbf{r}_2 \\ \beta_3 \cdot \mathbf{r}_3 & \mathbf{r}_3 \end{bmatrix}$$

Let $\boldsymbol{P} := \mathsf{adj}(\boldsymbol{M}) \in \mathbb{Z}^{3\times 3}$, the adjugate matrix of $\boldsymbol{M}$ such that $\boldsymbol{PM} = \det(\boldsymbol{M}) \cdot \boldsymbol{I}_3$, where $\det(\boldsymbol{M})$ is the determinant of the matrix $\boldsymbol{M}$ and $\boldsymbol{I}_3 \in \mathbb{Z}^{3\times 3}$ is the identity matrix. Defining $m := \det(\boldsymbol{M})$, it is now easy to see that:

$$\boldsymbol{PM} \begin{bmatrix} C_L \\ C \\ C_R \end{bmatrix} = m \cdot \begin{bmatrix} C_L \\ C \\ C_R \end{bmatrix} = (\boldsymbol{PR})\, \mathbf{G}'$$

Let $\boldsymbol{P}_1$, $\boldsymbol{P}_2$ and $\boldsymbol{P}_3$ be the rows of $\boldsymbol{P}$. It is easy to see that $\boldsymbol{a}' := \boldsymbol{P}_1\boldsymbol{R}$, $\mathbf{r}' := \boldsymbol{P}_2\boldsymbol{R}$ and $\boldsymbol{b}' := \boldsymbol{P}_3\boldsymbol{R}$ are vectors in $\in \mathbb{Z}^{2^{n-i}}$ such that $m \cdot C_L = \langle \boldsymbol{a}', \mathbf{G}' \rangle$, $m \cdot C = \langle \mathbf{r}', \mathbf{G}' \rangle$ and $m \cdot C_R = \langle \boldsymbol{b}', \mathbf{G}' \rangle$. Additionally, the extractor Ext can compute these vectors in $\boldsymbol{a}$, $\mathbf{r}$ and $\boldsymbol{b}$ in polynomial time.

The multi-rational root assumption (Definition A.7) implies that, except with negligible probability, m must divide $\boldsymbol{a}'$, $\mathbf{r}'$ and $\boldsymbol{b}'$. Thus we can define $\boldsymbol{a} := \boldsymbol{a}'/m \in \mathbb{Z}^{2^{n-i}}$, $\mathbf{r} := \mathbf{r}'/m \in \mathbb{Z}^{2^{n-i}}$ and $\boldsymbol{b} := \boldsymbol{b}'/m \in \mathbb{Z}^{2^{n-i}}$, such that $C_L = \langle \boldsymbol{a}, \mathbf{G}' \rangle$, $C = \langle \mathbf{r}, \mathbf{G}' \rangle$ and $C_R = \langle \boldsymbol{b}, \mathbf{G}' \rangle$.

Now, using all 4 transcripts, we can write:

$$\begin{aligned} \begin{bmatrix} 1 & \beta_1 & \beta_1^2 \\ 1 & \beta_2 & \beta_2^2 \\ 1 & \beta_3 & \beta_3^2 \\ 1 & \beta_4 & \beta_4^2 \end{bmatrix} \begin{bmatrix} \boldsymbol{a} \\ \mathbf{r} \\ \boldsymbol{b} \end{bmatrix} \mathbf{G}' &= \begin{bmatrix} \beta_1 \cdot \mathbf{r}_1 & \mathbf{r}_1 \\ \beta_2 \cdot \mathbf{r}_2 & \mathbf{r}_2 \\ \beta_3 \cdot \mathbf{r}_3 & \mathbf{r}_3 \\ \beta_4 \cdot \mathbf{r}_4 & \mathbf{r}_4 \end{bmatrix} \mathbf{G}' \\ \implies \begin{bmatrix} \boldsymbol{a} + \beta_1 \cdot \mathbf{r} + \beta_1^2 \cdot \boldsymbol{b} \\ \boldsymbol{a} + \beta_2 \cdot \mathbf{r} + \beta_2^2 \cdot \boldsymbol{b} \\ \boldsymbol{a} + \beta_3 \cdot \mathbf{r} + \beta_3^2 \cdot \boldsymbol{b} \\ \boldsymbol{a} + \beta_4 \cdot \mathbf{r} + \beta_4^2 \cdot \boldsymbol{b} \end{bmatrix} \mathbf{G}' &= \begin{bmatrix} \beta_1 \cdot \mathbf{r}_1 & \mathbf{r}_1 \\ \beta_2 \cdot \mathbf{r}_2 & \mathbf{r}_2 \\ \beta_3 \cdot \mathbf{r}_3 & \mathbf{r}_3 \\ \beta_4 \cdot \mathbf{r}_4 & \mathbf{r}_4 \end{bmatrix} \mathbf{G}' \end{aligned}$$

It must be the case that, except with negligible probability, $\boldsymbol{a} + \beta_j \cdot \mathbf{r} + \beta_j^2 \cdot \boldsymbol{b} = (\beta_j \cdot \mathbf{r}_i || \mathbf{r}_i)$ for each $j \in [4]$. If not, Ext (which runs in polynomial time) could use $\boldsymbol{a} + \beta_j \cdot \mathbf{r} + \beta_j^2 \cdot \boldsymbol{b} \neq (\beta_j \cdot \mathbf{r}_i || \mathbf{r}_i)$ to break the discrete logarithm assumption (Definition 3.1) for $\mathbf{G}'$, which is a uniformly random vector sampled from $\mathbb{G}^{2^{n-i}}$. Comparing the left half of each of these vectors with the right half multiplied by β_j, we get:

$$\begin{aligned} & \boldsymbol{a}_L + \beta_j \cdot \mathbf{r}_L + \beta_j^2 \cdot \boldsymbol{b}_L = \beta_j \cdot (\boldsymbol{a}_R + \beta_j \cdot \mathbf{r}_R + \beta_j^2 \cdot \boldsymbol{b}_R) \\ \implies & \boldsymbol{a}_L + \beta_j \cdot \mathbf{r}_L + \beta_j^2 \cdot \boldsymbol{b}_L = \beta_j \cdot \boldsymbol{a}_R + \beta_j^2 \cdot \mathbf{r}_R + \beta_j^3 \cdot \boldsymbol{b}_R \\ \implies & \boldsymbol{a}_L + (\mathbf{r}_L - \boldsymbol{a}_R) \cdot \beta_j + (\boldsymbol{b}_L - \mathbf{r}_R) \cdot \beta_j^2 - \boldsymbol{b}_R \cdot \beta_j^3 = 0 \end{aligned}$$

Defining the polynomial $p(X) \in \mathbb{Z}^{2^{n-i-1}}[X]$ of degree 3 to be $p(X) := \boldsymbol{a}_L + (\mathbf{r}_L - \boldsymbol{a}_R) \cdot X + (\boldsymbol{b}_L - \mathbf{r}_R) \cdot X - \boldsymbol{b}_R \cdot X$, this implies that $p(\beta_j) = 0$ for each $j \in [4]$. This implies that the coefficients of the polynomial must all be 0 and thus $\mathbf{r}_L = \boldsymbol{a}_R$, $\mathbf{r}_R = \boldsymbol{b}_L$ and $\boldsymbol{a}_L = \boldsymbol{b}_R = 0^{2^{n-i-1}}$. Plugging this back into the original equation, we get that for each $j \in [4]$,

$$\mathbf{r}_L + \beta_j \cdot \mathbf{r}_R = \mathbf{r}_j$$

By definition, $C = \langle \mathbf{r}, \mathbf{G}' \rangle$, so we just have to show that $\langle \mathbf{r}', \mathbf{r}' \rangle = v$. Recall that for each $j \in [4]$:

$$v_L + 2 \cdot \beta_j \cdot v_C + \beta_j^2 \cdot v_R = \langle \mathbf{r}_j, \mathbf{r}_j \rangle \mod \ell$$

$$\implies v_L + 2 \cdot \beta_j \cdot v_C + \beta_j 2 \cdot v_R = \langle \mathbf{r}_L, \mathbf{r}_L \rangle + 2 \cdot \beta_j \cdot \langle \mathbf{r}_L, \mathbf{r}_R \rangle + \beta_j^2 \cdot \langle \mathbf{r}_R, \mathbf{r}_R \rangle \mod \ell$$
$$\implies (v_L - \langle \mathbf{r}_L, \mathbf{r}_L \rangle) + 2 \cdot (v_C - \langle \mathbf{r}_L, \mathbf{r}_R \rangle)\beta_j + (v_R - \langle \mathbf{r}_R, \mathbf{r}_R \rangle)\beta_j^2 = 0 \mod \ell$$

As before, we can define a polynomial $p(X) \in \mathbb{Z}_\ell[X]$ of degree 2 to be $p(X) := (v_L - \langle \mathbf{r}_L, \mathbf{r}_L \rangle) + 2 \cdot (v_C - \langle \mathbf{r}_L, \mathbf{r}_R \rangle)X + (v_R - \langle \mathbf{r}_R, \mathbf{r}_R \rangle)X^2$. For each $j \in [4]$, define $\beta'_j \in \mathbb{Z}_\ell$ to be $\beta'_j := \beta_j \mod \ell$, and notice that each β'_j is a distinct element of $\mathbb{Z}_\ell$. Since for all $j \in [4]$, $p(\beta'_j) = 0$, the coefficients of the above polynomial must all be 0 (as elements of $\mathbb{Z}_\ell$), implying that: $v_L = \langle \mathbf{r}_L, \mathbf{r}_L \rangle \mod \ell$, $v_C = \langle \mathbf{r}_L, \mathbf{r}_R \rangle \mod \ell$ and $v_R = \langle \mathbf{r}_R, \mathbf{r}_R \rangle \mod \ell$. Thus $\langle \mathbf{r}, \mathbf{r} \rangle = \langle \mathbf{r}_L, \mathbf{r}_L \rangle + \langle \mathbf{r}_R, \mathbf{r}_R \rangle = v_L + v_R \mod \ell$. However, by definition, $v_R = v - v_L \mod \ell$, implying that $v = \langle \mathbf{r}, \mathbf{r} \rangle \mod \ell$.

To summarize, the extractor Ext works as follows:

```
$\mathsf{Ext}([\mathsf{tr}_j]_{j=1}^4) \to \mathbb{w}^*$

1. For each $j \in [4]$:
2.     Obtain $\mathbb{x}'_j = (C_j, \ell, v_j, ([\alpha_k]_{k=1}^i, \beta_j))$ and $\mathbb{w}'_j = \mathbf{r}_j$ from $\mathsf{tr}_j$.
3. Use $[\beta_j]_{j=1}^3$ and $[\mathbf{r}_j]_{j=1}^3$ to define the matrices $\boldsymbol{M} \in \mathbb{Z}^{3\times 3}$ and $\boldsymbol{R} \in \mathbb{Z}^{3\times 2^{n-i}}$.
4. Compute $\boldsymbol{P} := \mathsf{adj}(\boldsymbol{M})$ and $m := \det(\boldsymbol{M})$.
5. Compute $\mathbf{r} := \boldsymbol{P}_2\boldsymbol{R}/m$.
6. Output $\mathbb{w}^* := \mathbf{r}$.
```

Efficiency. The efficiency claims follow from inspection.

□